FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Algorithm Selection and Performance Prediction for the Examination Timetabling Problem

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Computational Intelligence

eingereicht von

DI Olga Zhukova
Matrikelnummer 1228302

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Priv.-Doz. Dr. Nysret Musliu

Wien, 1. März 2018

Olga Zhukova	Nysret Musliu

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Algorithm Selection and Performance Prediction for the Examination Timetabling Problem

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computational Intelligence

by

DI Olga Zhukova
Registration Number 1228302

to the Faculty of Informatics

at the TU Wien

Advisor: Priv.-Doz. Dr. Nysret Musliu

Vienna, 1st March, 2018

Olga Zhukova	Nysret Musliu

Erklärung zur Verfassung der Arbeit

DI Olga Zhukova
736 8th AVE NE Unit 509, 98029, Issaquah, USA

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. März 2018

Olga Zhukova

Acknowledgements

First and foremost, I am deeply grateful to my supervisor Priv.-Doz. Dr. Nysret Musliu. Completing this thesis would not have been possible without his patience, immense knowledge, invaluable support and great assistance.

Furthermore, I would like to express my gratitude to the researches for providing the material for my experiments: the instance sets, the implementations of the heuristics for the ETP and the instance random generator. Without their overwhelming support doing this thesis would not have been attainable.

Also, I would like to mention the Erasmus Mundus MULTIC Consortium which provided me the financial support for my masters program. In addition, my gratitude goes to the program coordinator, Prof. Nafisa Yussupova, for her continuous support and essential help during my study in Vienna.

Besides, I would like to thank John Yu for his kindness and instant help with proofreading.

Finally, I would like to express my gratitude to my family and friends, for their help, motivation and support throughout my life. Special thanks goes to my husband, Alex, who aided and stood by me through many hard times and without his help finishing this thesis would not have been achievable.

Kurzfassung

Das Examination Timetabling Problem (ETP) ist eine wichtige Aufgabe, die regelmäßig an Universitäten, Hochschulen und Schulen auftritt. Eine allgemeine Definition des ETP, welche die meisten Fälle abdeckt, lautet: ein Timetabing Problem ist ein Problem mit vier Parametern: T, eine endliche Menge von Zeiten; R, eine endliche Menge von Ressourcen; M, eine endliche Menge von Ereignissen; und C, eine endliche Menge von Bedingungen. Das Problem besteht darin, den Ereignissen Zeiten und Ressourcen zuzuweisen, um die aufgestellten Bedingungen so gut wie möglich zu erfüllen. Ein besonderer Fokus ist jedoch aufgrund ihrer praktischen Ausrichtung auf die Formulierung auf dem ITC 2007 gerichtet.

Die Prüfungspläne wurden seit Jahren untersucht und die NP-Vollständigkeit einiger Formulierungen wurde bewiesen. Daher ist die Verwendung von exakten Methoden zeitraubend. Obwohl viele verschiedene Heuristiken entwickelt wurden, gibt es keinen bekannten Algorithmus, der in allen Problemfällen optimal ist. Nach dem No Free Lunch Theorem ist diese Situation sogar unmöglich. Um eine bessere Performance zu erzielen, können wir daher die Heuristik mit den besten Ergebnissen für eine konkrete Instanz auswählen. Diese Aufgabe wird allgemein als das Algorithm Selection Problem (AS) bezeichnet, bei dem die Frage ist, welcher der gegebenen Algorithmen die besten Ergebnisse für eine gegebene Instanz unter Verwendung einer bestimmter Instanzattribute erhält. Zusätzlich kann es praktisch nützlich sein, die Qualität der Lösung vorherzusagen, die von einer bestimmten Heuristik erhalten wird. Dieses Problem wurde als das Algorithm Performance Prediction Problem (APP) bezeichnet.

In dieser Arbeit präsentieren wir die Lösung für das Algorithm Selection Problem und das Algorithm Performance Prediction Problem für die ITC2007 Formulierung das ETP mit Techniken des Maschinellen Lernens. Dazu präsentieren wir ein Set von 196 relevanten Attributen und evaluieren außerdem die 3 modernen Heuristiken für das ETP auf dem Datensatz von 2243 realen und künstlich erzeugten Instanzen. Die Ergebnisse dieser Experimente ist, dass kein Algorithmus die anderen für alle Instanzen übertrifft. Anschließend verwenden wir 6 Klassifikationsalgorithmen und 9 Regressionstechniken zur Lösung die AS und die APP Problems. Darüber hinaus untersuchen wir den Einfluss der Parametereinstellungen und Attributtransformationstechniken auf die Performance des Schaetzfunktion. Darüber hinaus untersuchen wir die Bedeutung bestimmter Attributgruppen für die AS- und APP-Probleme.

Als Ergebnis ist es uns gelungen, ziemlich genau die Performancevorhersagemodelle für das APP und die Konstruktion des AS-Solver zu erstellen, die alle ihre zugrundeliegenden Heuristiken einzeln übertreffen.

Abstract

The Examination Timetabling Problem (ETP)is an important task that appears periodically at universities, colleges and schools. A general definition of the timetabling problem which covers most cases is as follows: a timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of exams; and C, a finite set of constraints. The problem is to assign times and resources to the exams so as to satisfy the constraints as much as possible. However, due to its practical focus, a particular point of interest is the formulation presented on the ITC 2007.

Examination timetabling has been studied for years, and NP-completeness of some formulations was proven. Therefore exact methods can not always solve large instances in a reasonable time. Although a lot of different heuristics have been developed, there is no known algorithm that dominates on all problem instances. Moreover, according to the No Free Lunch Theorem, this situation is in fact impossible. Therefore, in order to achieve better performance, we can choose the best performing heuristic for a particular instance using a set of predefined instance characteristics. This task is commonly known as the Algorithm Selection Problem (AS). Additionally, it might be practically useful to be able to predict the quality of the solution obtained by a certain solver on a given instance. This problem has been named the Algorithm Performance Prediction Problem (APP).

In this thesis, we present the solution for the Algorithm Selection and the Algorithm Performance Prediction Problems for the ITC2007 formulation of the ETP using Machine Learning techniques. For that, we introduce a set of characteristics which consists of 196 features. Also, we collect 3 State-Of-The-Art heuristics for the ETP and run them on the dataset of 2243 real-world and artificially generated instances. We find none of the algorithms outperforms the others for all instances. Subsequently, we use 6 classification algorithms and 9 regression techniques for solving the AS and the APP problems respectively. Additionally, we investigate the influence of the parameter settings and preprocessing techniques on estimator performance. Moreover, we inspect the importance of particular feature groups for the AS and APP problems.

As a result, we succeed in the building of rather accurate performance prediction models for the APP and construction of the AS solver that outperforms all of their underlying heuristics individually.

Contents

CHAPTER 1

Introduction

The Examination Timetabling Problem (ETP) is an important task that appears at least twice in a year in different educational institutions. The quality of the constructed timetables is especially important as it influences numerous stakeholders including lectures and students. However, as real-word instances consist of thousands of students and hundreds of exams, it is challenging to construct manually a good quality timetable in a reasonable amount of time. Moreover, some of the formulations of the ETP have been proven to be NP-compete [CK96]. As a result, the ETP obtained high research attention since the 1960s, and it significantly increased in the last decades.

Burke et al [BKdW04] defined a general timetabling problem as follows: timetabling is a problem that consists of the following four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of events (exams); and C, a finite set of constraints. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible. However, there are many formulations of the ETP as scheduling requirements vary from one university to another.

Although there were many algorithms developed for different problem formulations, due to limitations and variations of the formulations the methods were barely helpful in practice. Thus, in the International Timetabling Competition 2007 the ETP formulation has been extended and unified with the potential to solve practical instances. Moreover, a new benchmark dataset of real-world instances has been presented.

However, due to NP-completeness and problem size, the usage of exact methods that explore the complete search space and guarantee the optimality of the solution is infeasible. Therefore, most approaches are heuristics that can provide a fairly good solution in a reasonable amount of time. Nevertheless, there is no known algorithm that outperforms the others in all problem instances. Moreover, conforming to the No Free Lunch Theorem [WM97] domination of one technique is even impossible. Due to the complexity of the Examination Timetabling Problem choosing the best solver for a particular instance manually can be hard and therefore we face the question: can we find a way to predict which of the given algorithms will obtain the best solution on a given instance. This task is commonly known as the Algorithm Selection Problem.

The Algorithm Selection Problem was introduced by Rice [Ric76] and he presented a formal abstract model with 4 main components, namely: the problem space $\mathcal{P}$ with instances of the problem, the feature space $\mathcal{F}$ with measurable characteristics of the instances, the set of available algorithms $\mathcal{A}$ and the performance space $\mathcal{Y}$. In present time Machine Learning techniques are widely used to solve the Algorithm Selection Problem where the features combined with performance metrics across the instances solved by different solvers. One of the most important questions in Algorithm Selection is still finding a set of relevant features which would be suitable for a given problem.

Additionally, as algorithm performance vary across the instance space, it might be practically useful to get an idea about the quality of the solution without actually constructing a timetable. This scenario arises when, for example, we need to optimize the usage of certain resources, therefore, we need to test several settings, and choose the best one. This problem is considered as the Algorithm Performance Prediction (APP) problem that has been formally described by Leyton-Brown [LBNS06]. Interestingly, we can also solve the APP problem by application of ML techniques.

The goal of this thesis is application of various Machine Learning techniques for solving the Algorithm Selection and Performance Prediction problems for the Examination Timetabling Problem on per instance basis using formulation from the ITC 2007 competition. For this purpose, we expect to find new features based on problem description of the ETP and by reducing the ETP to other well-known problems. Besides, using feature selection techniques, we try to determine which features provide the best characteristics of the instance in general and characterize the empirical hardness for a certain solver.

On the one hand, we will use various classification algorithms for tackling the Algorithm Selection Problem directly. Additionally, we will predict the quality of the solution obtained by a certain solver after a predefined amount of time by constructing performance prediction models using regression techniques. Finally, the performance of the Algorithm Selection approach will be compared with the results of State-Of-The-Art algorithms for the ETP. Additionally, the influence of different parameter settings and preprocessing techniques for the performance of different estimators will be evaluated.

1.1 Objectives

The objectives of this thesis are the following:

- Investigation of State-Of-The-Art techniques developed for the ETP formulation from the ITC 2007 competition and comparison of their performance on representative instance set;

- Examination of already known features for the ETP, construction of a new set of features and identifying the most important attributes that characterize the problem instance the best;
- Development of performance prediction models in order to predict the solver score obtained on given instance;
- Exploring Machine Learning techniques for solving the Algorithm Selection Problem for the ETP and analyzing the impact of different preprocessing techniques and parameter configurations for the estimator performance;
- Comparing the results of application of Machine Learning approaches for the Algorithm Selection with the results of State-Of-The-Art heuristics for the ETP.

1.2 Main Results

The main contributions of this thesis are:

- We investigated performance of 3 State-Of-The-Art techniques on set of 2248 instances that contains real-world and artificially created instances. We observed that none of the methods outperform all other algorithm in all instances;
- We constructed a feature set containing 196 attributes for characterizing the problem instance of the ETP based on the problem description and the reduction to graph-related problems. Additionally, we identified important features for the classification and the regression tasks using various variable selection methods;
- We built up a set of accurate performance prediction models using new constructed features where 9 different regression approaches have been tested. These models allow us to predict the solution quality obtained by a certain solver after predetermined amount of time;
- We applied 6 classification algorithms with the feature set to tackle the Algorithm selection problem directly;
- We performed proper parameter tuning and inspected the impact of various preprocessing methods into prediction results;
- Finally, we compared the developed Algorithm Selection approach with the performance of underlying heuristics. As a result, our approach was significantly better than any of the investigated algorithms individually.

1.3 Structure of the text

This thesis consists of the following chapters: **Chapter 2** focuses on the Examination Timetabling Problem, its formulation and various approaches developed for tackling the ETP. In **Chapter 3** we present the Rice framework for the Algorithm Selection problem and define the concept of Empirical Hardness Models for performance prediction. Additionally, we observe some components of the Rice framework in detail. Also, we provide a short overview on related work on algorithm selection and performance prediction. In **Chapter 4** we describe the experimental settings for the Algorithm Selection and Performance Prediction Problems for the ETP using components of the Rice framework. We specify the data and the methods used for identifying the instance, the algorithm, the feature and the performance space for the ETP. In **Chapter 5**, we present the experimental results and evaluate the impact of various preprocessing techniques and parameter settings on estimator performance. In **Chapter 6** we summarize the results and discuss possible future work.

CHAPTER 2

The Examination Timetabling Problem

2.1 Problem statement

The Examination Timetabling Problem (ETP) is one of the most challenging tasks that regularly appears at universities, schools and colleges and has been studied since the 1960s. Below a general definition of timetabling is provided as in [BKdW04] and is appropriate for most cases.

> A timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; C, a finite set of constraints.

The problem lies in assigning times and resources to the meetings so as to satisfy the constraints as much as possible. Based on institution requirements, the timetabling problem can be formulated either as a *decision problem*, where the question of the possibility of finding any schedule that satisfies all the constraints is faced, or as an *optimization problem*, where the solution does not violate any hard constraints and satisfies as many of the soft constraints as possible based on the given objective function. The NP-completeness of the decision problem was first proven in 1996 in five separate ways where the constructed instances were being actually appear in practice [CK96].

Educational timetabling problems can be also classified depending on their purposes and the institutions involved in school timetabling, course timetabling and examination timetabling. Every class of scheduling is usually discussed separately, and it is because of this that this thesis will concentrate on the Examination Timetabling Problem. In addition, there are different formulations of the ETP as scheduling requirements vary from one university to another; therefore, some benchmark datasets together with different problem formulations were established. Nevertheless, in most cases the ETP is represented as optimization problem.

One of the most well-known benchmark data is the Toronto dataset, and over the years, researchers have used it to compare designed algorithms. This dataset, based on 13 real-world exam timetabling problems from Canadian high schools and universities, was introduced by Carter [CLL96] in 1996. Two different objectives were defined: either to find the shortest feasible timetable or minimize the average cost per student with the calculation depending on the constraints' violation. Over the years, additional constraints for the Toronto dataset were also introduced and tested, for example in [BNW96] [TMW99]. Other benchmark datasets that regularly appear in the literature are Nottingham Benchmark Data, introduced in [BNW96], and Melbourne Benchmark Data, published in [Bar96]. Additionally, due to high research attention paid to educational scheduling, two International Timetabling Competitions were organized, namely in 2002 [Net] and 2007 [Gro]. In ITC 2007, the problem model for the ETP was extended and the new, highly-constrained formulation was put forward in comparison with the previously provided models. In this thesis, this model is the primary focus due to its high potential to solve the practical instances of the problem.

The ITC 2007 problem model [MM08] is formulated as an optimization problem and consists of the following:

- A list of periods over a specified length of time where the periods' lengths are provided;
- A set of exams is to be assigned to specified periods with sets of students enrolled into each exam where each student can be enrolled into several exams;
- A set of rooms with individual capacities;
- A set of soft and hard constraints.

A timetable is considered to be *feasible* if it satisfies all hard constraints of the problem:

- All exams must have assigned periods and rooms;
- No student may attend two or more examinations at the same time;
- The number of students attending any exam must not exceed the capacity of the assigned room;
- Period- and room-related constraints are met;
- Period duration restrictions are satisfied.

The penalty for the constructed timetable is calculated based on the number of occurrences of the following soft constraints:

- **Two in a row** – the student has to attend two exams in a row on the same day;
- **Two exam a day** – the student has to attend two or more exams on the same day;
- **Period spread** – the student has to attend more than one exam within a specified period;
- **Mixed duration** – exams with different durations that are assigned to the same room;
- **Large exam allocation** - large exams are assigned later in the timetable;
- **Room- and period-related soft constraints** – use of specific time periods and rooms that has an associated penalty.

The quality of the timetable is measured by the hard constraints violation (distance to feasibility) and the weighted sum of the soft constraints violation. As the new model was constructed using a real-world perspective, soft constraint weightings for resource specific constraints are provided in the data itself within the "Institutional Model Index" instead of in the problem formulation.

2.2 Algorithms for the Examination Timetabling Problem

In this section, we would like to provide a short overview of the different approaches that have been developed since the 1960s. As previously mentioned, the decision variant of the ETP is NP-complete, and the optimization variant can be formulated as a decision problem as well. A determination must then be made as to whether or not it is possible to find a solution with a given value of the objective function. Therefore, the optimal solution can only be found for small instances of the ETP, but usually in practice each instance has to assign at least few hundred exams. Another possibility would be to reduce the ETP to other well-known problems that can be solved separately, e.g., the satisfiability problem (SAT) or the Graph Coloring Problem (GCP). Nevertheless, soft constraints are often contradictory, so satisfying all constraints would be impossible. This fact makes reduction difficult because in complete mapping, no solution will be found, or, alternatively, the question of which constraints to satisfy will be faced. However, researches often use algorithms developed for other problems. For example, graph coloring approaches are commonly employed as construction heuristics for the initial solution.

Because of the reasons described above, heuristics methods are necessary. While they do not guarantee that the optimal solution will be obtained, satisfactory results can be reached within a reasonable amount of time. Moreover, many successful approaches consist of a combination of different techniques. Thus the classification of different methods depends on the primary technique in use. For more information, the following surveys [QBM^{+}09, Sch99, CLL96] may be referenced.

2.2.1 Graph Based Techniques

Firstly, the connection between the timetabling problem and the GCP was established in 1967 by Welsh and Powel [WP67], so that was when research on the use of graph heuristics in timetabling problems began. Graph coloring heuristics had a great impact on early timetabling research [Car86], and today they are often utilized as construction heuristics within hybrid techniques due to their relatively short runtime and simple implementation. As previously mentioned, there are some potential problems with reduction that can occur when searching for an optimal solution. However, it is possible to map the ETP to the GCP to some extent. For the given ETP, a graph is constructed in such a way that each exam is represented by a vertex, and an edge exists between two vertices if two exams cannot be assigned in the same period or one of the hard constraints will be violated. The typical question for the GCP is a graph can be colored using p different colors so that none of the adjacent vertices have the same color corresponds to the question of assigning exams into available timeslots. Nevertheless, soft constraints still need to be taken into account separately.

Graph heuristics mostly use a special order according to how difficult it is to place the exam into periods without violating any of the constraints. Moreover, some specific strategies for choosing the timeslot can be used as well. A wide variety of such strategies and their modifications have appeared in the literature [QBM^{+}09, Car86]. In the section below, some popular techniques from this family of algorithms will be be discussed.

The largest degree first (LDF) [WP67] heuristic is based on the idea that exams that have the most conflicts with other exams are harder to schedule. Hence, all courses are ordered decreasingly by the node degree of every vertex, and the hardest instances are placed foremost into the first suitable timeslot. In the technique **largest degree first: fill from the top (LDFT)** [PW66], exams have the same order as in the previously described method, but we assign as many exams as possible to the first time period, then move forward to the next timeslot and continue until all the vertices are assigned. **The Largest degree first recursive: fill from the top (LDFRT)** method was proposed in 1978 [Car78] and is similar to the latter algorithm. The difference is that after every assignment the exam is removed after coloring and then the new nodes' degrees are recalculated in the residual graph before the next iteration. **The saturation degree heuristic (SATUR)** [Bre79] was developed based on assumption that exams that have the largest amount of available timeslots are easier to assign, so it is used increasing order, depending on how many available timeslots an exam has. The coloring is performed similarly to the LDF heuristic. **The Largest Enrollment (LE)** strategy was first employed in [Zel74] and provides an order depending on how many students registered for the exam. **The color degree (CD)** algorithm [CLL96] uses the decreasing order of number of conflicts that an exam has with an already scheduled exam. Moreover, the method of using random ordering (RO) of the vertices was employed in [CLL96] to compare different techniques and is often used to introduce randomness into the combined approaches.

Additionally, advanced methods based on combining different techniques were widely developed. Burke et al [BNW99] introduced randomness in a hybridized algorithm utilizing LDF, SATUR and CD which uses the following two strategies for selection: tournament selection that randomly takes one of the top exams from one of the lists or bias selection. The adaptive ordering strategy [BN04] was implemented in 2004 as part of the iterative method which uses dynamic ordering instead of a static one. Depending on how difficult the exam assignment was during the previous iterations, a new order is created. Also, different reassigning techniques, such as backtracking (e.g. [CLL96]) and look ahead [BN99], in addition to graph coloring methods were widely studied and implemented in the literature.

2.2.2 Local-search Based Techniques

Local search techniques have been successfully employed to solve a lot of different and difficult computational problems, including timetabling problems. The main idea of these algorithms is to search in the neighborhood from the initial solution to what is deemed the optimal solution where the solution quality is defined by the given objective function. These methods have been widely used for the ETP as the constraints can be easily handled. In the following subchapter, we will describe the most popular methods while focusing on techniques developed for the ITC 2007 formulation.

Simulated Annealing

Firstly, Simulated Annealing (SA) was introduced by Kirkpatrick et al [KGV83] in 1983 and then widely investigated by Laarhoven et al [vLA87]. Simulated Annealing is a probabilistic local search algorithm that was inspired by natural annealing and the crystallization processes and proceeds with the goal to reach the global optimum of an objective function. At the beginning, the algorithm accepts worse moves with higher probability that helps to extend the search space. At the end of the procedure, the probability gradually decreases, so the algorithm converges to the optimum. This method has a lot of parameters that can be tuned, and the result is directly dependent on the parameter configuration.

One of the first successful applications of Simulated Annealing was presented in 1998 by Thompson and Downsland [TD98] when they developed the two-stage approach. The first phase was used to search for a feasible solution, and the quality of the timetable was improved during the second phase based on soft constrains satisfaction. Another hybrid technique using SA was developed in 2003 by Merlot et al [MBHP03] where the initial solution was constructed by Constraint Programming Techniques. SA was then used together with the modified Kempe chain neighborhood to find the optimal solution. Finally, the Hill Climbing procedure has also been employed for further solution improvement. Notably, this solver has achieved the best results for some of the benchmark instances. Another important outgoing work was published by Battistutta et al [BSU14] in 2014 where the single-stage SA technique was proposed together with feature-based parameter tuning. Optimal parameters for the algorithm were found using the regression model. Furthermore, this solver has been able to achieve the best scores for a few instances of ITC 2007 formulation. It has been shown that with good parameter settings, the solver based on SA can outperform some specific state-of-the art algorithms for certain formulation.

In addition, some modifications of the SA procedure are widely used in current research, such as the Great Deluge algorithm (GD) that was introduced by Duek [Due90] as an alternative to SA. In this procedure, bad moves are accepted regarding current solution just in case if the penalty of candidate solution is below predefined level. Such level is decreasing with the time and so the algorithm will converge at some point.

Burke et al in 2004 [BBNP04] applied GD algorithms for the Toronto and Nottingham datasets, and the authors were able to obtain good results. In addition, the initial solution in this solver was constructed using the Saturation Degree heuristic that runs several times. Mueller from Purdue University, the winner of Examination Timetabling track in the ITC 2007 [Mue09], proposed a three-phased approach where the last phase is based on a GD algorithm. In this solver, the feasible solution is constructed using the Iterative Forward Search, and in the next stage, the Hill Climbing method is employed to find the local optima. Lastly, as mentioned before, the Great-Deluge approach has been applied for solution improvement. This solver is still producing the best scores for some benchmark instances of the ITC 2007 formulation. Another successful approach for the ITC 2007 formulation was developed by McCollum et al [MMP$^+$09] where the extended variant of the Great Deluge Algorithm was employed. Construction of the initial solution was performed utilizing the adaptive ordering heuristic [BN04] where it is used a few times to achieve better solution quality.

Tabu search

Tabu search (TS) is a local-search technique that was invented by Glover [LG97, dG02]. It tries to move towards the best available solution in the neighborhood. The hallmark of this algorithm is its adaptive memory, or the list of the solutions that is forbidden for some iterations. However, the tabu status of the solution can be overruled if the aforesaid moves are better than the best existing solution (aspiration criteria). The number of iterations has usually been used as a stopping criteria. Moreover, different types or combinations of memory structures can be utilized.

One of the first practical applications of a simple Tabu Search to the ETP was performed in 1991 by Herz [Her91] to address a problem at the University in Geneva and Swiss Federal Institute of Technology in Zurich. Since then, it has been widely used to solve real timetabling problems, e.g., in [BN96].

In 2001, valuable work for the ETP using Tabu Search methods was carried out and analyzed by Di Gaspero and Schaerf in [dGS01] for the Toronto and Nottingham benchmarks. They implemented two TS-based procedures together with violations list structures. The course is considered to be in a violation list if it violates either hard or soft constraints. A comparison of a simple TS and tandem strategy has been studied together with biased and exhaustive selection methods. One year later, in 2002, the method was improved by Di Gaspero [dG02] by developing multiple neighborhood structures to escape the search from local optima. For this purpose, the token-ring search was used together with the following neighborhoods which were implemented: recolor (change a single exam), shake (swap groups of exams) and kickers (change the sequence of single exams).

In 2002, a four-stage Tabu search algorithm called OTTABU was developed by White and Xie [WX01]. This approach uses frequency-based short-term memory and frequency-based long-term memory for solution improvement. It was found that the length for long-term memory strongly affects the quality of the solution, and as a result, the quantitative analysis for appropriate length estimation was employed. The OTTABU technique was examined on real data collected from University of Ottawa. In 2004 [WXZ04], the authors enhanced the system by relaxing the tabu lists and adding several other features. When the algorithm is compared to 5 other techniques on the data set from Carter, it compares favorably.

The TS approach has been incorporated for solving the ETP within the Open Source Business Rule Management System called Drools [GAH08]. Firstly, the constraints are formulated as rules in Drools Rule Language, and the exams are assigned based on their size and duration of ordering strategy. Afterwards, TS is implemented with the following neighborhoods: exam, period and room moves and swaps. This solver participated in ITC 2007 as well and received third place.

Other Local Search Based Techniques

Besides the algorithms described above, a large number of methods based on the Local-Search concept have been developed and are used widely for solving the ETP. In this subchapter, some of these methods that have shown good results on benchmark instances are described.

Recently, the idea of having multiple neighborhoods structures to escape from local optima has become popular and has been heavily studied within the last decade. One of the techniques from Local-Search that has been employed for solving the ETP is a Large Neighborhood Search (LNS) that was first proposed by Shaw [Sha98]. In 2007, Abdullah et al [AABD07] utilized LNS for the Toronto variant of the ETP together with the cyclic exchange neighborhood where the enumerating strategy is based on an improvement graph. Furthermore, this method was late modified [AAB$^+$07] so that the tabu list is used to store the improvement moves. Another method based on a similar idea is the Variable Neighborhood Search (VNS) where the neighborhoods are iteratively changed during the search procedure. One example of the use of the VNS for the ETP was employed by Burke et al [BEM$^+$06]. In addition, the genetic algorithm was used for neighborhood selection.

In 2007, Gogos [dS08] implemented the GRASP (Greedy Randomized Adaptive Search Procedure), which won second place in the competition. First, he made 5 lists of exams using different ordering strategies. Then a tournament selection was applied for exam placement together with the backtracking technique and the obtained solution was improved using the Simulating Annealing procedure. Lastly, the Integer Programming technique was exploited with the Branch and Bound strategy to switch the rooms. In 2012, this approach was improved [GAH12], and the Hill Climbing with the Kempe chain neighborhood was implemented among the improvement phase in addition to other modifications. The scheduling difficulty of the exam was defined by the incorporation of various criteria such as student enrollment, conflicts and other criteria. Another alteration of Integer Programming formulation is used for the period swap. Furthermore, "shaking" the Kempe Chain moves for the rooms is utilized before passing the solution back to the SA stage. Finally, another heuristic is trying to generate new and promising possibilities for improvement by removing some exams stochastically and then rebuilding the initial solution for SA using the initial construction procedure.

2.2.3 Population-Based Approaches

Population-based approaches have become popular for combinatorial problems over the last decade, so a large amount of algorithms and their modifications have been developed. In this subchapter, some well-known techniques that have achieved good results on benchmark instances and their applications for real-world problems will be discussed.

Evolutionary Algorithms

Evolutionary algorithms are a huge family of algorithms that was inspired by evolution in nature and mimics the biological processes. Usually these algorithms consider a set of candidate solutions instead of a single assignment in comparison with Local Search methods. One of the most well-studied heuristics from population-based methods for timetabling research is the Genetic Algorithm (GA). At the beginning, the initial population is created and then, using mutation and crossover operators, the new generation is obtained and the best representatives are maintained to the next iterations. The quality of the candidate solution is defined in terms of fitness function.

Schebani et al [She02] applied the standard variant of the GA for the ETP and designed a mathematical model for problems in training centers with the fitness function concentrated on the maximum spread between meetings. At the same time, Wong et al [WCG02] presented work that created a timetable at the Ecole de Technologie Superieure that was formulated as a Constraint Satisfaction Problem. Additionally, tournament selection was employed for parent and mutation selection. In 2005, Cote at al [CWS05] studied a bi-objective evolutionary algorithm with TS and Variable Neighborhood Decent operators for recombination. Moreover, promising results have been obtained for some Toronto benchmark problems. In 2007, Ulker at al [OUOK07] proposed the use of the Linear Linkage Encoding representation for the GA. This approach has been employed and examined for both Graph Coloring and the ETPs.

For the ITC 2007 formulation, Pillay et al [PB08] investigated a technique inspired by cell biology . The algorithm mimics cell processes of division, interaction and migration. In 2010, Pillay and Banzhaf [PB10] developed a two-stage approach for the ETP incorporating the GA where the feasible solution is obtained in the first stage, and later an improvement that considers the soft constraints is made. Authors testing this technique on Carter's benchmark instances got results that were competitive with the best.

Other Population-based Algorithms

Memetic algorithms [KS05] are modifications of a simple GA. The main idea of this approach is that current population can be enhanced by employing Local-Search methods. Recently, this method has been widely employed in research, and some promising results have already been achieved.

Firstly, the Memetic approach for solving the ETP was developed by Burke et al [BNW96] in 1996. Additionally, different mutation operators for both a single as well as a set of exams were implemented and tested. For the improvement phase, the Hill Climbing technique was utilized. In 2014, the variant of the Memetic algorithm called Harmony Search was studied by Azmi Al-Betar et al [ABKD14]. Moreover, the authors investigated the influence of different settings on the solution quality, namely randomness, neighborhood structures and recombination.

One well-known technique from the population-based methods is the Ant algorithm [DT05]. This approach mimics ants' behavior of searching for the shortest path to food source by leaving pheromones. Moreover, different algorithm modifications such as the Ant Colony System, the Min-Max and others have been developed. However, this algorithm has not been widely studied in the area of examination timetabling.

Naji Azimi [NA04] developed the framework, including the Ant Colony System (ACS) algorithm, where the solution found by the ACS was later improved using simple Hill Climbing. Afterwards, the results obtained were analyzed and compared with other implemented techniques, namely the TS, SA and GA. It was shown that the ACS was able to deliver the best solution compared to the other approaches employed most of the time on the dataset proposed by Carter. In 2005, the author investigated hybrid combinations of the TS and the ACS in three different variants [NA05]. The best results were achieved by the sequential ACS followed by the TS, and it should be noted that all of the proposed hybrid approaches performed significantly better than any of the single methods. Downsland and Thompson [DT05] employed the Ant technique and carried out additional research on the influence of various algorithm configurations to reach the final solution, especially concentrating on different initialization techniques. The authors consider the Carter problem formulation without soft constraints together with three variants of the fitness function. Furthermore, the results obtained were competitive with others provided for this dataset.

2.2.4 Hyper-Heuristics

Most of the algorithms that display good results on benchmark instances highly depend on the parameters turning. Moreover, they are specifically designed for one particular problem and thus, do not perform well on other problems. Often, parameter turning is considered to be hard, which has led many researchers to focus their attention on developing more general approaches for solving combinatorial problems. The definition of Hyper-Heuristics provided by [BGH+13] is *"a search method or learning mechanism for selecting or generating heuristics to solve computational search problems".* In the following chapter, some of applications of hyper-heuristics will be discussed; however, for more detailed information the following surveys may be referenced [QBM+09, BGH+13].

One of the first times it was suggested that Hyper-Heuristics could be used to solve the ETP was in research done by Ross et al [RHC98]. The authors suggested the use of the Genetic Algorithm to search for an appropriate method instead of a specific solution because, as they pointed out, the GA with direct encoding does not perform well on some problem classes. Later, Ross et al [RMBH04] applied the GA to search in a simplified problem-state description space, and, for this purpose, associations between the problem states and heuristics have been established. The heuristics finding was used for solution construction. In this work, the authors deployed and tested combinations of 16 algorithms for exam ordering and 28 techniques for choosing the timeslot. Moreover, this approach has been shown to deliver promising results across both the exam and course timetabling formulations with three different fitness functions.

Pillay and Banzhaf [PB07a] studied the Genetic Programming approach for incorporating different graph-based heuristics into a Hyper-Heuristic system to define the order in which exams should be assigned. Pillay [Pil04] later extended the approach by three various representations for heuristic combination. The method was examined on Toronto benchmark instances, and a comparison of different representations was discussed. As a result, feasible solutions were obtained in all cases. In 2010, the same concept [Pil10] was applied for the ITC 2007 competition benchmark. In contrast, the combination of chromosome representations had been employed. For all chromosome combinations, a feasible timetable was produced, and one combination especially showed the solution quality that was comparable with the best results that had been obtained at the time. In 2009, Pillay [Pil12] applied Genetic Programming for heuristic generation where the algorithm searches for a new heuristic constructed from low-level methods combined by logical operators. This method calculates the difficulty of exam allocation for the construction phase.

Ahmadi et al [Lin02] investigated the perturbation-based VNS for parameterized construction heuristics. For defining heuristic space, different graph heuristics together with weighted objective function were employed. For testing, the authors used real-world data obtained from Nottingham University. The approach showed promising results in finding an appropriate combination of heuristics to solve that particular instances.

Burke et al [BQS14] provided a Hyper-Heuristic approach that combines low-level heuristic moves to improve the timetable. In the experiments, it was observed that the Kempe Chain Move heuristic and Time-Slot Swapping heuristic are the best suited for heuristic moves and that the most appropriate techniques for defining the ordering of exams are SD and breaking ties with the Largest Weighted Degree. Consequently, approaches described above have been incorporated into an iterative two-phase approach. During the first stage, graph-based algorithms are employed for feasible solution construction. In the next stage, heuristics' sequences have been established and are used to improve the solution by reassigning the exams that violate the soft constraints. This method has been examined for both the ITC 2007 formulation and the Carter benchmark. Furthermore, the algorithm was able to achieve competitive results compared to the other Hyper-Heuristic approaches.

Burke et al [BPQ06] describe a Hyper-Heuristic system based on case reasoning for the ETP in which simple graph-based algorithms have been employed together with the Hill Climbing technique. The methods that obtained good results in specific instances are memorized, and heuristic selection is later performed based on the similarity measure defined by problem features. During the case-base refinement process that follows, some cases can be discarded if they prove to be useless in solving new instances. The authors aimed to construct a more general timetabling system that performs well on a variety of problem formulations instead of comparing and tailoring one to a specific problem format.

2.2.5 Other Approaches

As previously mentioned, due to the lack of space it is impossible to describe all of the strategies used to solve the ETP. However, in this subchapter a brief overview of other approaches that were not yet considered are provided.

Due to their simplicity, constrained-based techniques are often used in different research areas; therefore, exam timetabling is no exception. In such cases, the problem can be modeled in terms of variables with a finite domain. The task is to assign values to these variables so as to satisfy all of the constraints. These methods are usually computationally expensive and, for this reason, typically incorporated into other algorithms.

There are also approaches based on the decomposition of complex problem into sub-problems where every single problem can be solved separately. Researchers do not often use this technique for the ETP as there are some significant drawbacks. Firstly, sometimes there is the possibility that the combined solution will produce an unfeasible timetable. Another complication is the evaluation of some of the soft constraints while the problem decomposed. However, some research has made an attempt to put this technique into practice.

Based on the concept described above, Lin [ABC$^+$03] created a multi-agent system where the ETP is decomposed for particular problems. Every problem is solved separately by agents, and the remaining schedules are solved by a broker. In the end the final solution is obtained by combining all of the constructed timetables. This approach has been investigated on both Toronto and randomly generated data, and it was observed that its performance was adequate on some types of the problem. Another example of decomposition techniques was proposed by Qu et al [QB07] in which the authors classified the exams into two sets arranging them by the difficulty of the assignment on previous iterations. The algorithm adaptively adjusts the ordering of the exams using the sets obtained. For this purpose, both exams that led to infeasibility and exams that had the highest impact on the solution score move forward in the ordering list. This approach performed well compared to other State-Of-The-Art techniques.

Another promising approach is the Multi-Criteria technique. Traditionally, algorithms consider constraints violations in terms of one objective function that is represented by a weighted sum. However, with regard to real-word problems, it is necessary to take every class of constraints into account separately, and Multi-Criteria techniques solve this problem efficiently by using the vector of the constraint violation instead of a single value. For a more detailed look at these algorithms refer to [PB07b].

CHAPTER 3

The Algorithm Selection And Algorithm Performance Prediction Problems

In the following chapter, the Algorithm Selection problem will be introduced. Moreover, different aspects of the Rice framework will be split up and examined in detail. Also, the concept of Empirical Hardness will be presented, and the Empirical Hardness Models (EHMs) employed for the Algorithm Performance Prediction Problem will be formally described. The methodology used for the construction of the EHM will be presented as well as some important aspects highlighted. Finally, work closely related to the Algorithm Selection and Performance Prediction problems will be also observed.

3.1 The Algorithm Selection Problem

For many years, researchers mainly developed or improved the algorithms for one of the combinatorial problems. However, even if in some cases one algorithm is significantly better than other approaches, there is no known algorithm that outperforms all of the others in all problem instances. Moreover, according to the No Free Lunch Theorem [WM97] for one technique to dominate is impossible. Instead of inventing new algorithms, some researchers started to work on the task of choosing an existing method that would provide the best solution to any given circumstance. This undertaking is commonly known as the algorithm selection problem and was first formalized by Rice in 1976 [Ric76]. In his paper, Rice presented a formal abstract model with 4 main components, namely:

- the problem space $\mathcal{P}$

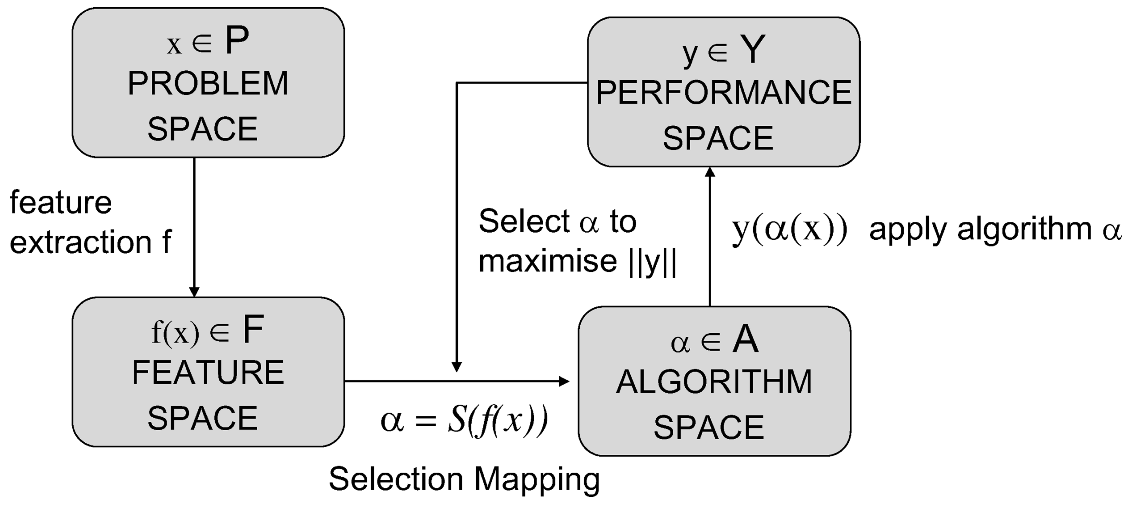

Figure 3.1: Schematic diagram of the Rice framework [SM08].

- the feature space $\mathcal{F}$
- the algorithm space $\mathcal{A}$
- performance space $\mathcal{Y}$

In the model the problem space $\mathcal{P}$ consists of the set of problem instances, and the feature space $\mathcal{F}$ represents the diversity of measurable instance characteristics that are suitable for the given problem. Moreover, the algorithm space $\mathcal{A}$ is characterized by the set of various existing algorithms, and performance space $\mathcal{Y}$ is identified by performance criteria applied to a given algorithm portfolio. Schematic diagram of Rice model can be seen on Figure B.5 and the formal definition of the Algorithm Selection Problem consists of the following [SM08]:

> For a given problem, instance x $\in \mathcal{P}$ with features $f_x \in F$ finds the selection mapping $S(f_x(x))$ into algorithm space $\mathcal{A}$, so the selected algorithm $\alpha \in \mathcal{A}$ maximizes the performance mapping y(α(x)) $\mathcal{Y}$.

In practice the use of good performance mapping is usually enough when it delivers satisfactory results on new unseen instances and outperforms State-Of-The-Art approaches; however, to even find good mapping is difficult. Moreover, it was also proven that the Algorithm Selection (AS) problem is undecidable [Guo03].

Further reading about algorithm selection framework and its applications can be found in the following papers [Kot14, SM08, HXHLB14].

3.2 The Empirical Hardness Models

It is widely known that the performance of most of the algorithms for combinatorial problems varies significantly across the instance space - even on instances of similar size and structure. Because of this, the research has begun to focus on understanding the complexity of an instance for a particular solver in terms of instance characteristics.

Firstly, the **Empirical Hardness** and **APP** concepts have been formalized by Leyton-Brown et al [LBNS06] where they characterize instance hardness by the runtime of a certain solver obtained on the instance. Moreover, the methodology of hardness models construction has also been provided. Later this idea was formalized into the Empirical Hardness Models (EHMs) [LBHHX14] where each instance has been characterized in terms of problem features, and Machine Learning(ML) techniques are used for prediction. Therefore, various performance measures have been employed in order to understand hardness, such as solution quality obtained by an algorithm after a fixed amount of time [KS04], a method's probability of success [RH07] and others. However, it must be noticed that the complexity of an instance heavily depends on the particular solver used. In some cases, the instance may be considered "hard" for one algorithm, but another technique could obtain a good solution relatively easily, so this instance could also be considered as "easy".

As previously noted, the EHMs can be constructed using the methodology proposed in [LBNS06] that can be summarized into the following steps:

- Step 1: Select the problem instance distributions;
- Step 2: Choose the algorithm set;
- Step 3: Select a set of inexpensive, distribution-independent features;
- Step 4: Generate data;
- Step 5: Feature preprocessing;
- Step 6: Application of Machine Learning techniques.

However, the Rice framework described in section 3.1 for the Algorithm Selection Problem is sufficient enough to describe the APP problem. Therefore, some important aspects of the Rice framework for the AS and APP problems will be discussed in more detail.

3.3 Instance and Algorithm Space

For combinatorial problems, the instance set is usually given; however, it must be representative to cover the different aspects of the problem and must contain the instances of various complexity. The instance distribution defines the boundaries of the instances to be considered, and so it outlines the scope of the EHM and AS application where the model is expected to show good results. Typically, research is interested in practical applications; therefore, in most cases distribution is drawn uniformly from real-world data based on an observed parameter range. Nevertheless, sometimes it is practically necessary to utilize instance selection methods such as, for example, [SMI11, Jan00, WM00] for large datasets without the loss of essential information in a reduced subset.

Another essential question in Algorithm Selection process is defining the set of algorithms that will be used. Generally, portfolios divided into two main categories: static and dynamic [Kot14]. In dynamic portfolios the configuration of the algorithms or portfolio itself can be changed during runtime. Alternatively, static portfolios are predefined and so the choice of the algorithms in the set become crucial. For the best results the algorithm set should be diverse and methods should complement each other for achieving good performance on various problem classes. Moreover, different research has been performed regarding selection the algorithms for the portfolio and it has been observed that choosing the algorithms with overall best performance not necessarily led to the best results [XHLB12, HP04]. For example, Xu et al [XHLB12] analyzed the portfolio selection for SAT using 2011 SAT competition track and noticed that often solvers using novel strategies in comparison with others contribute the most.

However, due to the complexity of the methods, explicit investigation of different approach is hard and hence, often the solvers that either provided good scores in solver competitions or obtain best results in comparison with other solvers on benchmark instances are used.

In contrast with the Algorithm Selection Problem for performance prediction, it is sufficient to select some available algorithms and suitable performance criterion.

3.4 Feature Space

3.4.1 Feature Engineering

One of the key issues of the APP and AS problems is finding a good feature set in order to cover the hidden properties of the instance that correlates with instance hardness [Ric76]. Moreover, it has been observed that finding a comprehensive feature set is not a trivial task [Ric76, SML12] and it is usually necessary to have deep domain knowledge. At the same time, it has been noticed that problem characteristics are strongly related to the successful use of prediction mapping [Ric76, XHHLB08]. One of the requirements in the feature engineering process is that characteristics must be calculated relatively quickly (in low-order polynomial time) because a slow metareasoning procedure (including feature extraction) reduces the benefits of APP and AS and at a certain point can be avoided. This situation occurs if the time required by the metareasoning process is compared with the time that it would take to run all selected solvers.

An excellent survey regarding the features classified by type for a variety of combinatorial problems can be found in [HXHLB14, SML12]. Moreover, authors in [SML12] tried to answer two main questions that occur during feature design process: how to find a suitable feature set, and why selected features are appropriate for the characterization of a particular problem. They also tried to find a relation between algorithm performance on a particular problem in terms of problem characteristics. With regard to the first objective, some similarities in feature sets between different combinatorial problems has been observed and noted. However, the second question remains for the most part open, and authors refer to the algorithm footprint as one of the possibilities to answer it [SML11].

Notably, attributes can be classified into two different categories: problem specific and domain independent features regardless of problem type. Problem independent features usually consider some approaches that can provide some insights into the search space of algorithms. One such method is the Fitness Landscape Analysis (FLA) [Ree99, ME14]. The fitness landscape structure is defined in terms of a search space that consists of a set of solutions. Objective function represents the quality of the solution by assigning it some value and a neighborhood that defines a distance metric over the solution space. Unfortunately, it is impossible to construct a landscape structure until the solution set is known; therefore, it might be most appropriate to regard a particular algorithm's performance on specific problem classes. Additionally, different metrics for FLA have been developed and widely used such as ruggedness, fitness distance correlation, distribution of local minima and others. Moreover, these features have been successfully employed for the characterization of difficulties for such problems as the Travelling Salesman Problem (TSP) [SS92], the Knapsack Problem [MF00], the Quadratic Assignment [TPC08] and others.

Another problem-independent approach that can be used as an alternative to landscape analysis and is conventional in APP and AS is the method based on the idea of landmarking [PBGC00]. The intention is to run some simple algorithm for a short time such as Local Search or the greedy method and then use information retrieved from these runs as new characteristics. Such measures could include, for example, the number of gathered local optima, changes of solution quality during search run as well as others. Surprisingly, the landmarking concept is similar to the hyperheuristic approach described in subsection 2.2.4, for example in [BPQ06] where simple heuristics and information obtained are employed for choosing a more sophisticated algorithm to solve the ETP. However, even if the idea of landmarking is to form domain-independent characteristics, the availability of domain-specific solvers is essential.

Nevertheless, most of the features used for the APP and AS problems are domain specific. Usually, such a set would consist of the set of features related to the problem itself, and this hardly depends on problem formulation: size and constraint measures, statistical measures or matrix characteristics. Moreover, the features obtained for one problem class can be reused for another problem in the case of time-efficient reduction between problems [SML12]. Recently more researchers have begun to utilize this approach as it has demonstrated promising results.

3.4.2 Feature Preprocessing

Surprisingly, some instance attributes may be irrelevant for a particular problem and reduce the quality of the performance model even if they seem reasonable. Furthermore, certain measures may be highly correlate with one another, producing additional noise. As a solution to this problem, a variety of feature selection methods have been developed, and nowadays, the feature selection step is one of the essential parts of the ML applications. As further reading, the paper from [GE03] can be referred to where the authors highlight some important steps to solve the feature selection problem and provide a comprehensive overview of different variable elimination procedures. A more recent survey published in 2014 can be found in [CS14].

In general, feature selection is the process where the subset of attributes is chosen in order to improve prediction quality or to reduce the dimensionality of the problem. Variable elimination approaches are usually classified into 3 categories: filter, wrapper and embedded methods. These groups will be discussed briefly below.

Filter methods are universal and may be used regardless of type of the predictor. The main principle of this approach is to rank the variables based on certain relevance criteria like, for example, the Pearson Correlation Coefficient (PCC) [Bat94], the Mutial Information Coefficient (MIC) or Information Theoretic Ranking Criteria [LTM+12].

The PCC characterizes the existence and level of linear dependencies between variables. Despite significant drawbacks, such as sensitivity to outliers [Ans73] and relevance only for linear dependence recognition,the PCC is still used as a preprocessing step and data interpretation in combination with other methods. PCC values vary from -1 to 1, where -1 indicates a total negative correlation (inverse proportionality), +1 displays a total positive dependence (direct proportionality) and 0 means there is no linear dependence between variables. The PCC is applied for the two datasets X and Y with the corresponding mean values x_m and y_m calculated using the following formula:

$$r = \frac{\sum_{i=1}^{n}(x_i - x_m) * (y_i - y_m)}{\sqrt{\sum_{i=1}^{n}(x_i - x_m)^2} * \sqrt{\sum_{i=1}^{n}(y_i - y_m)^2}} \tag{3.1}$$

Anoter filter-based approach, the MIC is related to the concept of Mutual Information (MI) [SW48]. MI is a quantified measure of information that one variable can provide about another determined in bits. Moreover, MI for certain n variables equals 0 only if these variables are statistically independent. The formal definition of MI for the two random variables x and y with the defined marginal distributions $P(x)$ and $P(y)$ and joint distribution $P(x, y)$ is:

$$I(x, y) = \sum_{y \in Y} \sum_{x \in X} P(x, y) log(\frac{P(x, y)}{P(x)P(y)}) \tag{3.2}$$

However, due to significant shortcomings, the direct use of MI in feature ranking can be troublesome. Firstly, discretization by binning is needed for the calculation of continuous variables. Moreover, as MI is not normalized, it leads to incomparability between different datasets. These drawbacks were treated in the MIC where continuous variables are discretized by searching for optimal binning; the MIC values are also located in a fixed range. Furthermore, in comparison with the PCC, the MIC is able to determine non-linear dependencies.

However, for filter methods the measurements are usually defined individually for every variable without taking into account the context of the others and that has led to the presence of redundant variables. Nevertheless, due to simplicity and computational scalability, it is often utilized as a preprocessing step.

Wrapper methods [KJ97] try to choose the best subset of characteristics where the performance of a certain predictor is used as evaluation criteria. It has been shown that finding an optimal subset is an NP-hard problem [AK98]. Because of this, heuristics are used instead of an exhaustive search. Wrapper methods could be roughly classified into heuristic search and Sequential Feature Selection (SFS) algorithms [CS14]. SFS methods represent greedy approaches and use either forward selection strategy, where the algorithm starts with an empty set and progressively add new attributes, or backward elimination strategy, which starts with a full set and gradually eliminates characteristics while maximizing performance.

As an example, two representatives of SFS methods are the Recursive Feature Elimination (RFE) [GWBV02] and the Stability Selection (SS) [MB10] techniques that are widely implemented as a part of different libraries.

RFE is based on backwards selection where feature sets are recursively evaluated using the accuracy of a chosen predictor leading to features with the smallest weights being removed from the set. The procedure is repeated until the desired number of features has been reached. However, as the results of RFE are heavily dependent on the predictor chosen, it is important to select it very carefully.

Another relatively new feature selection technique is the Stability Selection (SS). The main idea of this method is to subsample the data with various feature subsets and then to measure the performance using the selected predictor. The process is repeated a number of times, and then, the resulting features' importance is then gathered depending on the feature relevance of subsampling results.

As a consequence, characteristics that are considered important and thus, repeatedly chosen across the subsampling process, will get a higher score compared to rarely selected features.

As heuristic approaches for feature selection, a wide range of algorithms could be employed such as Tabu Search [CY09], Genetic Algorithm [SK04], Simulated Annealing [MZ06] and others. However, a serious drawback of wrappers is significant computational cost.

The third group, **embedded methods**, is an approach in which the feature selection procedure is employed as a part of the training process. These algorithms commonly combine the benefits of the previously described methods as they are computationally inexpensive and tend to overfit the data less. As an example, the Lasso and the Ridge regression belong to the family of embedded methods and additionally, are considered to be regularization methods.

Lasso (or L_1 regularization) [Tib96] is an approach where the loss function $E(x, y)$ (usually the Root Mean Square Error (RMSE) described in section 4.4) of the predictor is regularized by an additional penalty. So in the case of L_1 regularization the estimator tries to minimize the expression where the penalty is represented by the sum of the regression coefficients' modules and α is a parameter for model tuning:

$$E(x, y) + \alpha \sum_{i=1}^{n} |w_i| \tag{3.3}$$

Accordingly, as the penalty value grows in the case of non-zero coefficients, the Lasso approach reduces the number of characteristics used in the prediction process. Moreover, we can adjust the importance of model sparseness by the parameter α where, in the case of increasing the parameter, the number of non-zero feature coefficients decreases. Consequently, an α equal to 0 will correspond to a non-regularized model. A good regularization parameter, however, instead of being manually chosen, can be selected by the algorithm. An example of this is the Grid Search for the hyperparameter turning in Scikit-learn.

Another method relatively similar to the Lasso approach is Ridge Regression [HHLB10] (or L_2 regularization for LR). In this case the penalty is represented by the sum of squared coefficients of the model multiplied by the regularization parameter. Thus, the expression minimized by the algorithm is defined by the following formula:

$$E(x, y) + \alpha \sum_{i=1}^{n} w^2 \tag{3.4}$$

However, even if L_1 and L_2 regularizations seem similar, they can work differently in some situations. For example, in the case of an increasing α, Ridge regression can produce relatively small coefficients for less relevant features instead of zero values in Lasso regularization. Moreover, L_2 produces more stable models in comparison with the previously described method where the coefficients tend to fluctuate even if there are small changes to the data.

Additionally, the specific point of interest from the embedded methods' group are tree-based methods as they are able to model non-linear relationships. Hence, we can use their feature importance values in order to identify influential variables.

Another useful feature preprocessing technique is **discretization** that transfers continuous variables into nominal attributes. That is, as a variety of ML algorithms require categorical data because they cannot handle continuous values directly. Moreover, it was observed that other techniques provide better results with nominal attributes as well [ER04]. As a reference [KK06, GLS+13] can be used in which surveys about different discretization techniques as well as an empirical analysis of different discretizers in combination with supervised ML algorithms are presented.

Also, such methods as **scaling** and **normalization** become essential for the estimators where the calculation depends on feature values and feature ranges. For example, for SVM where we need to calculate the vector product, or KNN where the calculation of the distances between different samples is required. For this purpose, there exist a lot of different techniques, for example, Standard Scaler that standardizes features by removing the mean and scaling to unit variance; MinMax Scaler that transforms features into a specified range based on their minimum and maximum values; Robust Scaler that transforms features based on percentile statistics, and therefore, robust to outliers; Quantile Transformer represents non-linear transformation where each feature is transformed in a way that its probability density function mapped into either uniform or gaussian distribution, and Normalizer that scales the samples into unit form.

3.5 Performance Space

Performance space is characterized by the mapping of each algorithm into various performance measures such as running time, solution quality, etc. Surprisingly, the Algorithm Selection Problem could also be solved by utilizing the EPM for each algorithm in the portfolio and then ranking them depending on the predicted value. Apart from the EPM, various techniques could be employed to solve the AS and APP problems. Moreover, an interesting fact is that choosing the best method for selection mapping is also the Algorithm Selection Problem. However, one of the main requirements for such a technique is that it will be relatively fast. Nevertheless, Machine Learning algorithms have commonly been used in solving the AS and APP problems where each heuristic from the portfolio runs on a problem set and performance models are therefore constructed.

Interestingly, choosing a simple method does not necessarily lead to a significant decrease in the model's accuracy. For example, Ridge regression for APP could demonstrate relatively good results in comparison with more sophisticated methods such as Random Forest. As an example, Xu et al [XHHLB08] preferred the model that performed well and is computationally inexpensive rather than choose the model with the best performance.

3.6 Related work

In this subchapter, a short overview on related work on the AS and APP problems as well as the applications of the EPMs to other problems will be provided. Moreover, areas that are relatively close to AS and APP and the concept of Empirical Hardness will also be briefly observed.

3.6.1 The Algorithm Performance Prediction Problem

Over past decade, researchers have investigated various methods in order to predict algorithm performance. The diversity of problem properties has been presented, and its impact on empirical algorithm performance has been examined. Various metrics have been tested as performance measures such as running time, solution quality, etc.

As previously mentioned, regression methods were first successfully applied by Leyton-Brown et al [LBNS06] to the Winner Determination Problem in Combinatorial Actions in order to understand algorithm-specific Empirical Hardness. The authors characterize the problem using 35 features based on graph representations, problem size and price features. As a performance criteria, the logarithm of running time of CPLEX method has been examined. For testing, widely-used problem distributions have been considered. Later, Almajano et at [ACRA10] applied a runtime prediction to a closely related problem of Mixed Multi-Unit Combinatorial Auctions where simple Linear Regression (LR) was employed; a high accuracy of the performance model was achieved.

Nudelman et al [NLBD$^+$04] applied the concept of algorithm performance based on runtime for understanding the empirical complexity of two SAT classes for random instances. Furthermore, a comprehensive feature set for SAT was introduced. The authors considered three different methods for SAT solving and tested performance models obtained on two instance distributions. As an example for the application of their approach, the authors developed the first version of a famous algorithm-selection tool called SATZilla, where they used the EHMs to choose the solver for a specific SAT instance. More information about this approach will be provided in subsection 3.6.2.

Kostuch et al [KS04] employed LR methods in order to predict the solution quality of a MIN-MAX Ant System for University Course Timetabling. The authors characterize the problem by 27 features based on problem size, room and period conflicts; however, the final model only consists 8 features. The constructed prediction model resulted in average error less than 17%.

Recently, Hutter et al [HXHLB14] performed an exhaustive empirical analysis of regression techniques for APP and combined features from the literature. Moreover, new features for the TSP, Mixed Integer Programming (MIP) and SAT problems were introduced. In this work, the best results were achieved with the use of the RF and Gaussian Processes Predictors. The authors also presented a comprehensive overview of related work with a focus on the ML techniques used. Additionally, the authors proposed several methods for improving prediction quality obtained by constructed models for highly parameterized algorithms and experimented with automated parameter tuning for algorithms using method parameters as input for the learning process.

Messelis [Mes14] used EHMs for APP for both Nurse Rostering and Multi-Mode-Resource Constrained Project Scheduling Problems (MMRCPS). These problems have been characterized by various extensive feature sets, so the impact of certain problem characteristics to performance prediction quality was examined. Various performance criteria such as solution quality, solver runtime and the quality gap between the obtained and best known solution have been investigated.

3.6.2 The Algorithm Selection Problem

One of the most famous AS systems is SATzilla [XHHLB08] that combines several State-Of-The-Art SAT solvers. SATzilla has already won several medals in SAT challenges. Additionally, an approach based on a latter version of SATzilla demonstrated the best results in the 2015 ICON Challenge [ICfDA]. In a previous version, the main idea was to run pre-solvers for a short amount of time in order to understand an instance's difficulty. If pre-solvers did not solve the problem, the feature computation time is predicted using the EHM. If the feature calculation is too expensive, the back-up solver is used. Otherwise, features are calculated, then solvers are ranked using the EPMs and finally, the best predicted algorithm is used to solve the SAT instance. An updated, newer version of SATzilla was presented in 2012 [XHHLB12] and won the 2012 SAT Competition. Instead of the EHMs, cost-sensitive classification is employed for each pair of algorithms in order to estimate the candidacy of methods and rank the solvers accordingly using a voting system. Finally, the algorithm with the highest number of votes is applied to solve SAT.

Another successful system for automatic AS is the AutoFolio framework [LHHS15] that showed also good results in the 2015 ICON Challenge [ICfDA]. This system makes use of FlexFolio [HLS14] which is comprised of a variety of AS approaches and the SMAC framework [HHLB11] that perform automatic algorithm configuration for hyperparameters' settings of AS approach. An interesting fact was observed; as SATzilla is the part of the FlexFolio framework, it was quite often chosen in various scenarios tested in [LHHS15].

However, different AS approaches have mostly been developed and tested on a particular combinatorial problem. Messelis [Mes14] constructed the Automatic AS framework for the MMRCPS problem in two different ways. First this was done by employing the EHM for the score prediction of each algorithm in the portfolio, which would therefore choose the algorithm with best result. The second approach is based on a classifier that predicts which algorithm will demonstrate the best result for a particular instance. It has been observed that the second approach yields better results for the MMRCPS in comparison with AS tool based on the use of EHMs. Furthermore, a successfully automated AS based on ML was applied in such problems as Graph Coloring [MS13, SMWLI13], TSP [KdCHS11, PM14], Nurse Rostering [MdC11], Quantified Boolean Formulas (QBF) [PT09] and others.

Nevertheless, other approaches could often be employed instead of ML techniques. For example, CPHYDRA [OHH^{+}08], a portfolio-based solver for the Constraint Satisfaction Problem (CSP) applied case-based reasoning in order to solve the ASP. In [LL98] the Performance of Branch and Bound, an algorithm for the Maximal Constraint Satisfaction Problem was estimated by iteratively generating random paths in the search tree.

Vasilevska et al [VWW06] proposed a hybrid approach for graph problems and the CSP where the selector was based on certain assumptions about a certain problem.

3.6.3 Algorithm Footprints

The concept of **Algorithm Footprints** is closely related to the idea of APP and Empirical Hardness. The concept was first presented in [CR10] where the authors emphasized the problems that developed techniques are usually tested on benchmark instances and the conclusion about the performance of the algorithm is drawn from the average performance on these specific problem sets in comparison with other methods. However, this does not guarantee that the method will perform well on other instance distributions, so it is important to know the boundaries of the algorithm performance. The authors proposed an "Algorithm Footprint" to use for this purpose that shows how an algorithm performance generalizes in instance space. The main idea is to identify the area where the method shows good results based on instance characteristics. However, this method also has certain drawbacks. For example, even if clusters help to understand the relationship between problem characteristics and algorithm performance, the actual footprint may still look different.

Nonetheless, this concept has been successfully applied on several problems. Smith-Miles et al [SML11] used the Algorithm Footprint in order to identify performance boundaries for two techniques developed for University Timetabling Track in the ITC 2007. The authors used Self-Organizing-Maps (SOM) with the aim of performance visualization and developed comprehensive set of characteristics. Moreover, using these features they visualized instance space where a clear distinction was made between real-world and generated instances. In addition, the authors visualized algorithms' performance; however, the regions of the methods were superimposed with only two small areas where the difference was clearly distinguishable. Later in [SMT12], new methodology for defining the Algorithm Footprint was presented, using the TSP as a case of study where features were collected from the literature. In the article the Principle Component Analysis was employed together with new metrics to measure relative performance. The authors also developed a new instance set with specific distribution using the Evolutionary Algorithm proposed in [SMvHL10]. As a result, the regions where each method performed well was easily recognized.

3.6.4 Other Applications

Besides the AS and APP problems, the EHMs could be put to use in various applications such as, for example, algorithm configuration in order to optimize performance of the method either on a per instance basis [HLBHH06] or for specific instance distribution [HLBHH06, HHLB11, HLBH12, JSW98]. Recently Algorithm Configuration Problem received significant attention from researches as generally methods are highly parameterized and finding good parameter settings is challenging [HHLB10]. Due to that, various automated algorithm configuration techniques have been developed. However, we will highlight just those that are built upon the EHMs.

In the first case, parameter settings incorporated into the model as additional features where the EHMs are used in order to predict performance of certain algorithm. This approach has been tested on several problems such as SAT [HLBHH06], the TSP and MIP [HXHLB14].

However, specifically researches were concentrated to find a configuration that performs well for an instance set or distribution. One of the approaches that employed the EHMs is the Sequential Model-Based Algorithm Configuration [HHLB11] where the promising method's settings are chosen using the EHM. Then the algorithm is run with the settings obtained in the previous step. Subsequently, the EHM is updated based on the resulting information. The process continues iteratively until a certain stopping criteria is met.

Another example of the EHMs application is in generating hard benchmarks where the EHMs are used to set the parameters for benchmark generators. However, detailed observation of this technique is out of scope this master thesis. More information about this application can be found in [LBNA+03].

CHAPTER 4

Algorithm Selection and Algorithm Performance Prediction for the ETP

In this chapter, the models for Algorithm Performance Prediction and Algorithm Selection for the ETP will be built using the Rice Framework described in chapter 3. Moreover, the experimental setup and environment for these experiments will be described in detail. Some methods for building AS and the EHMs and several evaluation criteria will also be introduced.

4.1 The Algorithm Space for the ETP

An overview of the approaches of different origins that have been developed for solving the ETP was provided in chapter 2. However, the most interesting algorithms for the AS and APP problems are the highly competitive State-Of-The-Art solvers developed for the ITC 2007 formulation, especially for investigating and comparing the impact of various problem characteristics on Empirical Hardness. For this purpose, exhaustive research has been performed on already existing approaches and contact with the authors has been established in order to obtain information on the original implementation of the techniques. With the overwhelming support of the researchers, 5 different solvers have been collected, namely [GAH12, Mue09, dS08, BSU14, ANI08]; however, [dS08] had to be excluded due to its incompatibility with the test environment. Moreover, [ANI08] has been removed because of incomparability of performance with other tested algorithms. As a result, these experiments have been performed using 3 State-Of-The-Art heuristics for the ETP.

One of the techniques considered in this work is the SACP solver developed by Mueller [Mue09], which was the winner of the ITC 2007. The method is an interesting candidate for this investigation as it comprises three various LS-based approaches: Iterative Forward Search, Hill Climbing Technique and Great Deluge Approach. Moreover, this solver still produces the best results for some benchmark instances. Because this solver was developed in order to comply with competition rules, the runtime limit must be set. As a result, the solution and its quality is stored.

Then an approach based on the GRASP technique was chosen as a candidate for Algorithm Selection and Performance Prediction. Developed by Gogos [GAH12], this solver combines different methods such as Hill Climbing with the Kempe Chain Neighborhood for the improvement phase, tournament selection, the backtracking phase and various ordering strategies for exam placement in the initial solution. However, similar to the previously described solver, a runtime limit is required. Moreover, this approach is considered to be one of the State-Of-The-Art solvers.

Another approach used in our investigation is the SA solver [BSU14] based on Single-Stage Simulated Annealing where the parameters were carefully tuned. This solver demonstrates relatively good performance on both generated and benchmark datasets. However, as a stopping criteria the number of iterations has been employed; therefore, it is necessary to configure the solver manually on a per-instance basis in order to compare the algorithms' solution score after a predetermined amount of time. Nevertheless, one solver run resulting in such parameters as the running time of the solver and the solution quality.

4.2 The Problem Space For The ETP

One of the main problems when evaluating different techniques to address the timetabling problem is the lack of sufficiently large benchmark sets of instances that consist exclusively of real-world problems. This complication arises because universities usually construct timetables just twice a year, and the diverse requirements for different institutions lead to a variety of soft and hard constraints. Hence, publically available datasets generally only contain a few examples, like the 12 instances in the Toronto dataset.

For the ITC 2007 formulation, the results of the algorithms in the literature are usually compared using 8 instances that were released for the Second International Timetabling Competition. However, the application of ML techniques requires at least a few hundred instances to be available, which could then be solved by using an instance generator.

In this master thesis, the union of 2 datasets have been used for the experiments, namely the dataset from the ITC 2007 competition [Gro] and 2248 generated instances that are similar to the real-world instances that have been obtained by the random generator provided by Prof. Andrea Schaerf. All generated instances were checked and discarded if deemed infeasible or similar to other already obtained instances based on a certain similarity filter. Moreover, the instances were created in order to gain better coverage of

Problem	**Conflict density (%)**	**No.of exams**	**No. of Students**	**No. of Periods**	**No of Rooms**
Exam 1	5.05	607	7891	54	7
Exam 2	1.17	870	12743	40	49
Exam 3	2.62	934	16439	36	48
Exam 4	15.0	270	5045	21	1
Exam 5	0.87	1018	9253	42	3
Exam 6	6.16	242	7909	16	8
Exam 7	1.93	1096	14676	80	15
Exam 8	4.55	598	7718	80	8

Table 4.1: Characteristics of the ITC 2007 problem set

feature space based on the ITC 2007 problem description. Detailed characteristics of the ITC 2007 dataset can be found in Table 4.1, and the properties of the generated set is provided afterwards.

The problem size of generated instances changes significantly across the instance space; the number of exams is located in a range between 100 exams and 1000 exams with 563 exams per instance on average, and the number of students varies between 386 and 29592 with a mean value of 9565 per instance. However, the available resources differ from one problem instance to another as well, but generally the variations are not very large. The number of rooms range from 1 to 54 rooms with the average amount of rooms per instance being 8, and the number of periods varies just between 10 and 80 with average number of 49 periods. Conflict density, which reflects how the problem is constrained regarding student enrolment, is mostly quite low for the dataset. It is located in a range from 1 to 18 with a median value of 10.

For the experiments, two datasets were combined. The first dataset was compiled for the SACP and GRASP solvers with the time point 210s having been obtained using the benchmarking program provided for the ITC 2007 competition [140]. The second dataset was combined for all solvers using the point of time where the SA solver finishes. All runtime data has been collected using a machine with eight Intel Core i7-5960X processors at 3.5 GHz with 32 GB RAM. All instances were solved simultaneously on cores with a memory limit of 4 GB.

As a performance criterion for APP, the solution quality has been chosen and measured for every solver run. Additionally, runtime for the SA solver has been recorded. However, as almost all solvers are based on some random criteria, each solver has been executed 7 times for every instance with different random seed value. As a result, a median value obtained by the solver on the same instance has been recorded.

However, the solution quality and runtime could differ significantly across the instance space (in several orders of magnitude: from hundreds for simple instances to hundreds of thousands for hard ones). Therefore, the logarithm transformation of the performance criterion has been chosen as a response variable for APP instead of a raw metric. Moreover,up to this point most of the researchers used the logarithmic transformation of the performance measure [HXHLB14, LBNS06, KS04] in their work because of the same problems. Furthermore, the results will be still relevant as most of the time for measuring the prediction quality of ML techniques the relative error has been used instead of the absolute.

For classification, due to high variance between different solver runs we examined only the instances where one solver dominated over the other solvers. We consider that the solver outperformed the other solvers if its score is at least 10% better than the scores of the other solvers in the dataset. Also, as for the `SA_time` dataset the SACP solver shown the best performance only in 8% of the cases, we have constructed the second version of the `SA_time` dataset where we only considered the results of the GRASP and the SA solvers.

In in order to understand the class distribution among newly constructed datasets, we calculated the fractions where each solver outperformed the other solvers on the training and the test datasets. As a result, we obtained the training datasets with the following class fractions: `SA_3_class` (710 instances) SA: 63.94%, SACP: 8.6%, GRASP: 27.5%; `SA_2class` (871 instances) SA: 66.6%, GRASP: 33.3% and `ITC_2class` (843 instances) GRASP: 76.15% and SACP: 23.84%. The characteristics and the class distribution of the test datasets are as follows: `SA_3class_test` (99 instances): SA solver: 64.64%, SACP solver: 7.07%, GRASP: 28.28%; `SA_2class_test` (110 instances): SA: 66.62%, GRASP:33.37%; `ITC_2class_test` (94 instances): GRASP: 70.21%, SACP: 29.78%.

4.3 The Feature Space For The ETP

4.3.1 Feature Engineering for the ETP

As noted, a representative set of features is a crucial aspect of AS and APP. In this section, the feature set for the ETP will be introduced that could reflect the Empirical Hardness for a certain algorithm, thus making it suitable for our tasks. For the ETP all features can be classified into three main categories, namely: domain-dependent characteristics that are related to the problem itself, features related to the landmarking concept and features obtained by reduction into the underlying graph problem.

Even if some characteristics seem redundant, the responsibility of feature elimination is left to the application of feature selection techniques which were described in section 3.4.

Furthermore, in some cases, the use of a ratio between two values or fractions can be also beneficial. Additionally, in the case that the series of values for some specific instance property are produced, various statistical measures have been utilized, namely:

- maximum (max), minimum (min) or for certain cases 5th (q_5) and 95th (q_{95}) percentiles;
- median (med) and mean values;
- standard deviation and skew
- variation coefficient and entropy;
- 1th (q_{25}) and 3th (q_{75}) quartiles of the population.

However, although in our first experiments the complete feature set was used, we excluded the features related to the Local Search (LS) as the results obtained by it was too close to the scores obtained by the other solvers. Due to that, the LS score was strongly correlated with the response variable for all solvers, and mainly used for Performance Prediction by most of the models. Our guess is that it may be the case because we took the Iterated Local Search [Mue09] that was implemented and optimized by Mueller, and the running time of LS (60s) was too close to the benchmark running time (210s). Hence, as we wanted to investigate other parameters that influences Instance Hardness for a particular solver, we performed the the experiments without the usage of the LS features.

In order to differentiate between newly introduced characteristics and the features, that have been already appeared in literature for the Educational Timetabling (including the UCTP), the latter are marked by asterisk (*).

Problem Dependent Features

In this section, features that are considered domain-dependent and cover the different aspects of the ETP will be introduced. These characteristics directly reflect the structural properties of the problem and can be classified into the following categories: features related to the problem size, features related to exam information and resource data.

Apart from most of the domain-specific features for the ETP that have been introduced, research that is closely related to that of this paper already exists. These investigations generally concentrate on another Educational Timetabling formulation, namely the University Course Timetabling Problem (UCTP) that is reasonably similar to the ETP. For example, Kostuch et al [KS04] presented a study that predicts the performance of the MIN-MAX Ant System based on 8 features for the UCTP. The authors chose several features that were based on period and room conflict graphs as well as some characteristics related to room constraints. Additionally, slackness was also considered. As mentioned in section 3.4, Smith-Miles [SMvHL10] characterizes the UCTP to obtain an Algorithm Footprint in the instance space using the following categories: 3 size-related features, 2 landmarking features based on the running of the DSATUR algorithm and 21 features based on Graph Coloring interpretation.

Related To The Problem Description

All of the features introduced below are straight from the problem description and are mostly based on the information about problem size. Moreover, as weights for objective function in case of soft-constraint violation could be individual for every instance of the ITC 2007 model, these parameters are considered as separate characteristics as well.

Features related to problem size (15):

- the number of students;
- the number of exams(*);
- the number of rooms(*);
- the number of periods;
- the number of days (concerning periods);
- slackness that is calculated by multiplying the number of periods to the number of rooms(*);
- the number of time-related constraints, the total number of constraints;
- the fraction concerning the constraints' type: the number of period and time-related constraints calculated over the total number of constraints;
- the ratio of the number of exams over slack;
- the number of constraints of each type: AFTER, COINCIDENCE, EXCLUSION, ROOM EXCLUSIVE.

Features related to the weights of objective function *in case of specific soft-constraint violation (7):*

- TWOINAROW, TWOINADAY, PERIODSPREAD, NONMIXEDDURATIONS weights;
- FRONTLOAD weight, threshold and value.

Features Related To The Problem Structure

The set of features related to the problem structure itself is the largest group for the ETP, and it describes the statistics and the relationships between different problem variables. These characteristics are mainly classified into two categories where the first group contains detailed information about exams, its conflicts with other exams concerning time periods, the data about exam length, possible periods' and rooms' assignments. Note that the second group incorporates all the data and statistics about available resources such as time periods and room information, including penalty information for the use of a certain resource. Additionally, to construct a feature set concerning exam data the information regarding period-based exam conflicts is also considered where two exams are considered to be in conflict if they have two students in common or a specified exclusion hard constraint. Another characteristics' group viewed separately are the properties of the problem with regard to resources, namely information about periods and rooms. Moreover, various ratios related to room occupation are also taken into account.

Features related to exam information (63):

- min, max, median, mean, standard deviation, variation coefficient, entropy, skew, q_{25}, q_{75}, the ratio of min over max, mean over max regarding:
  - the number of exams per student;
  - the number of students per exam (* mean, SD);
  - the number of conflicts
- median, mean, q_5, q_{95}, standard deviation, variation coefficient regarding:
  - the number of appropriate time periods per exam;
  - the number of appropriate rooms per exam (* mean, SD);
  - exam duration information;
- the number of exams
  - that only fit in one room (*);
  - that only fit in one period;
  - that fit in all rooms;
  - that fit in all periods;
- the fraction of exams
  - that fit all rooms over all exams;
  - that fit all periods over all exams;
  - that fit only one room over all exams;
  - that fit only one period over all exams;

Features related to period information (15):

- min, max, median, mean, standard deviation, variation coefficient:
  - for the number of exams per period;
- maximum, minimum, median, mean:
  - for period length information;
- maximum period penalty value;
- the number of periods with penalties;
- the number of periods to which each exam can be assigned without penalty;
- the fraction of the number of periods to which each exam can be assigned without penalty over the total number of periods;
- mean number of periods per day.

Features related to room information (19):

- median, mean, q_5, q_{95}, standard deviation, variation coefficient regarding:
  - the number of exams per room(* mean);
- maximum, minimum, median, mean over:
  - the room capacity;
- maximum room penalty value;
- the number of rooms with penalties;
- the number of rooms to which all exams can be assigned without penalty(*);
- the ratio of rooms that all exams can be assigned to without penalty over the total number of rooms;
- rooms' capacities available for one period calculated as the sum of all room capacities;
- total available rooms' capacities for all periods calculated by multiplying room capacities for one period by number of periods;
- the ratio of total capacities of all exams over total available rooms' capacities;
- the ratio of maximum exam capacities over maximum room capacity;
- the ratio of mean exam capacity over mean room capacity.

Features Related to The Landmarking Concept

In this section features that were obtained using the concept of landmarking [PBGC00] are provided; the concept for this approach was described in detail in section 3.2. Landmarking has been successfully employed for APP and ASP in such systems as SATzilla [XHHLB08], CPHYDRA [OHH$^+$08]. Additionally, it has been utilized for other combinatorial problems like TSP [HXHLB14], GCP [MS13, SMWLI13], SAT [XHHLB08, HXHLB14] and many others. For this case two different methods have been employed to explore the search space, namely a LS method that combines the Hill Climbing technique with a construction heuristic and a greedy approach that is based on the Greedy Coloring method.

Local Search Probing Features

LS techniques are often used for landmarking due to their simplicity and fast implementation. However, a complete LS run would be extremely time-consuming, so every LS run has been limited by a timeout and furthermore, the search stops if no further improvement has been made during a certain number of iterations. The LS used in this thesis is evaluated in 5 runs, and the timeout provided for this LS method is the 60s. The information from different LS runs is combined, so different statistical information has been obtained.

In this work, the implementation of the Hill Climbing technique from Mueller solver [Mue09]has been used. For neighborhood exploration, the swapping of exam periods has been employed so that only moves that improve the current solution are accepted. The initial solution has been constructed by iterative variables' assignment in order to prevent a hard constraints violation. The variables are sorted in decreasing order based on a ratio of the domain size to the number of hard constraints. If there is a tie, it is broken randomly. If it is not possible to assign the variable so as satisfy all hard constraints, the conflicting variables become unassigned. Additionally, conflict-based statistics is employed to avoid the repetition of conflict assignments. As a result, in most cases a feasible solution is already obtained after the construction phase.

For every search run, the initial score is stored after the construction phase and improvement steps namely iteration and scores where the overall solution quality has been improved. For each of these iterations, the score, iteration number and improvement per step is stored. As a result, the following **Local Search Probing features (12)(*)** was obtained:

- the number of assigned variables, the ratio of the number of assigned variables over total number of variables, the number of iterations corresponding to
  - min, max, median LS scores over all runs;
- min, max, median LS scores over all runs.

Greedy Algorithm Probing Features

Another popular approach related to the landmarking concept is the use of information obtained by the application of greedy methods, which is due to their short runtime and simplicity of implementation. For Examination Timetabling, most of the greedy algorithms are based on Graph Coloring approaches that were previously described in subsection 2.2.1. In this thesis the modification of the SATUR heuristic [Bre79] that demonstrated better results in comparison with other ordering methods has been used.

The main idea of an arrangement based on the saturation degree is to assign the exams with the least available timeslots first. However, in case of the ITC 2007 formulation, additional specified hard constraints need to be considered, namely constraints that require the exams be assigned into the same periods or provide specific exam ordering where one exam must be placed strictly later than another one. Due to the limitations of such constraints, the ordering has been adjusted so that the exams that are restricted by these constraints have the highest priority and must be placed first. Moreover, for exams with the AFTER constraint, the priority assignment is given just for one exam that should be assigned earlier. Subsequently, exams that are not restricted by the previously described hard constraints are ordered as in the SATUR algorithm. If two exams have the same number of possible assignments, the exam with a higher node degree is chosen. The method is designed so as to minimize the number of time periods used; however, room restrictions are considered as well.

As noted, the ETP is easily reducible to the famous Graph Coloring Problem. Furthermore, due to the construction of the described algorithm, the number of periods it obtains corresponds to the minimum number of colors needed to color the underlying graph without conflicts. Additionally, some profitable results can be attained by applying reduction from the GCP into the Independent Set Problem where the subsets of exams that have the same period (or color) assignment are considered to be independent sets of the graph. Therefore, some extra information based on the size of color classes can be obtained.

Accordingly, **the Greedy Algorithm Probing features (15)** consist of the following:

- the number of periods used (*);
- the ratio of the number of periods used over the total number of available periods;
- the number of unassigned variables, equals 0 in the case of a feasible solution;
- final score obtained, equals 0 in case of infeasible solution (*);
- computational time (*);
- min, max, median, mean, standard deviation, variation coefficient, entropy, skew, q_{25}, q_{75} regarding (*):
  - independent set size.

Graph Coloring Features

As previously mentioned in subsection 2.2.1,the ETP can be reduced to the GCP where exams correspond to the nodes and are connected by an edge if they have students in common or specified exclusion hard constraints. However, some of the hard constraints from the ITC 2007 formulation such as exam coincidence or exam precedence constraints could not be easily represented. Nevertheless, the fraction of these constraints compared with the total number of conflicts is relatively small. Thus, the described reduction can provide some useful insights into the problem structure.

From a graph representation, useful features concerning period-based conflicts' statistic can generally be extracted in addition to some features that have been introduced for the GCP that could also be reused in the ETP. Most of the characteristics presented below have already been observed in several studies for the GCP [MS13, SMWLI13]; however, a few novel features have also been introduced, such as the modularity related properties. Features that have been already presented in literature, are marked by asterisk.

Features Related to Graph Size

The use of size properties of the graph as features are important due to conflict representation that could directly influence problem difficulty. Moreover, the number of edges corresponds to the total number of period-based conflicts, and graph density conforms to the conflict density of the problem. Therefore, the conflict density of the EPT for the underlying graph G(V,E) will be calculated using the following formula where m = |E| and n = |V| corresponds to the number of edges (conflicts) and nodes (exams) respectively:

$$D = \frac{2 * m}{n * (n - 1)} \tag{4.1}$$

Therefore, the following properties regarding **the Graph Size(3)(*)** are employed for the ETP characterization:

- the number of edges;
- graph (conflict) density;
- the ratio of number of edges over the nodes.

Features Related To Clustering Coefficient

The Clustering Coefficient measures the tendency of the nodes to cluster together. Moreover, two types of clustering coefficients are distinguished: local and global. The local clustering coefficient [DWS98] used in this work is computed for every node and measures the probability that two randomly selected adjacent nodes of a certain node are connected to each other. In other words, it can be also interpreted as the ratio of the number of triangles connected to the node over the possible number of triangles.

A formal definition for the local clustering coefficient will now be provided. Consider a node $v_i \in V$ in a graph G (V, E) where N_i denotes the set of adjacent nodes to v_i: $v_j \in N_i$ if and only if there exists an edge such that $(v_i, v_j) \in E$. Let tr_i denote the number of edges $(v_k, v_m) \in E$ such that $v_k, v_m \in N_i$. Therefore, the number tr_i is precisely the number of triangles that node v_i forms with two of its neighbors. Let k_i define the degree of node v_i, where $k_i = |N_i|$. Accordingly, the maximum possible number of edges between the node neighbors in undirected graph is defined by the formula $\frac{k_i(k_i-1)}{2}$; finally, formula of clustering coefficient C_i can be obtained:

$$C_i = \frac{2 * tr_i}{k_i * (k_i - 1)} \tag{4.2}$$

Therefore, as the problem features different statistics based on the Local Clustering Coefficient has been employed. However, due to the drawback that the Clustering Coefficient does not take into account the node degree, vertexes with a low node degree have higher weights. Nevertheless, this problem could be solved with the Weighted Clustering Coefficient. The Weighted Clustering Coefficient is calculated by multiplying the Local Clustering Coefficient by the node degree, and various statistical measures have also been applied.

Therefore, the following features related to **Clustering Coefficient(21)(*)** have been obtained:

- min, max, median, mean, standard deviation, variation coefficient, entropy, skew, q_{25}, q_{75}, regarding:
  - Local Clustering Coefficient;
  - Weighted Local Clustering Coefficient;
- computational time for all computation regarding clustering coefficients.

Features Related To The Community Structure Of The Network

One of the issues in Network Science is the characterization of its community structure, or, more specifically, how easily the network could be broken down into groups that are densely connected internally and have sparser connections between communities. Modularity [New06b] is a measure that characterizes quality over possible divisions into communities in the network and is widely used in community detection algorithms where high modularity indicates a good split of the network into groups. Generally, the idea of modularity is in comparison to the quality of the current division in the given graph with the quality of the same division in a graph where edges are placed randomly. Formally, modularity can be defined as follows [New06b]. Suppose a certain network has already been split into communities. Then let e_{ij} define a fraction of edges in the graph between i^{th} and j^{th}, and e_{ij} denotes a fraction of the edges within community i. Then $a_{ij} = \sum_{j=1}^{n}(e_{ij})$ is the fraction of edges with one end-point in i_{th} community, and a_i^2 - is the probability that the randomly selected edge falls into i. Accordingly, the modularity Q of a graph G(V, E) is defined as the following:

$$Q = \sum_{i}(e_{ii} - a_i^2) \tag{4.3}$$

Moreover, in practice the Q $>$ 0.3 is an indicator of significant community structure. However, an exhaustive search over all possible splits for modularity maximization leads to an exponential amount of time, so the algorithm for community detection that was described by Neumann [New06a] has been employed. This technique is based on modularity maximization and belongs to the group of agglomerative hierarchical clustering methods. The method greedily searches for a possible division of the vertexes into groups in order to find a meaningful network split with high modularity. In detail, the idea is to start with communities represented by single nodes and iteratively join the communities' pairs that lead to the greatest increase (or smallest decrease) in modularity value. As a result, the split with the biggest modularity value is selected.

It is supposed that the characterization of graph structure could provide additional benefits in the description of the internal properties of the ETP instance; the following **features based on community detection (9)** were used:

- best modularity value obtained;
- the number of communities corresponding to the best modularity value;
- min, max, median, mean, standard deviation, variation coefficient over:
  - community size corresponding to the best modularity value;
- computational time (*).

Features Related To The Maximum Clique Problem

In graph theory, a clique is a subset of vertices in an undirected graph such that all vertices are adjacent to each other. Formally, in the graph G (V, E), a **clique** C $\leq$ V is such that for each pair of nodes $v_1, v_2 \in C$ there exists an edge $(v_1, v_2) \in E$. The maximum clique is a clique where there is no clique with a greater number of vertices. The problem of finding a maximum clique is famous in graph theory; moreover, it is proven to be NP-hard [Kar72]. Due to this importance, this problem has been widely studied; therefore, a large number of different approaches have been employed. Additionally, it has been investigated with regard to the characterization of the GCP [MS13, SMWLI13]. Furthermore, features based on the maximum clique could be profitable for describing the ETP as the number of nodes in the maximum clique corresponds to the minimum number of periods needed to obtain a feasible solution.

Due to the difficulty of the maximum clique problem and because of the relatively fast performance required for feature calculation, the greedy approach has been used in this research. The method starts with the single node as a clique, which iteratively enlarges the clique by adding a new vertex from the list of the vertexes that are connected with all of the nodes in the clique and has the highest node degree in comparison with other vertices from the list. If there are no nodes that can be added to the current clique, the method tries to drop one node from the clique in order to make it possible to obtain a bigger clique. For that, it forms the list of the nodes that are connected to all nodes in the clique except one and also were not previously dropped from the clique. The vertex in the clique that has the least number of connections to nodes in the list is the best candidate to drop. The method finishes if the current clique could not be enlarged and if there are no new candidates' nodes to drop. The algorithm is run for every node in the graph, and a series of cliques is produced. Based on the information obtained, **the features related to the maximum clique problem (11)(*)** are calculated:

- min, max, median, mean, standard deviation, variation coefficient, entropy, skew, q_{25}, q_{75}, regarding:
  - clique size;
- computational time.

4.3.2 Feature Preprocessing

The importance of the feature preprocessing step and its impact on prediction results has already been mentioned. In these experiments, the Python Scikit-learn 0.18.1 library [sci] has been used due to its simplicity, a large range of implementations of different ML algorithms as well as good visualization tools.

In our experiments the feature selection methods do not depend on variance, and all estimators mostly show better performance on the preprocessed datasets, therefore, the feature selection experiments have been performed on the standardized versions of the datasets for both classification and regression. Also, the discretized variants of the datasets have been used for classification task.

In this work, we investigated several feature selection approaches, the wrappers such as Recursive Feature Elimination (RFE) [GWBV02] where as the estimators Linear

Regression and Random Forest have been employed and Bidirectional Best-First Search (BFS) [Hal99] with the ClsSubsetEval criteria and limited backtracking from Weka.

Additionally, as filter methods are based on various assumptions about statistical significance and tend to evaluate every feature separately, we combined several filter techniques with the embedded methods into the compound approach, named as Filter Feature Selection (FFS or F), in order to obtain better results. For regression problem, the filter methods are the Pearson Correlation Coefficient (PCC) [Gay51] and the Maximal Information Coefficient (MIC) [RRF+11]. From the ensembles methods we used the coefficients from Lasso and Ridge regression and the features' importance provided by RF. Additionally, we experimented with Randomized Lasso (or Stability Selection) that resamples the train data and then, repeatedly ran Randomize Lasso on it, and chose the features correspondingly. For classification, the incorporated methods are Anova F-value, Mutual Information, Stability Selection that represented by Randomized Logistic Regression and features' importance from the Random Forest Classifier. After that, we scaled all the results and calculated a mean feature importance for each variable. The attribute will be chosen to be presented in the reduced dataset if either it has been ranked as one of the most important among others based on one selection technique, or it has a high mean importance based on the results of several techniques.

Based on empirical observations, different threshold values for the FFS method have been chosen. For regression problem, the mean threshold values is: 0.15, the Correlation Coefficient: 0.8, the Linear Regression coefficient: 0.8, the Lasso regression coefficient: 0.5,the Ridge regression coefficient: 0.5,the Stability Selection coefficient: 0.5,the MIC coefficient: 0.8,the Random Forest importance: 0.5. For classification problem the mean threshold value: 0.2, the Anova value: 0.5, the Stability Selection coefficient: 0.3, the MIC coefficient: 0.8, the Random Forest importance: 0.5.

Additionally, we constructed a new extended feature set that contains composite features in addition to the original features, where the composite features are obtained in the following way: for each features' pair f_x, f_y we calculated multiplication $f_x * f_y$, division f_x/f_y, addition $f_x + f_y$, and subtraction $f_x - f_y$. However, as in that case the number of features increase drastically in comparison with the number of samples (more than 68 000 vs 2000), the models will overfit and take an extreme amount of time for training. Therefore, we only experimented with the reduced version of the features. However, as the wrappers are computationally expensive and our experiments on the original datasets reduced by FFS show that the results of the estimators do not worsen and even improved, we applied the wrappers only after the usage of the FFS method. Due to high computation cost we had to run the RFE procedure two times: at first, with the step equals ten where ten features may be removed at once, and then, a second time with the step one for the reduced dataset obtained on the previous step. These datasets are named RFE_1 and RFE_2 correspondingly. Additionally, we have constructed the datasets using the BFS method applied to the extended datasets reduced by FFS.

For regression tasks, as evaluation criteria for RFE `RMSE` has been used, while `ROC_AUC` and `Log_Loss` metrics have been employed for the binary and the multiclass classification cases respectively.

For scaling, several algorithms represented in Scikit-Learn [sci] have been employed. These methods are Standard Scaler, MinMax Scaler, Robust Scaler, Quantile Transformer and Normilizer. Although all the methods (except Normalizer) show comparable results in our preliminary experiments, Standard Scaler outperforms other techniques for the most estimators. Therefore, it will be used as a default scaling method for all the experiments, unless otherwise noted.

Also, as it has been shown, that for classification problems discretization of continuous variables can have positive effects on prediction results [KK06, DKS95]. Therefore, we experimented with several discretization methods, namely Equal-Frequency, Equal-Width methods with various number of bins represented in Orange library [DCE+13] and the MDL discretization with Mutual Information[FI93] and Kononenko criteria [Kon95] from Weka [HFH+09]. According to our preliminary experiments, the MDL discretization with Kononenko criteria shows the best performance for the ETP problem across different estimators, therefore, it has been chosen as a representative of discretization methods for further experiments.

4.4 The Performance Space for the ETP

As noted, ubiquitously various ML have been used for the AS and the APP problems. Moreover, it has already been pointed out that choosing the best algorithm for these problems is the Algorithm Selection Problem, making it undecidable [Guo03]. Manually testing all available ML approaches is time consuming, so several State-Of-The-Art techniques have been chosen from different categories from the Scikit-learn library. However, a detailed observation of the ML algorithms is not within the scope of this work. For that please refer to this book with a comprehensive survey of various ML techniques [WFH11].

The regression techniques used in this work are Linear, Ridge, Elastic Net and Lasso regressions. Despite their simplicity, many researchers preferred the use of these methods instead of more sophisticated ones due to their low computational cost and relatively good results [XHHLB08]. Moreover, they have already been successfully applied for APP for several combinatorial problems such as the Winner Determination Problem for Combinatorial Actions [LBNS06], SAT [NLBD+04] and Course Timetabling [KS04]. Additionally, for classification problem we tested the Naive Bayes algorithm.

Another group of methods investigated for the ETP performance prediction and algorithm selection is tree-based approaches. These techniques have become fairly popular recently due to their relatively fast model building and often perform the best compared to other techniques [HXHLB14, Mes14]. Moreover, various tree-based methods for regression and classification have been implemented in the Scikit-learn library. While we have tested most of them, the Gradient Boosting Machine (GBRF) has been left out of the final evaluation as other versions of Decision Trees, such as ID3/4/5 decision trees tend to have less predictive power compared to the boosted version. Additionally, we included the Random Forest estimator in our experiments.

The regression and classification implementations of popular ML approaches represented within the Scikit-learn library such as KNN, Multi-layer Perception (MLP) and SVM regressors that are considered to be among the most influential ML techniques [WKRQ$^+$07] has also been included. We experimented with various configurations and the proper tuning of the estimator parameters as well.

For regression case, in order to measure the quality of the constructed models, several performance metrics have been introduced, namely the Root Mean Squared Error (RMSE) of log and raw response, and the CC between the predicted and actual value of response variables has been calculated. Also, the Median Absolute and Percentage Errors (MedAE and MedAPE), the Mean Absolute and Percentage Errors (MAE and MAPE) have been used.

For classification, it is essential to use a variety of classification metrics, because different estimators are based on optimization of different loss functions, especially, in the case of imbalanced datasets where we can obtain the Accuracy Paradox. Further, we separated the metrics used for the binary and the multiclass classifications, as some of them may be less efficient in either case. However, one of the main metrics is still classification accuracy – the portion of correctly classified instances. In addition, for the binary cases we used Precision, Recall, F1-score, Area Under ROC curve (ROC). For the multiclass problem F1, Accuracy, Recall and Precision metrics in both micro and macro variants have been employed. Moreover, in case if the classifier was able to produce the probability of belonging to a certain class, the log-loss metric has been calculated also.

It is a widely-known problem that ML algorithms tend to overfit when the same dataset is used for learning and testing. In order to prevent this, a 10-fold cross-validation has been used where the sample is randomly divided into 10 subsets. The process is then repeated 10 times where each subsample is used precisely once as a test set for model, and the rest is employed as a training set. As a result, the average performance statistic is reported over all runs. Moreover, we additionally validate the performance of the best model obtained by using a set of new unseen instances as a test set.

For statistical significance testing between the results of various estimators, we have used the Welch's statistical test [Wel47]. The test has been performed with the threshold value p=0.01.

CHAPTER 5

Experimental Results

In this chapter, we present the results of the experiments for the APP and AS problems for the Examination Timetabling Problem using different classifications and regressions algorithms. Also, the performance of the regression and classification methods is investigated in terms of different performance measures discussed before. In addition, using various feature selection methods, the importance of particular features for APP and AS is investigated in detail, and then, employing the information obtained, the feature set is reduced. Finally, the best estimators together with the reduced feature set are used for prediction on the test set of unseen instances. Additionally, we study hyperparameter optimization for different estimators in detail.

5.1 Feature Selection Results

In the next subsections, we look closely at application of the feature selection approaches for Algorithm Selection and Performance Prediction. Moreover, the importance of the feature families and its impact into estimator performance are described for each dataset separately. Additionally, our feature selection results include information about the most frequently selected features that have been chosen by the different methods and statistics regarding the most frequent features that are used as constituents for the compound features in the reduced versions of the expanded datasets. For that, we decomposed each feature in the reduced datasets into original components and then, count these features together with individual attributes. Moreover, a feature in an expanded dataset is considered to be related into two groups if its constituents are from the different attribute classes, otherwise it corresponds to a single group. All statistics and all features chosen by the different methods for APP and AS can be found in section A.1 and section A.2 respectively.

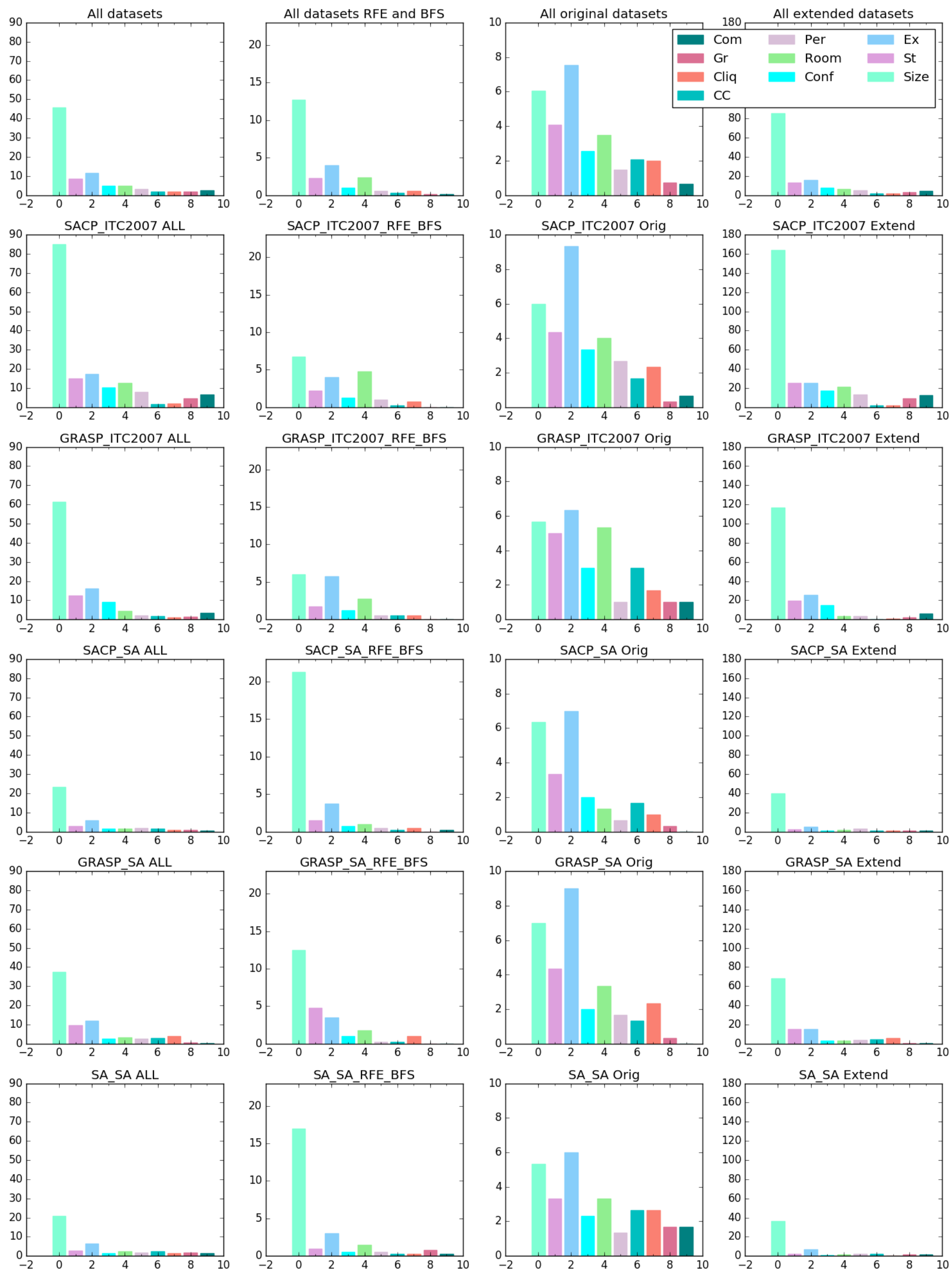

Figure 5.1: The average number of features per attribute class chosen by the methods for the APP problem

5.1.1 Application Of The Feature Selection Methods For Performance Prediction

In this subsection, we briefly analyze the feature selection results for Performance Prediction. The detailed results regarding the different estimators tested on the reduced datasets can be found in section B.1. The average number of features per attribute class chosen by the methods which has been normalized by the number of the selection results used can be seen on Figure 5.1.

First, let's start with the performance comparison of the feature selection techniques. As mentioned, FFS has been used as a preliminary step in order to reduce computational time. Although we used comparably low threshold values for the FFS method in order to retain important features in the datasets, we were able to reduce the features in the training dataset from 184 to 65-75 for the original datasets and by factor ten (from 68 000 to 6000-10000 features) in the case of the extended datasets. Further, while retaining good estimator performance, we were able to significantly reduce the datasets by the RFE method for the original and the expanded datasets. For the original `ITC2007_time` datasets, RFE keeps about 10-17 features, while for the solvers in the extended SA versions of the datasets it leaves about 25-35 features. On the other hand, while the number of features in the datases obtained by BFS were even lower, about 5-15 attributes per the dataset, in comparison with RFE the regression results have been worsen. This can be clearly recognizable by the quantitative comparison of the performance metrics and the visual comparison of corresponding scatter plots.

The opposite situation may be noticed on the test datasets. Although performance of the estimators based on the RFE results drops significantly, the estimators show rather good performance on the dataset reduced by BFS. Therefore, we may conclude that the features chosen by BFS are appropriate for more general models, while RFE tends to overfit. However, the RFE results may be further improved, but this question needs further investigation.

Now, let's analyze the importance of various feature groups. Surprisingly, we can see that most of the time we obtained similar results for the different solvers. It is clearly recognizable that the group of features that corresponds to the instance size prevails in most cases, especially for the extended datasets. Another attribute group that was considered the most statistically significant by various filter and embedded methods in the original datasets is the features related to exam information. These attributes mostly consist of various statistics about exams per student. Less frequently, but still employed in the reduced datasets, are the features based on period, student, conflict and room information. Interestingly, the features which are based on the reduction of the Examination Timetabling Problem into the graph problems, such as Community features, Greedy algorithm features, Clustering Coefficient and Clique size attributes are less frequently chosen. This can be the case because we consider only the hard constraints for the graph construction and do not take into account the soft constraints which are used for the score calculation. Therefore, these attributes may be more useful in case we want to predict whether the ETP algorithm will be able to obtain a feasible solution. Also, as in most cases it may be impossible to satisfy all soft constraints, another possibility would be to construct several graphs based on the hard constraints and some soft constraints and rerun all graph-based algorithms on these new graphs.

The important point from the previous observation is that as the most selected feature groups are based on the problem formulation, therefore, they are easy to compute. On the other hand, a lot of algorithms for the corresponding graph problems may take a lot of computational time and are also harder to implement. Therefore, it may be also interesting to investigate the Performance Prediction Problem using only the features that can be directly obtained from the problem formulation. However, it is also out of scope of this work, and left for future work.

Concerning individual attributes, based on statistics, we can see that the **ExamPerStudentSD** is the most frequent feature chosen, which is individually presented in 30% of the different datasets. Additionally, the statistics about the **ExamPerStudent** information frequently presented in the reduced datasets: the **ExamPerStudentMax** (25%), the **ExamPerStudentMean** (25%),the **ExamPerStudentVarCoef** (20%), the **ExamPerStudent1Q** (15%),the **ExamPerStudenEntropy** (15%) etc. This result seems logical as the violation of the soft constraints such as, for example, two in a row, two in a day or the period spread constraints, depends on placements between different exams taken by one student, and therefore, the less exams one student takes, the easier it will be to find a better solution. Moreover, in most cases the weights of the scoring function for the violation of two in a day or especially for two in a row are greater than the weights for other soft constraint violations, and moreover, the violation of these constraints happened more often than the violations regarding exams or room placement.

The **NumberOfStudents** feature is dominant in comparison with other features (30% vs 2%) based on both the expanded decomposed and the original feature sets. The importance of the attribute may be explained by the same argument as above, because the possibility of violating one of the student-related constraints increases with the number of students to be placed. A similar situation takes place with other related groups of the attributes such as, for example, the features corresponding to the **StudentPerExam** information. This can be the case because due to room and student constraints larger exams are usually harder to place. Additionally, there is a special penalty about large exam allocation where it is desired that these exams will be assigned later in a timetable. Less frequently, but still relatively often, the other features related to the problem description are used as the constituents in the extended datasets, namely the NumberOfDays, the NumberOfRooms, the NumberOfPeriods, the NumerOfDays, Slackness , WeightPeriodSpread, FlontLoadWeight, the NumberOfExclusionConstraints, the NumberOfCoinsidenceConstraints, the NumberOfExams, the NumberOfRooms and others.

Surprisingly, only one feature from the graph features, namely the **CliqueCalculationTime**, has been chosen by the wrappers in 25% of the datasets. Although at first sight the choice of this feature is not obvious, it directly reflects the complexity of the related conflict graph. As we used the greedy approach that gradually tries to enlarge the clique based on neighbor information, calculation time depends on the graph density, that corresponds to the conflict density.

5.1.2 Application Of The Feature Selection Methods For Algorithm Selection

In this subsection, we briefly analyze the feature selection results for the Algorithm Selection Problem. Visual comparison of the feature selection results obtained on the datasets regarding the different attribute groups can be found on Figure 5.2. Detailed results achieved on the reduced datasets by the classification algorithms can be found in section B.2.

As we can see, the attribute selection methods were able to reduce the number of features noticeably, while retaining rather good estimator performance. However, the number of features differs drastically depending on the preprocessing methods and the feature selection techniques used. For example, for the extended version of the `ITC_2class` dataset RFE left 239 and 259 attributes for the standardized and the discretized versions respectively, while BFS considered important 30 and 14 variables only. Surprisingly, the performance of the estimators on the datasets reduced by both methods are comparable, and moreover, in some cases the estimator performance on the subsets reduced by BFS is even better. However, in most cases the tendency is that BFS chose less variables than RFE, and based on the classification results, it was more appropriate for our task. However, RFE is also a powerful method, and can be further improved by the usage of another estimator or proper parameter configuration of the algorithm employed. Nonetheless, this question is left for future work.

In general, we may acknowledge that the feature subsets that are based on the reduction of the extended datasets are more informative than the standardized ones. This may be the case as the compound features help to model additional relations that are absent in the original datasets. Another observation is that, in general, discretization improves the results while decreasing the number of features needed for prediction. One possible explanation is that the MDL discretization takes into account the target variable, and therefore,the discretized features may be more informative compared to the original or the standardized variables. However, detailed investigation of this question is out of scope of this thesis.

Let's start to analyze various feature groups chosen by the FS techniques. Notably, one of the most selected group of the features are those related to the results of the greedy heuristic. Moreover, the following attributes regarding the heuristic results itself such as the **GreedyScore**, **GreedyNumberOfUnassignedVariables**, **GreedyNumberofUsedColorsPerPeriod** and **GreedyNumberOfUsedColors** are the most frequent constituents for the compound features in the reduced datasets. The reason might be that the performance of the Greedy Coloring Algorithm is strongly correlated with the performance of another ETP algorithm. Moreover, the number of used colors in the best

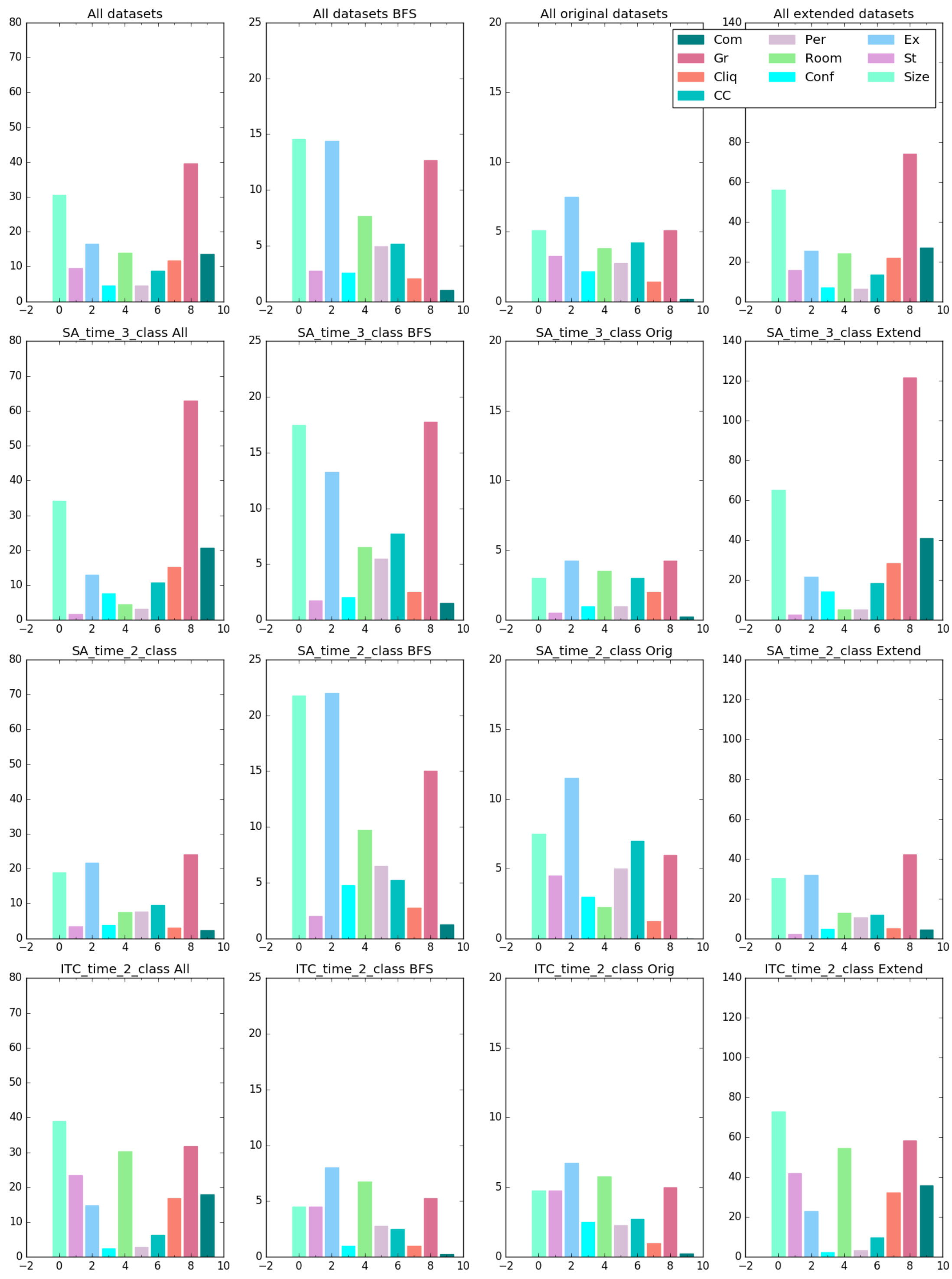

Figure 5.2: The average number of features per attribute class chosen by different methods for the AS problem

case corresponds to the minimum number of periods required for the construction of a feasible timetable. Other frequently chosen variables are the features related to the **GreedyIndependentSet** characteristics. These attributes may also provide additional information about performance of the algorithms as it corresponds to the exams that potentially might be assigned into the same timeslots, and therefore, may correlate with instance complexity. As future work, it may be interesting to implement other features related to the landmarking concept, and investigate their impact into the results of the AS solver.

Another group that is deemed to be important according to different FS methods is the information regarding the instance size. One of the most frequently chosen feature corresponds to the **NumberOfExamsOverSlack** that characterizes the distribution of the exams over available resources, and therefore, may be helpful. Another interesting fact is that specifically important are considered the features related to the different constraint types: the **NumberOfCoincidenceConstraints, the NumberOfAfterConstraints, the NumberOfRoomExclusiveConstraints** and the others. One explanation might be that the higher number of constraints may cause inefficiency of some algorithms, and as a result, increase the number of the soft constraint violation of the ETP heuristic. However, understanding the instance characteristics where one of the ETP heuristic may potentially have problems, could help to improve the heuristic itself. Nonetheless, this observation need further investigation.

There are several other features groups that are supposed to be significant by the FS algorithms, such as room and exam information. These attributes include also the **ExamPerPeriod** and **StudentPerExam** information and the features related to the resource information such as room capacity information and **RoomPerExam** statistics. Similar to the previous group, these variables may define some instance specific properties of the ETP that characterize the instance complexity for the algorithms, and therefore, are considered to be important.

In most cases, the other features groups are chosen for some datasets, however, relatively less frequently. The least chosen feature groups include the attributes related to the period and the conflict information. However, some features that may be considered to be presented in these groups are already included into other attribute groups, for example, as **ExamPerPeriod** information mentioned before. Therefore, it is harder to conclude whether these feature groups are fairly insignificant for our task.

5.2 Performance Prediction for the ETP

In this subsection, we carry out the experiments for the Algorithm Performance Prediction problem. At first, we started with evaluation of various regression methods using cross-validation on the training datasets, namely the `ITC2007_time` and the `SA_time` datasets using the original feature set. Also, for the experiments on the reduced datasets we used the estimators that have shown the best results across the original datasets. At last, we train these estimators on the standardized versions of the `SA_time`

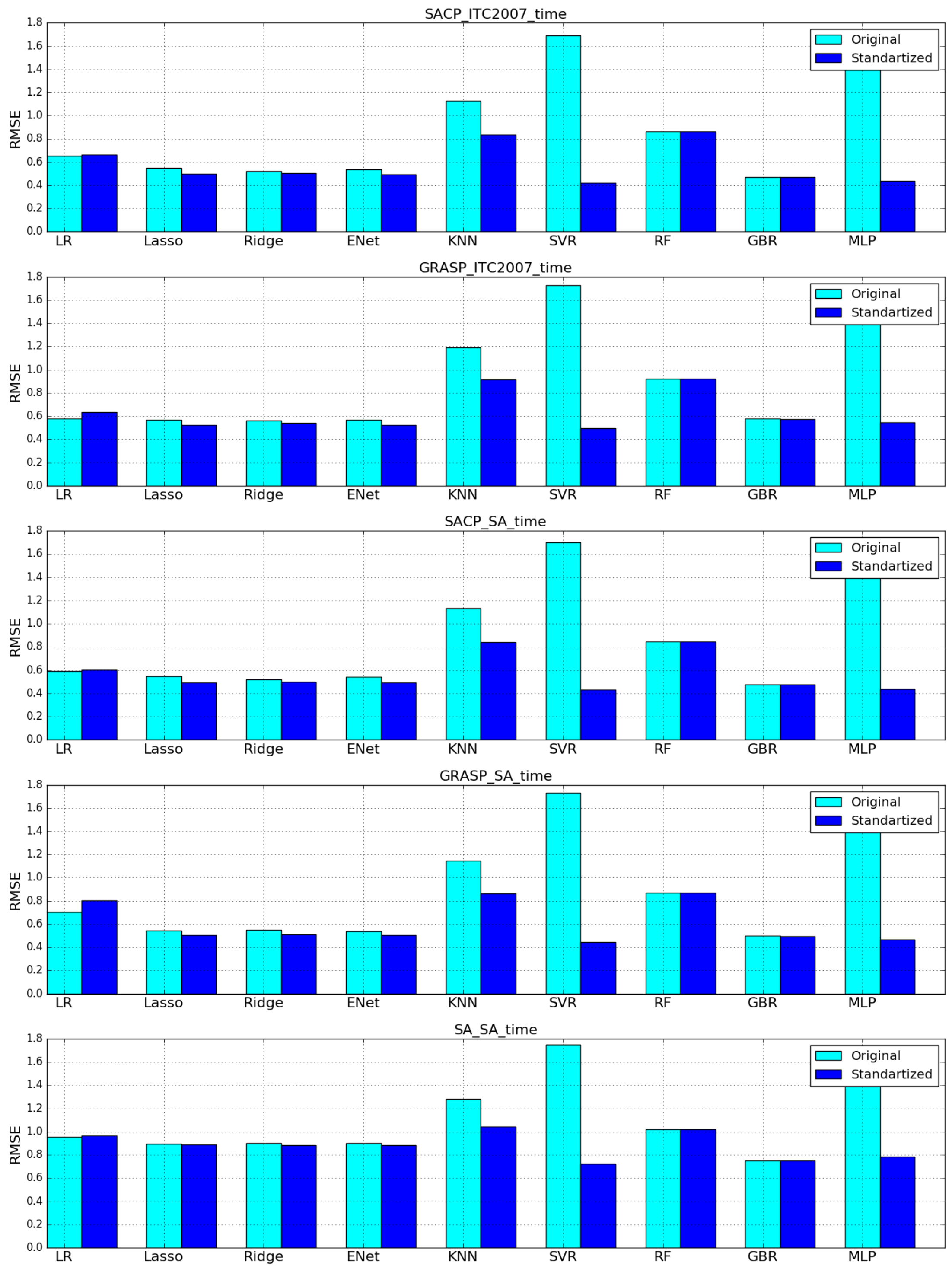

Figure 5.3: Visual comparison of RMSE obtained by different regression models on the original and the standardized versions of the `ITC2007_time` and the `SA_time` datasets

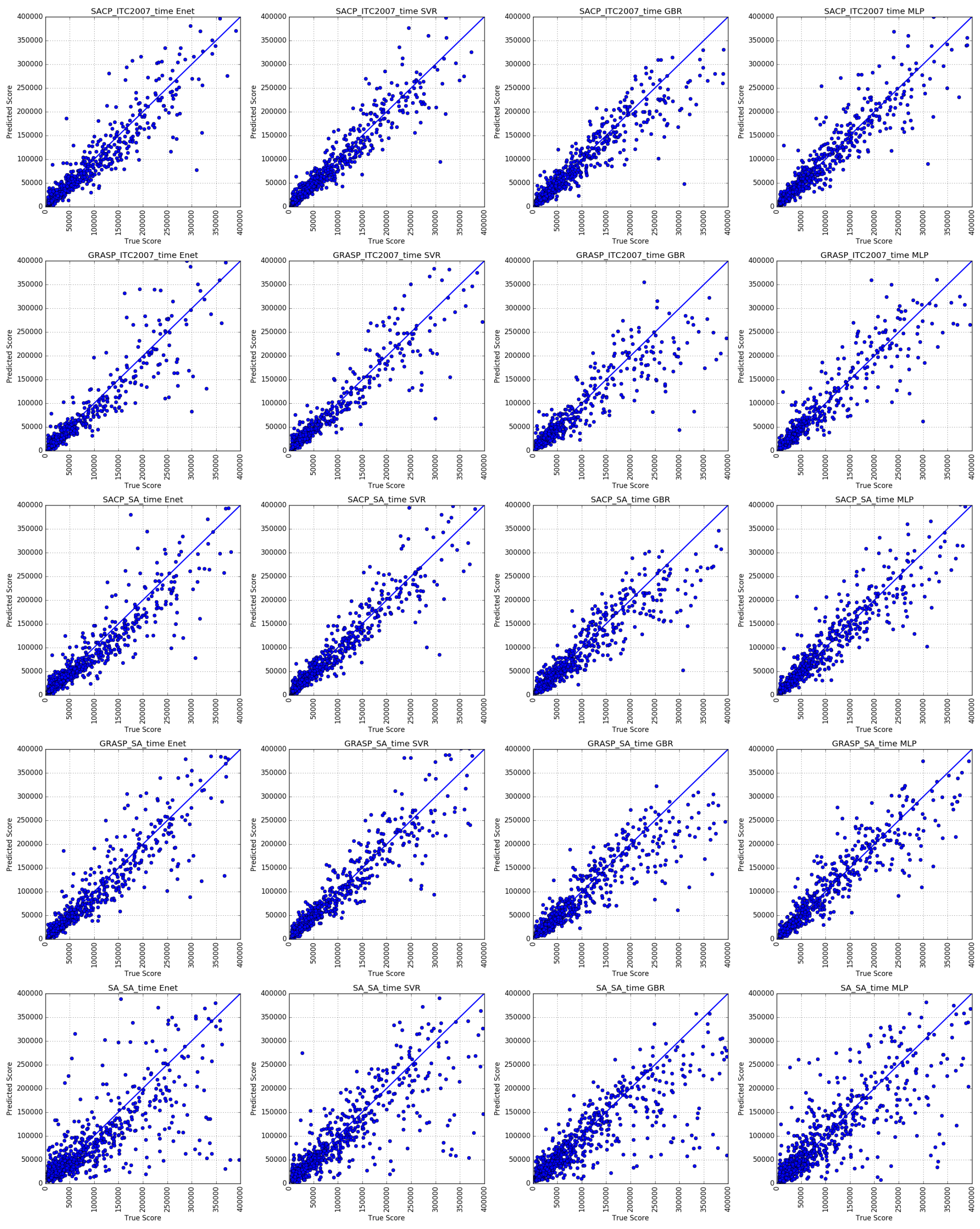

Figure 5.4: Scatter plots obtained by ENet, SVR, GBR and MLP for the solvers on the standardized versions of the `ITC2007_time` and the `SA_time` datasets

and the ITC2007_time datasets, and then, test them on the test datasets named ITC2007_time_test and SA_time_test respectively. The best parameters found for each solver in the datasets can be found in section C.1. The results obtained for the datasets can be found in section B.1. Quantitative comparison of RMSE obtained by the different methods can be seen on Figure 5.3 and Figure 5.5 for the training datasets and on Figure 5.7 for the test datasets.

Also, in order to gain a better understanding of the model performance, the scatter plots for some estimators have been employed, where the x-axis corresponds to the predicted score and the y-axis corresponds to the true score. The constructed scatter plots for the training set can be found on Figure 5.4 and for the test dataset on Figure 5.6. Additionally, the scatter plots for the reduced training datasets can be found in section B.1.

5.2.1 Performance Prediction On The Training Dataset

To begin with, let's start with analysis of the performance of different estimators on the original dataset. We can see that some of the estimators were able to predict the solver's score fairy accurately for all solvers even without any additional preprocessing. Besides the quantitative comparison, we can acknowledge this conclusion by using the scatter plots where in most cases the predicted score was relatively close to the true score. Even in the case of the SA solver, despite some outliers, for most instances the results were relatively accurate.

In almost all cases the GBR provided the best results for the original dataset, except the GRASP for the ITC2007_time dataset where Ridge regression won first place. However, after scaling SVR outperformed other algorithms for all solvers. Regardless of simplicity, LR-based techniques showed good results, especially Ridge and ENet. Interestingly, after preprocessing, Lasso, Ridge and ENet slightly improved their results, while the performance of LR became a bit worse. Moreover, for the GRASP solver in the standardized version of the ITC2007_time dataset LR dropped its performance, and it was unable to predict some outliers as RMSE notably increased, while MedAPE and MAPE values remain close to the non-standardized version. It is also clearly seen that KNN, SVR and MLP suffered from non-scaled data, while after preprocessing its performance increased significantly. However, even after scaling, KNN still provided poor performance in comparison with the other methods, even with simple LR. The results obtained by RF and GBR on both versions of the datasets are statistically insignificant, as expected, as tree-based methods do not usually require any additional scaling. Also, it worth mentioning that in our experiments the GBR always outperformed RF, and similar observations have been examined in other works, for example in [RNM06, RNMY08]. The MLP regressor after standardization showed good performance, however, the training process remained computationally expensive.

From the scatter plots we can observe that models constructed for the SA solver consisted of more outliers than the models for the other solvers. Interestingly, the best models have been obtained for the SACP solver on both datasets, and moreover, with regard to

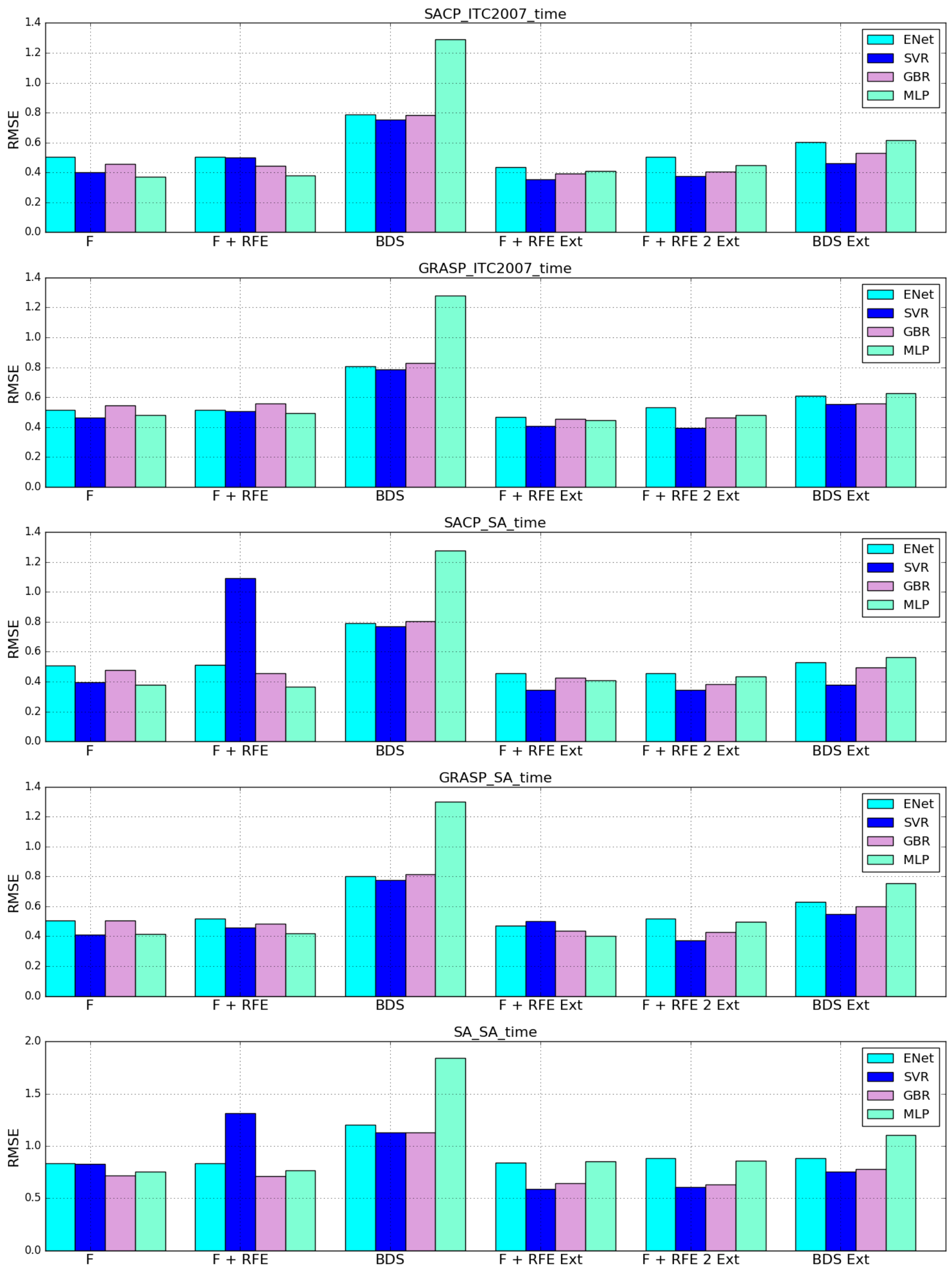

Figure 5.5: Visual comparison of RMSE obtained by different regression models on the reduced versions of the `ITC2007_time` and the `SA_time` datasets

the scatter plots, the difference between the predicted and the true value is relatively low even for the outliers, especially in comparison with the SA solver.

Let's discuss the impact of the feature selection techniques into the original and the extended datasets. In most cases, the best results were obtained either by MLP or SVR with only one exception where GBR obtained the best results for the SA solver for the FFS and the RFE methods. However, in most cases GBR obtained the results relatively close to the best, while the ENet results were comparably worse.

Also, we can see that for the original datasets, the FFS method was able further improve the results and reduce the number of outliers, however, the number of chosen features still remained relatively high. Furthermore, after application of the RFE method there was no significant decrease in comparison with the previous results and the performance is fluctuated just slightly. However, in some cases the feature selection led to an increase in a median error while RMSE was decreasing. Therefore, sometimes it is hard to conclude whether the results have been improved or worsened. The opposite situation can be seen in the case of BFS application: while the number of features is similar to that obtained by the RFE method, based on RMSE and MedAE metrics the results were worsened significantly, even by several times in some cases.

For the extended version of the datasets, after the application of FFS and RFE, we obtained a small decrease in RMSE while MedAE and MeanAE sometimes even increased, however, they still remain pretty low. A similar situation, as in the case of the original dataset, was obtained by the application of BFS where the method worsened the results, even in comparison with results obtained by other methods on the original dataset.

5.2.2 Performance Prediction On The Test Dataset

Although we already carried out some experiments using cross-validation techniques, in the real-world we need to predict the performance of the algorithm based on new instances that have not been used during the training process. Moreover, some ML methods tend to overfit, and as a result, do not generalize well. Therefore, we used the best performing regression methods from the cross-validation experiments, trained them on the standardized versions of the `SA_time` and the `ITC2007_time` datasets, and then, tested them on the test datasets named `ITC2007_time_test` and `SA_time_test` respectively which consist of 249 unseen instances. Additionally, we would like to mention that in order to obtain appropriate results and do not provide additional information about the test set, standardization techniques have been trained on the training dataset only, and afterwards, this model has been applied to the test dataset.

Also, we investigated the behavior of the performance models on the reduced versions of the datasets. In the experiments, GBR and SVR with the optimal parameter settings obtained during the cross-validation process have been used. The detailed results received by the regression models can be found in section B.1. The best parameter configurations for the estimators tested can be found in section C.1. Visual comparison of the constructed

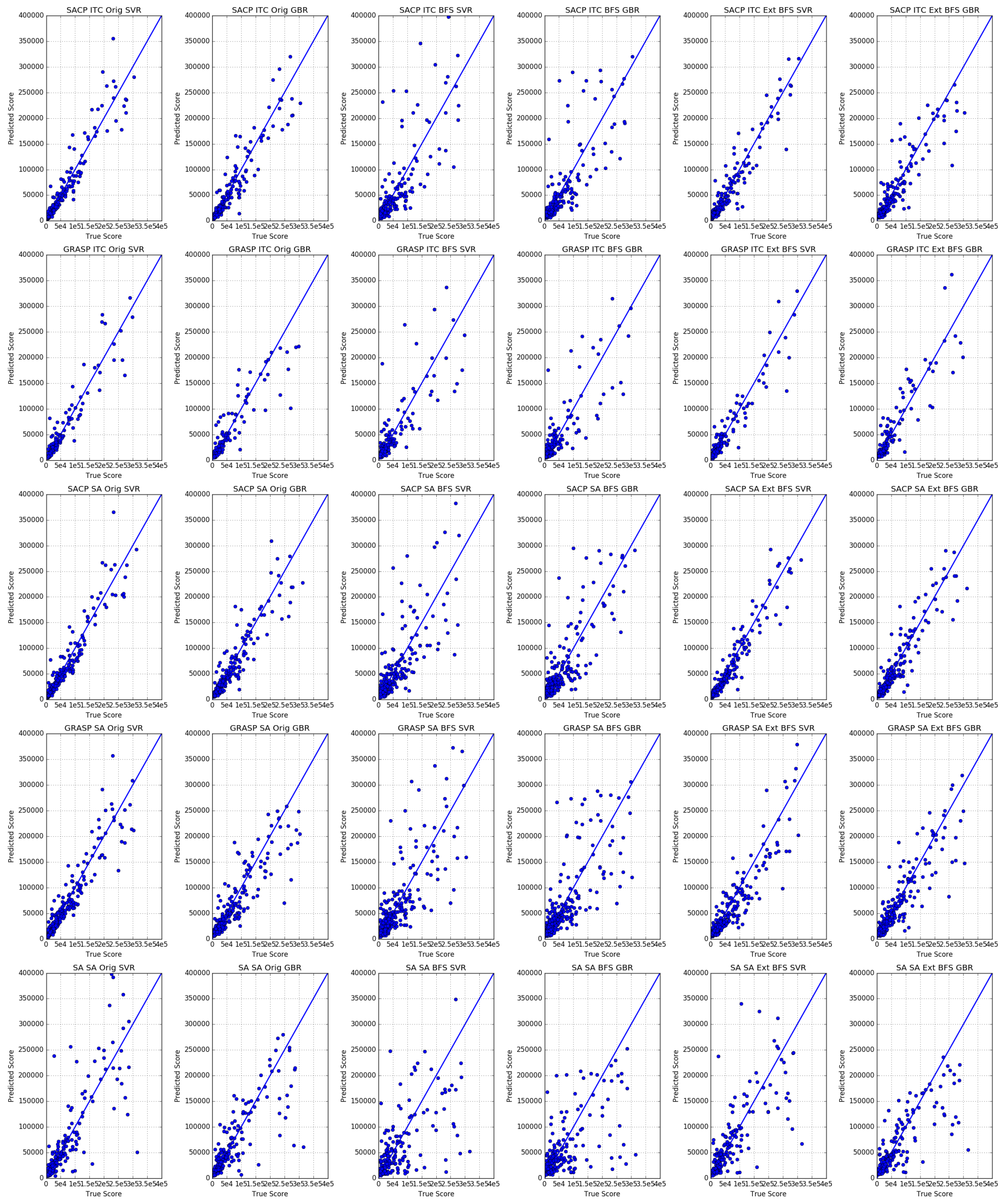

Figure 5.6: Scatter plots obtained by the regression models on the test set

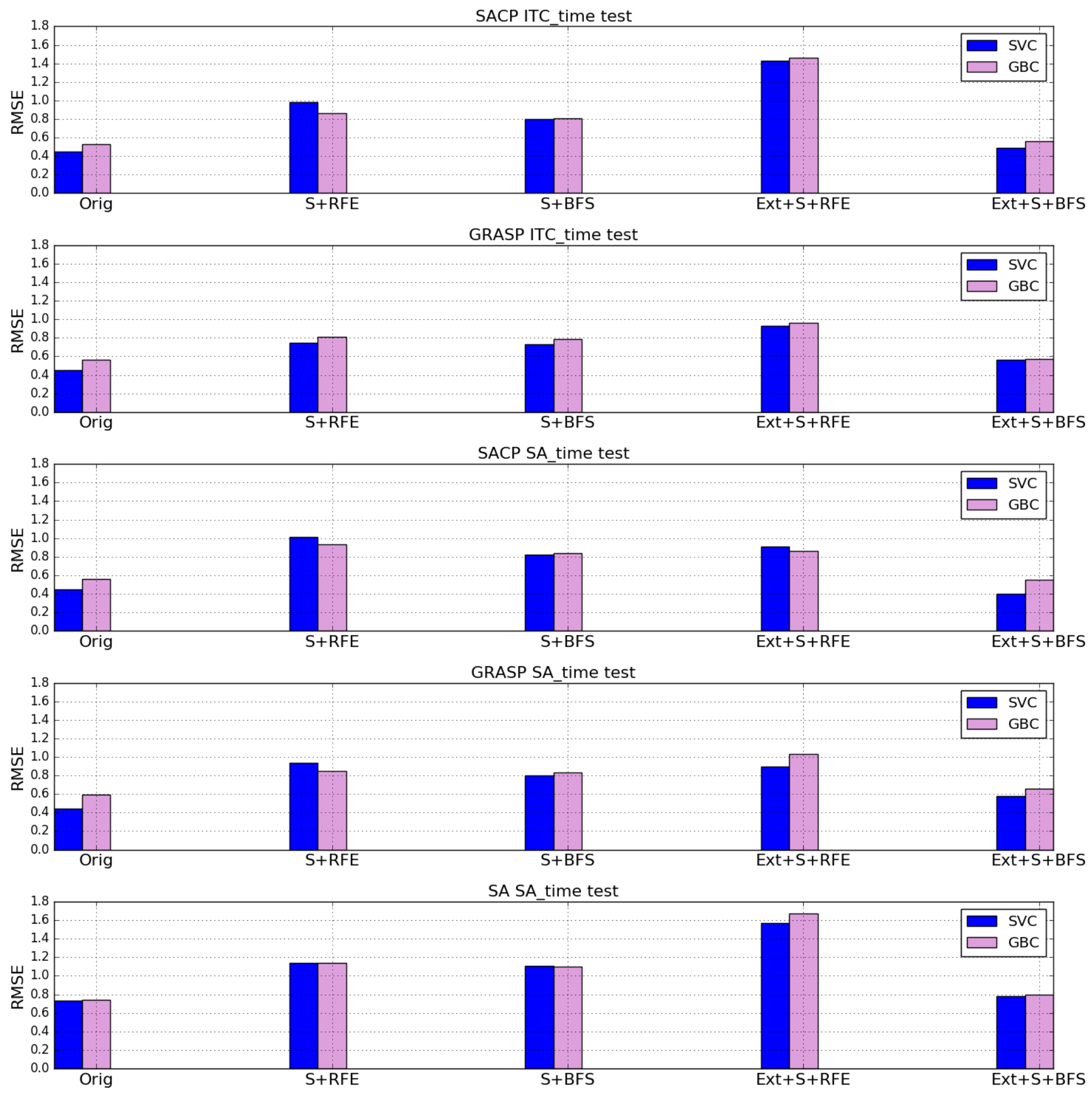

Figure 5.7: Visual comparison of RMSE obtained by different regression models on the test datasets

models can be observed on Figure 7. In addition, quantitative comparison of the RMSE results on the test dataset can be found on Figure 5.7.

In general, even though performance of different estimators obtained on the training dataset by 10-fold cross-validation is better, we still could predict the solver score for the test set rather well. Interestingly, the best results have been obtained on the original feature set, while in some cases the estimator performance for the reduced datasets worsened significantly. Also, SVR performed just slightly better than GBR for most of the datasets.

We may notice that BFS technique was able to determine the most important features for the test datasets, while significantly reducing the number of attributes, especially for the extended versions of the datasets. In comparison, RFE performed relatively worse, and for some datasets, especially for the expanded versions of the `SA` and the `SACP_ITC`, was not able to obtain appropriate results. On the other hand, in cross-validation experiments, the performance models based on the features chosen by RFE dominated over the BFS feature-based models. This may be the case because the RFE technique is based on the application of the specific estimator, RF in our case, therefore, we could easily overfit for certain problem subclasses, while missing some important characteristics needed for the performance prediction. This problem could be potentially solved by choosing another estimator and its proper configuration for the RFE procedure, however, this question is left for future investigation.

Another observation is that the constructed feature set characterized the performance of the SACP and the GRASP solvers better for both versions of the datasets, while performance models for the SA solver obtain more outliers. The same results have been noticed for the training dataset where we were able to construct better performance models using cross-validation for the SACP and the GRASP solvers in comparison with the SA solver.

As a result, we may conclude that during our experiments we were able to construct rather good performance prediction models in cross-validation settings for the different solvers presented in the datasets using the introduced feature set. Moreover, we were able further reduce the datasets, while preserving the performance of the prediction models. Additionally, we confirmed the performance of the estimators by testing them in the real-world settings using the test dataset of unseen instances.

5.3 Algorithm Selection for the ETP

In this subchapter, we describe the experiments conducted for the Algorithm Selection (AS) Problem using various classification algorithms and preprocessing methods. At first, we carry out the experiments on the training datasets using 10-fold cross-validation. For this purpose, all classifiers with the optimal parameter configurations have been tested on a complete feature set where the datasets have been presented in the original, the standardized and the discretized forms. Further, we reduced the datasets by application of BFS and RFE on the original and the extended versions of the datasets. As performance of the estimators is usually better on the preprocessed datasets in comparison with the original ones, we applied the attribute selection methods on the standardized and the discretized versions only. Afterwards, we tested the best performing classification techniques with the optimal parameter settings on the reduced datasets.

Additionally, we checked the performance of the estimators on the real-world settings where the test dataset of unseen instances has been used. In addition to the initial feature set, we tested various classification algorithms on the reduced versions of the training and test datasets where the feature subsets have been obtained by the RFE

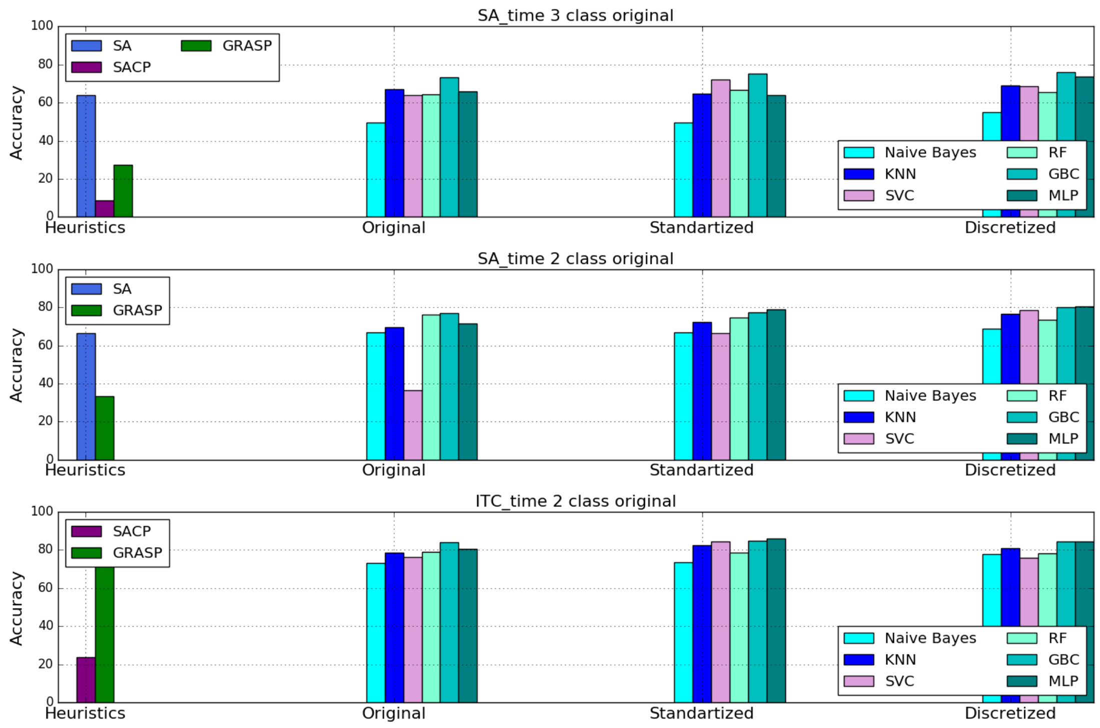

Figure 5.8: Visual comparison of the accuracy obtained by the classification algorithms on the original training datasets

and the BFS techniques on the previous step. The estimators have been examined with the best parameters settings obtained on the cross-validation step. All datasets have been presented in the standardized and the discretized versions, where the preprocessing methods have been trained on the training dataset only, and then afterwards, applied to the models on the test dataset.

The detailed results obtained during the experiments can be found in section B.2. Quantitative comparison of the accuracy obtained on the original and the reduced training datasets by different methods in comparison with the underlying heuristics can be seen on Figure 5.8 and Figure 5.9 respectively. The parameter configurations used in the experiments can be found in section C.2. The features chosen by the variable selection methods for each dataset can be found in section A.2.

5.3.1 The Algorithm Selection Problem For The ETP On The Training dataset

First, let's starts with the performance analysis of various classification methods on the initial feature set. As we can see, we were able to achieve 13%, 15% and 11% accuracy improvements for the `SA_3_class`, `SA_2_class` and `ITC_2_class` respectively in

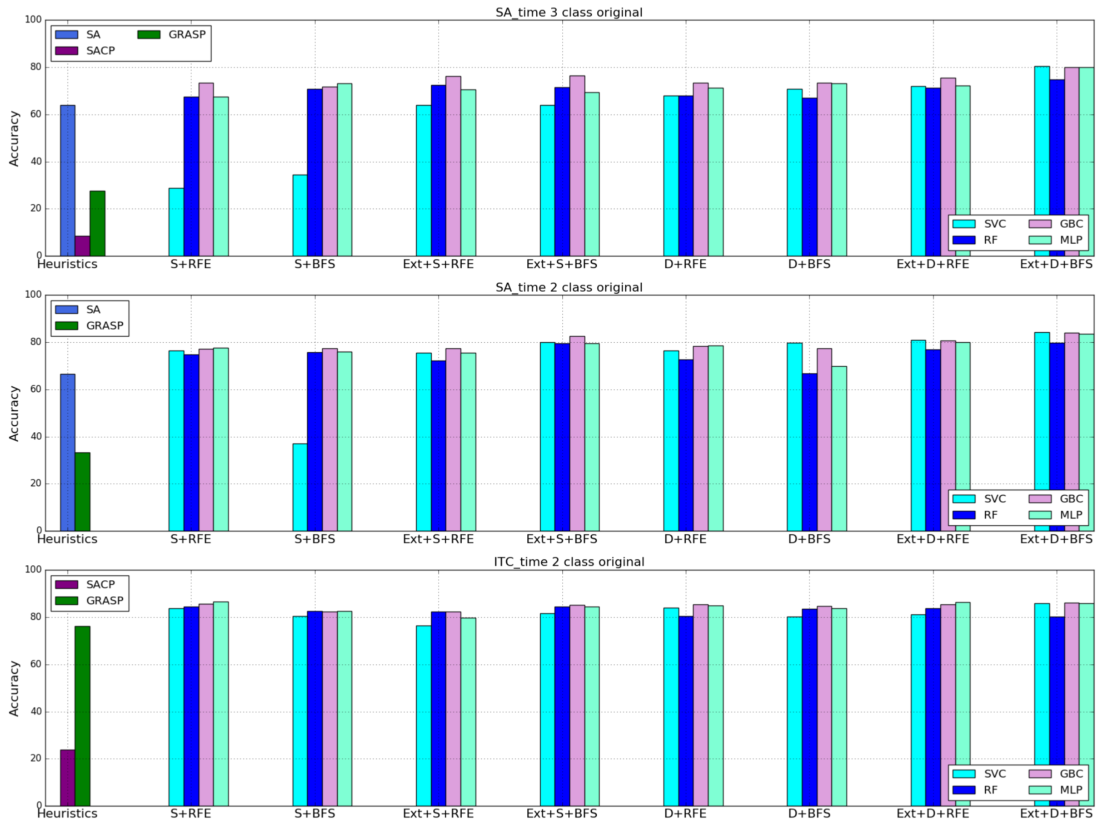

Figure 5.9: Visual comparison of the accuracy obtained by the classification algorithms on the reduced training datasets

comparison with the best performing heuristics for the ETP. In most cases, the best results have been obtained by either GBC or MLP classifiers, however, SVC with the proper parameter configuration was able to achieve analogous results on the preprocessed datasets. Interestingly, Naïve Bayes and KNN have shown rather worse performance even on the standardized and the discretized datasets compared to the other classification algorithms, and therefore, they were excluded from further investigation. It is also worth mentioning that preprocessing, especially discretization, improved the performance of almost each every estimator, especially in the case of SVC and KNN. On the reduced versions of the datasets, we were able further improve the accuracy of the AS-based solver for the `SA_time` datasets on 18% achieving 81% and 84% accuracy respectively. However, we did not obtain any additional improvements on the `ITC_time` dataset. One assumption is that this might be the case because it is harder for the estimator to differentiate between the SACP solver and the others using the current feature set. However, we significantly increased the ability of the classification methods to identify the instances where the GRASP outperformed the SA solver.

Although, in general, almost all methods have shown comparable performance, similar to the original datasets, in most cases the best accuracy has been achieved by GBC and MLP. Interestingly, the best performance models for all datasets have been obtained on the expanded discretized versions of the datasets reduced by BFS. Besides, for most datasets we were able to achieve at least 10-12% improvement in comparison with the best performing algorithm.

Surprisingly, although the number of features selected by the RFE procedure is significantly greater than the number of attributes selected by BFS, we observed that primarily, the classification algorithms performed better on the datasets reduced by BFS, especially on the extended versions of the datasets. Similarly to the Performance Prediction problem, the RFE procedure may require additional parameter configuration, however, it is left for further investigation.

Another interesting observation is that the models constructed by various estimators are better on the extended datasets than on the original ones. One explanation may be that as the number of features increases drastically, and therefore, we may find more appropriate attribute subsets for our task and model new dependencies that were missing in the original datasets. Also, it is worth mentioning that even if, in general, the preprocessing methods improved the estimator performance, the classifiers have shown better performance on the discretized datasets compared to the standardized ones.

5.3.2 The Algorithm Selection Problem For The ETP On The Test dataset

In our experiments on the test dataset, we were able to get rather good results, especially on the extended versions of the datasets. The best accuracy improvement for all datasets have been achieved on the extended discretized datasets reduced by BFS, specifically about 17% for the `SA_3class` dataset, 18% for the `SA_2class` dataset and 16% for the `ITC_2class` dataset in comparison with the best performing heuristics. Moreover, in most cases the estimators were able to achieve at least 8-10% improvement. Another observation is that we also got the results close to the best on the discretized datasets with the original feature set and on the extended standardized datasets reduced by BFS.

In general, we might notice that the performance of the different models in our investigation was better on the discretized datasets than on the standardized ones. Also, based on the results, the BFS technique was able to characterize the solvers rather well, while RFE was not always successful, and as mentioned, apparently need to be properly configured. Similarly, we were able to construct better classification models on the extended version of the datasets, while the preprocessing and variable selection processes may be rather time consuming. Mainly, GBC and MLP showed the best performance, while SVC and RF performed comparably worse, however, in some cases SVC with the proper configuration was also able to get good results. Interestingly, the described results and observations are fairly similar to those that we have been already obtained on the training datasets using cross-validation.

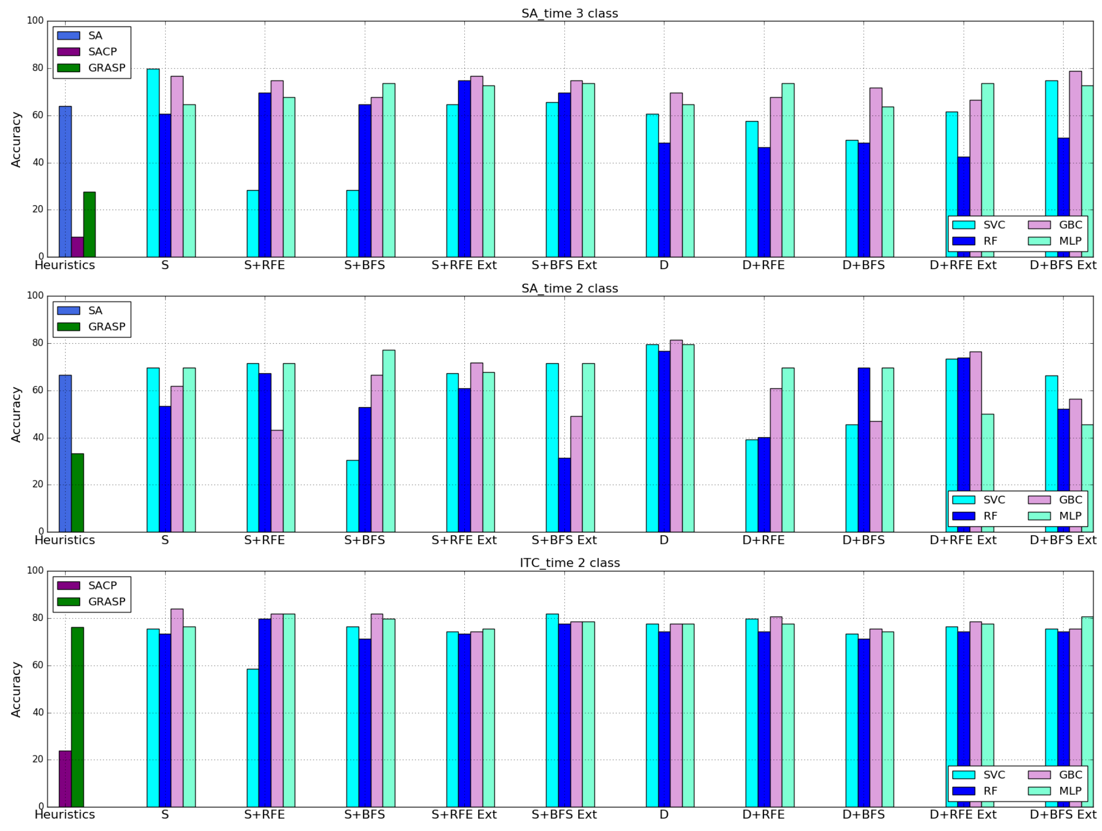

Figure 5.10: Visual comparison of the accuracy obtained by the classification algorithms on the test datasets

Based on the experiment's results on the training and the test datasets we may conclude that we were able to construct the AS-based solver that outperformed the underlying heuristics for the ETP for most cases, especially for the `SA_time` dataset. In detail, the solver based on the AS technique surpassed the best heuristics for the ETP up to 18% on `SA_time` and up to 11% on `ITC_time` training dataset. Moreover, almost all constructed AS models outperformed the algorithms developed to the ETP by at least 10-12% on the training datasets. For the test datasets, we achieved 17%, 18% and 16% accuracy improvement for the `SA_3class`, `SA_2class` and `ITC_2class` respectively.

However, the models might be further improved by introducing new features, finding more appropriate feature selection techniques and finding new real-word based instances. Nonetheless, all these questions are left for further investigation.

5.4 Optimization of parameter settings

In this subsection, we discuss hyperparameter optimization for the estimators, as performance of the model strongly depends on the parameters used. Some methods have a lot of parameters to configure, and therefore, we will choose the parameters which are considered to be the most influential for the outcome. For each estimator a range of parameters is defined separately. Also, the estimators that can be used for both classification and regression problems will be examined together as the working principle of the algorithm and most of the estimator-specific parameters remain the same. Additionally, we illustrate the comparison of some parameter configurations by the graphics where it is considered to be useful. However, in most cases we display the results for the one dataset only, as, in general, the results obtained on the different datasets are quite similar to each other.

For parameter optimization, Scikit-learn provides several built-in methods, such as the Exhaustive Grid Search (EGS) and the Randomized Parameter Optimization (RPO) which simplify hyperparameter optimization process. In EGS you need to specify a grid of parameters that will be tested and then, using the specified estimator, the best combination of the parameters will be returned. In contrast, RPO randomly searches over the provided parameters drawn from the distribution over the parameter values. While the Grid Search approach looks more intuitive, RPO shows good results if the number of iterations is sufficient [BB12]. Moreover, for the methods with a lot of parameters, exhaustive enumeration will be fairly time-consuming.

Usually, in EGS and RPO the performance metrics for evaluation of the constructed models are taken from the estimator tested, however, they may be specified separately, as, for example, accuracy in case of imbalanced datasets can be uninformative. Moreover, it is permitted to use multiple metrics or implement your own metric. Also, most binary metrics can be averaged for the multi-class problems with several options for different scenarios, such as unequal class importance or imbalance learning.

Also, there are some model-specific methods, for example, the Coordinate Decent Method for Lasso and E-net regressions or the Leave One Out implementation for Ridge regression. Therefore, as these approaches specifically optimized for a certain estimator, we used them for finding the best parameters for regularization techniques.

For the other estimators in our experiments, we used the Grid-Search method, specifically for K-Nearest Neighbors and Support Vector Machine. Due to the model complexity and the large number of configuration parameters, for Random Forest, Gradient Boosting Machine and Multi-Layer-Perceptron the RPO method has been employed with 90 as the number of iterations, which is considered to be sufficient for achieving appropriate results. As a measure of estimator performance, for regression techniques `RMSE` has been employed, while for the classification case `ROC_AUC` and `Log_loss` have been used for the binary and the multiclass problems respectively. The best parameters found for the estimators and corresponding parameter range tested can be observed in section C.1 and section C.2.

Additionally, we experimented with several preprocessing techniques represented in Scikit-Learn, namely Standard Scaler, MinMax Scaler, Robust Scaler, Quantile Transformer and Normilizer. Also, we explored several discretization methods, namely the Equal-Frequency, Equal-Width methods with the different number of bins represented in Orange library and the MDL discretization with the Mutual Information and the Kononenko criteria from Weka.

5.4.1 Linear Regression based Techniques

First, we start with analysis of the algorithms which are based on Linear Regression and are mainly used for regression problems. These approaches are Lasso (L_1), Ridge (L_2) regularizations and Elastic Net that compromises between Lasso and Ridge methods. The main principle of these methods is to penalize regression coefficients in order to prevent overfitting.

For Lasso and Ridge regressions we need to configure one parameter α that corresponds to the penalty coefficient of the regularization term. Additionally, in Elastic Net, we need to set up the parameter corresponding to the ratio between the usage of the L_1 and the L_2 penalty terms.

As mentioned, Scikit-learn also has several built-in methods that help to find the optimal parameters for regularization techniques. As a result, for Lasso and E-net parameter search we used the Coordinate Descent (CD) method that is based on approximate error minimization. However, in order to obtain appropriate results, we need to specify the ϵ parameter, which defines the length of regularization path and roughly corresponds to the number of parameters tested. For Ridge regression, we tested a technique which is based on the efficient Leave One Out implementation of cross validation. The example of RMSE obtained for the different parameter configurations of various approaches and preprocessing methods for the `ITC2007_time` dataset can be found on Figure 5.11.

Based on the quantitative results, we observed that for our experiments with Lasso regression the best α values were small, while for Ridge they were relatively greater, especially in the case of the standardized datasets. However, after comparison the parameters for Ridge regression visually, we discovered that in most cases RMSE just slightly fluctuated around one value, and therefore, good results for Ridge regression may be also obtained with the smaller α values. On the other hand, the performance of Lasso regression hardly depended on the α parameter and RMSE grown together with the α value. Another important observation was that in most cases Elastic Net showed the best performance when it mostly used the L_2 penalization together with the small penalty parameter. Moreover, when we considered the impact of both regularization terms equally (`l_1`=0.5), while the α was increasing, RMSE also started to grow due to the impact of the L_1 term. However, the growth was still slower than in the case of Lasso due to the presence of the L_2 component.

Regarding different preprocessing techniques, we may notice that, in general, for Ridge regression the results did not depend on the choice of the method, while for Lasso and

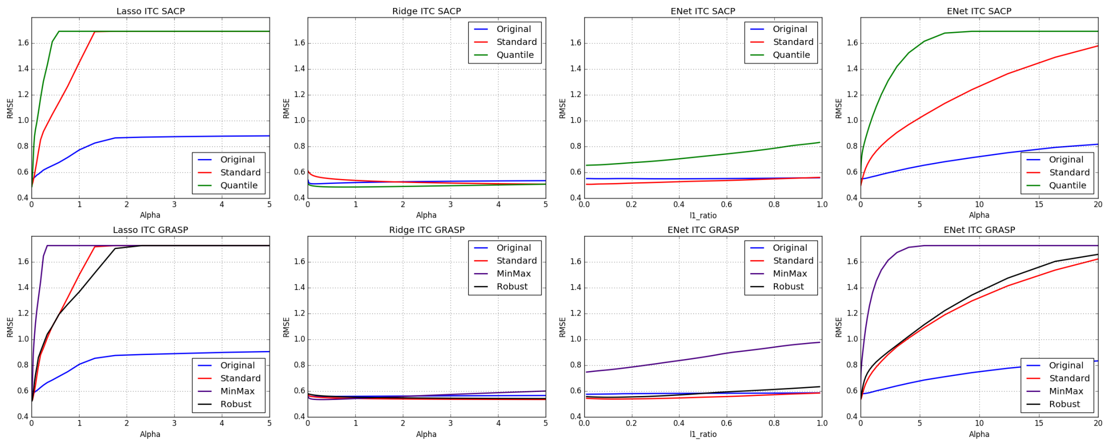

Figure 5.11: RMSE obtained by different regularization techniques with various parameter settings. For ENet $L_1 = 0.5$, α=0.05

ENet it becomes essential. Based on the visual comparison we concluded that the best results were obtained by the usage of Standard and Robust scalers for both the Lasso and the Elastic Net models.

Based on these facts, we can assume that Ridge regression showed better performance on the datasets than Lasso. This can be the case because Lasso tends to eliminate certain features completely by assigning them zero coefficients, while missing some important relations. On the other hand, although Ridge regression minimizes the impact of non-informative features, it still considers them in the model. Another note is that, regardless of simplicity, after proper configuration, all regularization methods were able to construct the accurate predictive models for all solvers that are comparable with more sophisticated methods, especially Elastic Net technique.

5.4.2 K-Nearest Neighbors

K-Nearest-Neighbor (KNN) is another State-Of-The-Art technique that is widely used in regression and classification problems, and moreover, is considered one of the most influential Data Science algorithms [WKRQ$^+$07]. The main idea of this method is to predict the label of a given instance based on k closest training samples for the instance. Thus, as we need to compute the distances between all pairs of the samples, the application of the brute-force implementation may be rather time consuming. Therefore, for our experiments in order to address this drawback we chose time-efficient Ball-tree implementation [Omo89] represented in Scikit-learn that encodes the aggregate distance information and stores it into a tree-based data structure.

There are three main parameters that need to be configured for the supervised version of KNN: **the number of neighbors** that corresponds to the number of the closest samples used for the label prediction, **distance measure** and **weight function**. As

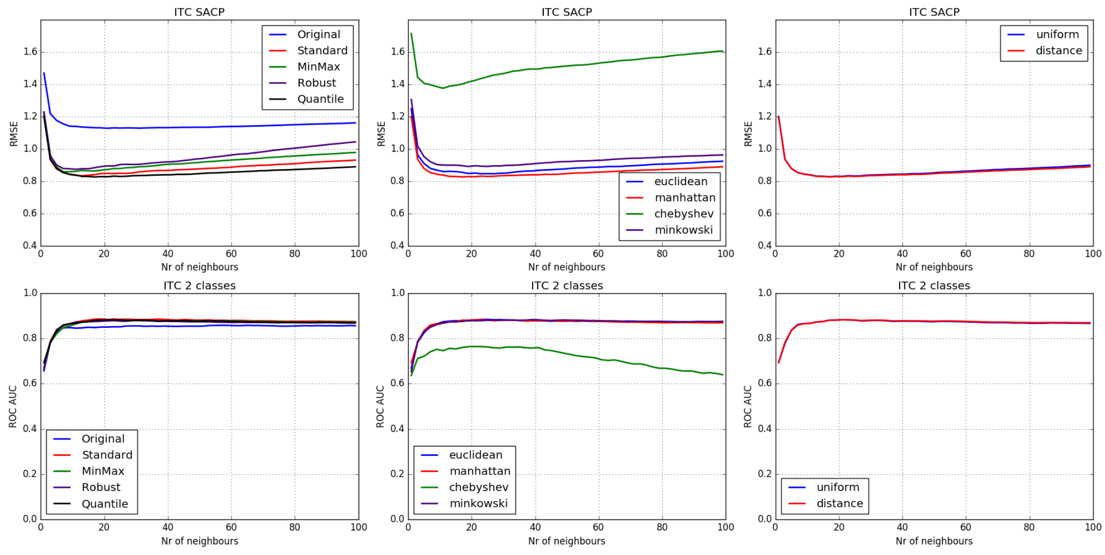

Figure 5.12: The impact of different parameter settings of the KNN algorithm on prediction quality

distance metric we tested several measures represented in Scikit-Learn, namely Euclidian, Manhattan, Chebyshev and Minkowski distances. Weight function characterizes the influence of different neighbors on prediction results: the "distance" function assigns weights corresponding to the inverse of their distance, therefore, closer samples will get greater impact, and the "uniform" function weights all samples equally. Also, as KNN is a distance-based algorithm, it is essential to scale all attributes in the same range, and therefore, data preprocessing is required. The impact of different parameters on performance of the model can be seen on Figure 5.12.

For regression, the optimal `n_neighbours` for all solvers lies between 10 and 20 for both the standardized and the original versions of the datasets. However, for the binary classification the preprocessing significantly decreased the number of neighbors needed for the prediction from 50 to 20-25, although for the multiclass case `n_neighbours` increased from 50 to 75 for the standardized and the discretized versions of the datasets. Therefore, we can conclude that the optimal value of `n_neighbours` is extremely data dependent. However, after a certain point larger `n_neighbours` helps to suppress the noise, therefore, the error starts to grow smoother.

Regarding the distance metrics, we observed that the Manhattan distance showed the best performance, while the Chebyshev metric is the most unsuitable for our task. Although the Manhattan measure has been frequently chosen as the most appropriate one, visually it is recognizable that the results based on the Minkowski and the Euclidian measures are relatively similar. Similar observation was for the weight function: KNN with the "uniform" and the "distance" weights functions obtained approximately identical performance. Also, the estimator constructed on the preprocessed versions of the datasets, in general, outperforms the algorithm which is based on the original version.

5.4.3 Support Vector Machines

SVM [CV95]is another popular method that is commonly employed in classification, regression and outlier detection tasks. The main idea of SVM is a construction of separation hyperplanes while maximizing the margin. Moreover, there is a kernel trick where the data can be transformed into higher dimensional space, and as a result, SVM can also model non-linear dependencies.

In Scikit-learn there are implementations of SVM for classification and regression tasks with the Linear, the Radial Basis Function (RBF), the Sigmoid and the Polynomial kernels. Parameter configuration for SVM hardly depends of the kernel transformation used, because in addition to general SVM parameters we need to configure free parameters of the kernel function. Also, SVM requires data preprocessing as the constructed model depends on the vector product calculation, therefore, the attributes with higher spread range will get more influence on the results. Additionally, different feature scales can also lead to numerical difficulties in calculation.

In our work, we only experimented with the Radial Basis Function (RBF) kernel. RBF is the most common choice due to the possibility for modeling non-linear relations and is less computationally expensive than, for example, the Polynomial kernel [HCL10]. Also, it has been observed that the Sigmoid kernel behaves like RBF with some parameters [CV95], and the Linear kernel is a special case of RBF [KL03]. However, we also experimented with several configurations of the Polynomial Kernel, but after several hours of computational time we stopped it manually and considered it unsuitable for our problem.

The common parameter for all SVM kernels is the parameter `C` that corresponds to the penalty of an error term. This parameter characterizes the tradeoff between the training error and simplicity of the decision surface. If C is too big, then the model overfits, and computational time increases also as the model needs more support vectors. However, if C is too small, it leads to underfitting, therefore, we need to find the parameter that keep the training error small while provides good generalization. Also, SVM supports the cost-sensitive classification for unbalanced cases where we can define an additional multiplier of the error term for a given class. For regression task, we also need to configure the parameter ϵ that is responsible for the margin of tolerance where the errors do not receive any penalty. Also, we need to configure the RBF-specific parameter γ, that is related to the spread of influence between different vectors. Visual comparison of SVM with the different parameters can be found on Figure 5.13 and Figure 5.14.

In general, we observed that the SVM results hardly depended on the parameter configuration and the preprocessing methods used. Moreover, properly configured SVM can outperform the other classification techniques for the ETP. This dependency is clearly visible on the heatmaps which illustrate the relations between the γ and the C values for classification and regression. The best parameter settings lie somewhere in the diagonal,

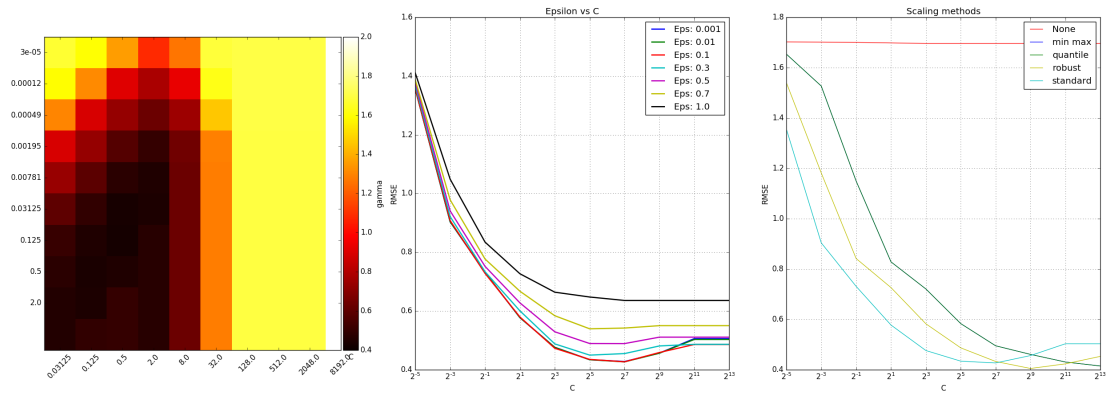

Figure 5.13: The impact of different parameter settings of the SVM regressor on prediction quality

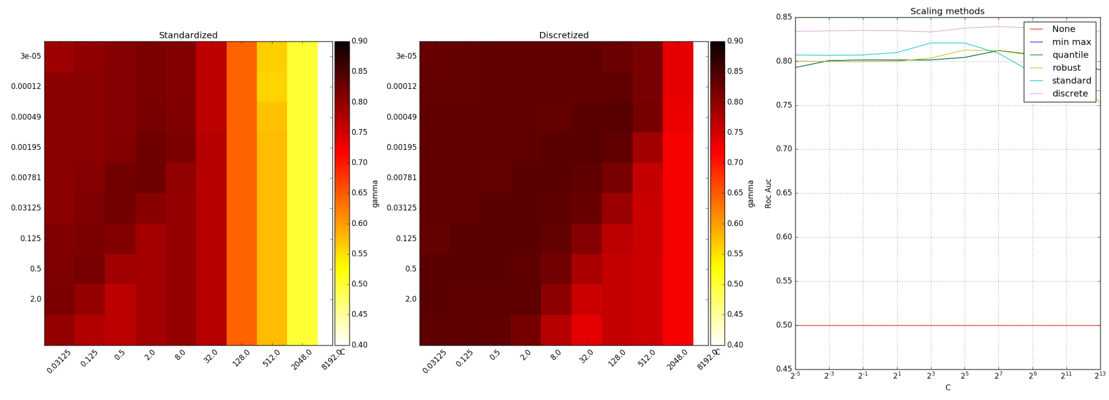

Figure 5.14: The impact of different parameter settings of the SVM classifier on prediction quality

where the small γ values, and therefore, the larger spread of influence of the support vectors may be compensated by the larger number of support vectors (larger C). However, for extremely large γ values it is nearly impossible to find such a C value that will be able to prevent overfitting, and when the γ values are too small, the model will be too constrained, and therefore, be unable to provide appropriate results. The best values for the margin of tolerance (or ϵ) for the regression methods in most cases is below 0.1, because otherwise RMSE started to increase. This observation may be examined based on the parameter configuration results and the graphics.

Regarding the preprocessing techniques, for the regression case we may notice that Standard Scaler may be considered the most suitable for our task, while for classification in most cases all preprocessing techniques provided similar results. However, it is clearly visible that the performance of SVM on the discretized datasets is better compared with the other techniques.

5.4.4 Tree-based approaches

Recently, Decision Trees (DT) and various Tree-Based approaches became popular due to its simplicity, high performance and relatively simple parameter tuning. However, as a single Decision Tree tends to have problems with overfitting, the most popular methods from this family are the ensemble methods such as Random Forest (RF), Extra Tree and others. Moreover, in Scikit-Learn there are boosting versions of the tree-based approaches, for example, Gradient Boosting Machine (GBM) where the estimators are sequentially built on each other.

In this thesis, we experimented with RF and GBM as they belong to the different families of the ensemble approaches, and usually perform better than a single DT. RF is an approach that is based on construction of multiple DT based on the subset of the features, and therefore, reduce the tendency to overfit. Also, RF was noted as one of the best performing models for the Runtime Prediction Problem employed for the SAT problem [HXHLB14].

Regardless of diversity, RF and GBM have a lot of parameters in common which correspond to the decision tree structure. One of the main parameters to configure is the **maximum number of features** used to create a single tree. This parameter must be chosen carefully as too small parameter may increase bias in a single DT, especially in the case of noisy variables. However, too high a parameter may increase the correlation between different trees, and therefore, we will lose the main advantage of RF. Additionally, we need to tune the depth of a single DT, for example, by the **minimum sample leaf size** that corresponds to the minimum number of samples required in a leaf node. Another measure to prevent overfitting is the **minimum number of samples** expected for the node to be further expanded. Finally, we need to specify the ensemble parameter, namely the **number of trees** that we want to create. This parameter depends on available computational resources, and also, after a certain point increasing the number of estimators will not increase the performance of the estimator. Additionally, we can choose the splitting criteria, for example, the gini or the entropy criteria for classification or MSE or MAE for the regression problems.

Although GBM and RF have a lot of common parameters, the idea behind these methods differs. While RF constructs the number of fully-grown trees in parallel, GBM builds a set of weak learners iteratively, taking into account the samples that have not been predicted correctly in the previous iterations. After the algorithm built the specified number of estimators, the complex predictor will be constructed. Therefore, GBM has an additional parameter, namely the **learning rate**, which defines the estimator influence on the final outcome. Therefore, in order to obtain appropriate results for the low values of the learning rate we need more trees. Additionally, we can choose a **subsample size** that corresponds to the fraction of observations to be randomly selected for each learner.

Let's start with analysis of the best parameter configurations for Random Forest. For regression problems, in most cases `max_features` vary between 0.7-0.8 and `min_sample_split` differs between 80-100. Therefore, these values are large enough in order to get appropriate results, while still providing tree diversity and preventing overfitting. For classification, these attributes are more dispersed, for example, while in most cases the `max_features` values lie between 0.6-0.8, there are some cases where the fraction of features is relatively low, about 0.2-0.3. The average number of `min_sample_split` is also lower compared to the regression settings, and is around 70-80. This is because our classification tasks are consists of the imbalanced classes, and therefore, for some instance groups we need to create more specific subtrees. Interestingly, for regression, the number of estimators is fluctuating around 100, however, the classification tasks require the larger number of decision trees, therefore the optimal values are about 130-140 trees. For all problems `min_sample_leaf` values are analogous and around 30.

Surprisingly, in most cases for GBM we got rather similar parameter settings regarding the decision tree structure in comparison with RF. For regression,the `max_features` values are larger compared to the classification task, and are about 0.8. The optimum values for the maximum fraction of features for classification vary 0.7-0.8 for the `SA_time` datasets and are about 0.6 for the `ITC_time` dataset. For regression, the values for the `min_sample_split` feature are fairly analogous, and changes between 150-200, while for the classification tasks they are more scattered from the dataset to the dataset and dispersed between 90 and 200. For classification, the optimal `learning_rate` parameter is rather small, about 0.02-0.05, and therefore, required a large number of estimators, about 180-200. However, for regression, the learning rate values are larger, about 0.1-0.2, and accordingly, the optimal number of trees is smaller, about 150. Further, the optimal values for the subsample size and the leaf size parameters are quite similar for both problems, and are 0.8-0.9 and 30 respectively.

Another interesting observation is that in most cases the tree-based algorithms are invariant to the scaling techniques,and therefore, they provide similar results regardless of the preprocessing method used. However, discretization improves the performance of Gradient Boosting and allows it to better differentiate between different classes. Nevertheless, RF also improved its performance after the discretization procedure, although just slightly.

5.4.5 Multi-Layer Perception

Another approach that is able to model non-linear relationships is Multi-Layer Perception (MLP). However, it requires rigorous tuning, because it has too many parameters to set. Besides, some parameters depend on each other, and therefore, every time we need to check the full network configuration. However, for some cases the methods were developed which simplify the process. For example, we can adaptively decide when the learning process must stop by defining the number of non-improving iterations after which the process must be interrupted. Another possibility is to vary the learning rate instead of using the constant one, for example, employing the bigger values in the beginning of the learning process, and the smaller values in the end.

Nonetheless, still plenty of uncertainty is left. For example, two of the most important parameters are the number of hidden layers and the number of neurons in them. However, the large number of neurons lead to a lot of free parameters, and therefore, MLP tends to overfit and requires plenty of computational resources. As a solution for the overfitting problem, in Sklearn L_2 regularization has been implemented. However, the appropriate regularization parameter α must be set. In addition, we need to choose other parameters such as activation function, weight optimization algorithm and in some cases define the algorithm specific parameters. Another MLP problem is the weight initialization of the neurons, because performance mostly depends on how the initial weighs are assigned. Also, it is required to scale the data, as MLP is sensitive to it.

Because of the configuration difficulty, instead of finding the optimal parameters for each dataset, we tried to find some general parameter settings that would provide rather good results for the most datasets. At first, we experimented with the numbers of neurons and the hidden layers separately up to 10 layers and up 1000 neurons in each layer. However, due to the increasing number of neurons, the learning time started to rise significantly, although it did not significantly influence the performance. Therefore, we considered 3 layers with 300 neurons each sufficiently enough for our experiments. The number of non-improving iterations has been set to 2000. Although after initial experiments we set all regression parameters, for classification we used the Grid Search in order to find the appropriate parameters, namely the solver, the learning rate and the activation function. For classification, all parameters settings for MLP can be found in Table C.11.

For regression, we determined that the SGD that corresponds to the Stochastic Gradient Descent solver outperforms the others, and therefore, used it for the regression experiments. Similarly, for classification, the SGD has been chosen as the most appropriate in most cases. For regression, after initial experiments the adaptive learning rate has been employed, while for classification either the constant or the adaptive learning rates have been chosen from one dataset to another. Regarding the last parameter, we left the relu activation function as the best performing method for the regression problem, but for classification the function choice alternates between relu, logistic and tanh almost equally.

Despite limitations, we achieved rather good performance improvement and moreover, MLP outperformed most of the methods in our experiments with the exception of tree-based methods.

CHAPTER 6

Conclusion

In this thesis, we developed an approach for the Algorithm Selection and Performance Prediction Problems for the ITC2007 formulation of the Examination Timetabling Problem based on Machine Learning. For this purpose, we introduced 196 features based on the ETP itself and by reducing the ETP to the other well-known problems. Furthermore, we collected the best performing heuristics (SA, SACP, GRASP) for the ITC2007 formulation of the ETP and run them on the dataset of 2248 instances which consists of real-word and artificially generated instances.

Afterwards, we investigated performance of different Machine Learning techniques for the Algorithm Selection and Performance Prediction problems, namely SVM, KNN, RF, GBR, MLP and also experimented with various estimator settings. In addition, we explored LR, Lasso, Ridge and Naïve Bayes for the regression and the classification tasks respectively. Moreover, the most relevant feature sets have been identified using various feature selection techniques.

Finally, we concluded that SVM, GBR and MLP are the most appropriate ML algorithms for solving the Algorithm Selection and Performance Prediction problems for the ETP. In general, the preprocessing methods, such as scaling and supervised discretization, improved the performance of the estimators. Moreover, the estimators' performance has been further increased by introduction of new composed features and application of the feature selection techniques. As a result, the estimators were able to construct rather accurate models for all solvers and the datasets in cross-validation settings and the real-world-based scenario, using a test set of unseen instances. In detail, for the regression problem the Correlation Coefficient between the true and the predicted solver score exceeded 0.9 and the Algorithm Selection Solver outperformed the best performing heuristics for all datasets.

Potential future work may include introduction of a new feature set and exploring other formulations of the ETP. Additionally, it might be interesting to experiment on new instance distributions including the instances where none of the heuristics can achieve a feasible solution and consider it as a separate class. Moreover, for the Algorithm Selection Problem an algorithm portfolio may be further extended by other heuristics and also, other approaches, such as dynamic algorithm portfolio, could be applied.

Another possibility for further investigation would be employing the performance prediction models for the Algorithm Selection problem or experimenting further on closely related domains, such as, for example, Runtime Prediction. Additionally, the performance prediction models may be used as a part of an instance generator in order to obtain hard instances. This question is especially essential for the ETP as it is rather hard to find good benchmark datasets.

APPENDIX A

Feature Selection Results

In this appendix we will provide detailed information about the variables chosen by the Feature Selection methods for the Algorithm Performance Prediction and Algorithm Selection problems. Additionally, we will observe statistics about the most frequently chosen features by RFE and BFS for the original and the extended versions of the datasets. However, as the extended datasets mainly consist of the compound features, we also investigated the original features that have been used for construction of the variables in the reduced datasets in addition to the original features. For that, after decomposing these features into the original ones, we calculated the occurrence of each original feature in the decomposition of the reduced dataset.

A.1 Feature Selection Results for The Algorithm Performance Prediction Problem

Selected features	**Frequency**
ExamPerStudentSD	30.0%
NumberOfStudents	25.0%
WeightPeriodSpread * TotalRoomCapacity	25.0%
ExamsPerStudentMax	25.0%
NumberOfStudents * NumberOfStudents	25.0%
CliqueCalculationTime	25.0%
ExamPerStudentMean	25.0%
ExamPerStudentSD * TotalExamCap/TotalRoomAndNrPeriods	20.0%
ExamPerStudentVarCoef	20.0%
StudentPerExam3Q	20.0%
StudentPerExamMeanMax	20.0%
TotalExamCap/TotalRoomAndNrPeriods	20.0%
ExamsPerStudentMax + RoomsPerExamSD	20.0%
NumberOfRooms	15.0%
StudentPerExam1Q	15.0%
RoomsPerExam95P	15.0%
ExamPerStudentEntropy	15.0%
NumberOfStudents + NumberOfDays	15.0%
StudentPerExamVarCoef	15.0%
NumberOfStudents * NumberOfPeriods	15.0%
StudentPerExamMax	15.0%
StudentPerExamEntropy	15.0%
NumberOfStudents / NumberOfDays	15.0%
NumberOfStudents - NumberOfRooms	15.0%
StudentPerExamEntropy + CliqueCalculationTime	15.0%
NumberOfStudents + NumberOfPeriods	15.0%
NumberOfStudents / NumberOfAfterConstraints	15.0%
NumberOfDays	15.0%
NumberOfRoomExclusiveConstraints	15.0%
NrOfConflictMax	15.0%

Table A.1: The most selected features among all datasets for the Performance Prediction problem

Selected features	**Frequency**	**Frequency (%)**
NumberOfStudents	132	28.09%
ExamPerStudentSD	14	2.98%
NumberOfDays	13	2.77%
NumberOfRooms	12	2.55%
TotalRoomCapacity	12	2.55%
NumberOfPeriods	11	2.34%
WeightPeriodSpread	10	2.13%
ExamsPerStudentMax	10	2.13%
Slackness	9	1.91%
StudentPerExamEntropy	9	1.91%
TotalExamCap/TotalRoomAndNrPeriods	9	1.91%
NumberOfExamsOverSlack	8	1.7%
StudentPerExamMax	8	1.7%
ExamPerStudentMean	8	1.7%
ExamPerStudentVarCoef	8	1.7%
RoomsPerExam5Per	8	1.7%
CliqueCalculationTime	8	1.7%
NumberOfAfterConstraints	7	1.49%
NumberOfCoincidenceConstraints	7	1.49%
NumberOfExams	6	1.28%
NumberOfExclusionConstraints	6	1.28%
FrontLoadWeight	6	1.28%
StudentPerExam3Q	6	1.28%
ExamPerStudentMedian	5	1.06%
ExamPerStudent1Quant	5	1.06%
ExamPerStudent3Q	5	1.06%
AllRoomExams	5	1.06%
RoomsPerExam95P	5	1.06%
RoomsPerExamSD	5	1.06%
NumberOfRoomExclusiveConstraints	4	0.85%
WeightNumberOfLargeExams	4	0.85%
StudentPerExamMin	4	0.85%
StudentPerExam1Q	4	0.85%
StudentPerExamMeanMax	4	0.85%
RoomsPerExamMean	4	0.85%
RoomsPerExamMed	4	0.85%

Table A.2: The most selected features among the original and the decomposed versions of the extended datasets for the Performance Prediction problem

FS method	Selected features
FFS (74)	NumberOfStudents, NumberOfExams, NumberOfRooms, NumberOfPeriods, NumberOfDays, Slackness, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfExclusionConstraints, NumberOfRoomExclusiveConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, WeightPeriodSpread, StudentPerExamMax, StudentPerExamMean, StudentPerExamMed, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, ExamsPerStudentMax, ExamPerStudent5thPerExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent1Quant, ExamPerStudent3Q, ExamPerStudentEntropy, ExamPerStudentSkew ExamPerStudentMinMax ExamPerStudentMeanMax, NrOfConflictMax, NrOfConflictMean, NrOfConflictVarCoef, NrOfConflictSD, NrOfConflictEntropy, OneRoomOverTotalExams, RoomsPerExam95P, RoomsPerExamMed, OnePeriodExamOverTotal, AllPeriodExam, PeriodPerExam95P, PeriodPerExamMean, PeriodPerExamSD, nrOfConflicts, RatioNrOfConflictsOverExams, RoomCapacityMax, RoomCapacityMed RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, TotalExamCap/TotalRoomAndNrPeriods, ExamPerRoomMax, ExamPerRoomVarCoef, RoomForAllExamsWithoutPenalty, RoomForAllExamsWithoutPenaltyOverTotal, ExamPerPeriodMax, ExamPerPeriodMin, ExamPerPeriodMean, ExamPerPeriodMed, CCMinMax WCCVarCoef, WCCMinMax, WCCMeanMax, CCCalculationTime, CliqueSizeMax, CliqueSizeMin, CliqueSizemean, CliqueSizeVarCoef, CliqueCalculationTime, GreedyScore, ComputationalTime, MaxMembers
RFE (15)	NumberOfDays, TotalNumberOfConstraints, WeightNumberOfLargeExams, StudentPerExamMax,StudentPerExam1Q, StudentPerExamMinMax, ExamPerStudentEntropy, ExamPerStudentSkew, ExamPerStudentMeanMax, NrOfConflictVarCoef, NrOfConflictEntropy, RoomsPerExamMed, OnePeriodExam, AllPeriodExam, PeriodPerExam95P
BFS (15)	NumberOfStudents, NumberOfRooms, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, ExamsPerStudentMax, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, NrOfConflictMax, RoomsPerExam95P, TotalExamCap/TotalRoomAndNrPeriods, CliqueSizeVarCoef, CliqueCalculationTime
F+RFE2 (17)	NumberOfStudents * NumberOfStudents, NumberOfStudents + NumberOfDays, NumberOfStudents + FrontLoadWeight, NumberOfStudents * AllRoomExams, NumberOfStudents + AllRoomExams, NumberOfStudents - RoomsPerExam95P, NumberOfStudents * RoomsPerExam5Per, NumberOfStudents / RoomsPerExam5Per, NumberOfStudents - RoomsPerExam5Per, NumberOfStudents + RoomsPerExam5Per, NumberOfStudents * RoomsPerExamMean,NumberOfStudents / RoomsPerExamMean, NumberOfStudents - RoomsPerExamMean, NumberOfStudents * RoomsPerExamMed, NumberOfStudents / RoomsPerExamMed, NumberOfStudents + RoomsPerExamMed, NumberOfStudents / ExamLengthVarCoef
F+BFS (10)	NumberOfExams * TotalRoomCapacity, WeightPeriodSpread * TotalRoomCapacity, StudentPerExamMed + NrOfConflictMean, StudentPerExamEntropy / RoomForAllExamsWithoutPenalty, StudentPerExamEntropy + CliqueCalculationTime, ExamsPerStudentMax + RoomsPerExamSD, ExamPerStudentMean / PeriodPerExamMean, ExamPerStudentMean - ExamLengthMin,ExamPerStudentSD * TotalExamCap / TotalRoomAndNrPeriods, nrOfConflicts / WCCSD

Table A.3: The features selected by different methods for the SACP solver from the original and the extended versions of the `ITC2007_time datasets`

FS method	**Selected features**
FFS (71)	NumberOfStudents, NumberOfExams, NumberOfRooms, NumberOfPeriods, NumberOfDays, Slackness, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfExclusionConstraints, NumberOfRoomExclusiveConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, WeightPeriodSpread, StudentPerExamMax, StudentPerExamMean, StudentPerExamMed, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, StudentPerExamMinMax, StudentPerExamMeanMax, ExamPerStudent5thPer, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent1Quant, ExamPerStudent3Q, ExamPerStudentEntropy, ExamPerStudentSkew, ExamPerStudentMinMax, ExamPerStudentMeanMax, NrOfConflictMax, NrOfConflictMean, NrOfConflictSD, NrOfConflict1Q, NrOfConflict3Q, RoomsPerExam95P, RoomsPerExamSD, PeriodPerExam95P, PeriodPerExamMean, RatioNrOfConflictsOverExams, RoomCapacityMax,RoomCapacityMin, RoomCapacityMed,RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, TotalExamCap/TotalRoomAndNrPeriods, RoomForAllExamsWithoutPenalty, RoomForAllExamsWithoutPenaltyOverTotal, PeriodLengthMean, CC5th, CCmean, CCMinMax, CCMeanMax, WCCVarCoef, WCCMinMax, WCCMeanMax, CCCalculationTime, CliqueSizeMax, CliqueSizeVarCoef, CliqueSizethird, CliqueCalculationTime, Greedy Score, Greedy Independent Set - Min,Greedy Independent Set MinMax, Modularity, MaxMembers, MeanMembers
RFE (17)	NumberOfDays, TotalNumberOfConstraints, WeightNumberOfLargeExams, StudentPerExamMin, StudentPerExamVarCoef, StudentPerExam3Q, StudentPerExamEntropy, StudentPerExamMeanMax, ExamPerStudentSD, ExamPerStudent3Q, ExamPerStudentEntropy, NrOfConflict5thP, NrOfConflictMed, AllRoomExams, AllRoomOverTotalExams, RoomsPerExam5Per, RoomsPerExamMean
BFS (11)	NumberOfStudents, StudentPerExamMax, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentSD, NrOfConflictMax, RoomsPerExamSD, TotalRoomCapacity, TotalExamCap / TotalRoomAndNrPeriods, WCCMinMax, CliqueCalculationTime
F+RFE2 (16)	NumberOfStudents * NumberOfStudents, NumberOfStudents + NumberOfPeriods, NumberOfStudents + ExamPerStudentMedian, NumberOfStudents * ExamPerStudentVarCoef, NumberOfStudents * ExamPerStudentSD, NumberOfStudents / ExamPerStudentSD, NumberOfStudents - ExamPerStudentSD, NumberOfStudents + ExamPerStudentSD, NumberOfStudents * ExamPerStudent1Quant, NumberOfStudents / ExamPerStudent1Quant, NumberOfStudents * ExamPerStudent3Q, NumberOfStudents - ExamPerStudent3Q, NumberOfStudents + ExamPerStudent3Q, NumberOfStudents - NrOfConflictMeanMax, NumberOfStudents - RoomsPerExam5Per, NumberOfStudents + RoomsPerExam5Per
F+BFS (8)	NumberOfRooms * CliqueSizeThird, WeightPeriodSpread * TotalRoomCapacity, StudentPerExam1Q + NrOfConflict1Q, ExamsPerStudentMax + RoomsPerExamSD, ExamPerStudentMedian / PeriodPerExamMed, ExamPerStudentVarCoef + OnePeriodExam, ExamPerStudentSD * TotalExamCap/TotalRoomAndNrPeriods, TotalRoomCapacity / WCCVarCoef

Table A.4: The features selected by different methods for the GRASP solver from the original and the extended versions of the `ITC2007_time` datasets

FS method	Selected features
FFS (52)	NumberOfStudents, NumberOfExams, NumberOfRooms, NumberOfPeriods, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfExclusionConstraints, NumberOfRoomExclusiveConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, TimeRelatedFractionOfConstraints, WeightPeriodSpread, StudentPerExamMax, StudentPerExamMean, StudentPerExamMed, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, ExamsPerStudentMax, ExamPerStudent5thPer, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent1Quant, ExamPerStudent3Q, ExamPerStudentEntropy, ExamPerStudentSkew, ExamPerStudentMinMax, ExamPerStudentMeanMax, NrOfConflictMean, NrOfConflictSD, PeriodPerExamMean, nrOfConflicts, RatioNrOfConflictsOverExams, RoomCapacityMax, RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, TotalExamCap/TotalRoomAndNrPeriods, ExamPerPeriodMax, ExamPerPeriodMean, AvNrOfPeriodPerDay, CCmean, CCMinMax, WCCVarCoef, WCCMeanMax, CCCalculationTime, CliqueSizeVarCoef, CliqueCalculationTime, Greedy Independent Set - Max
RFE (9)	NumberOfPeriods, NumberOfTimeRelatedConstraints, WeightMixedDuration, FrontLoadWeight, StudentPerExamSD, StudentPerExamMeanMax, ExamPerStudentVarCoef, NrOfConflictMean, NrOfConflictVarCoef
BFS (10)	NumberOfStudents, NumberOfRooms, NumberOfRoomExclusiveConstraints, StudentPerExam3Q, ExamsPerStudentMax, ExamPerStudentMean, ExamPerStudentVarCoef, ExamPerStudentSD, TotalExamCap/TotalRoomAndNrPeriods, CliqueCalculationTime
F+RFE2 (38)	NumberOfStudents ⋆ NumberOfStudents, NumberOfStudents / NumberOfStudents, NumberOfStudents - NumberOfStudents, NumberOfStudents + NumberOfStudents, NumberOfStudents ⋆ NumberOfExams, NumberOfStudents / NumberOfExams, NumberOfStudents - NumberOfExams, NumberOfStudents + NumberOfExams, NumberOfStudents ⋆ NumberOfRooms, NumberOfStudents / NumberOfRooms, NumberOfStudents - NumberOfRooms, NumberOfStudents + NumberOfRooms, NumberOfStudents ⋆ NumberOfPeriods, NumberOfStudents / NumberOfPeriods, NumberOfStudents - NumberOfPeriods, NumberOfStudents + NumberOfPeriods, NumberOfStudents ⋆ NumberOfDays, NumberOfStudents / NumberOfDays, NumberOfStudents - NumberOfDays, NumberOfStudents + NumberOfDays, NumberOfStudents ⋆ Slackness, NumberOfStudents / Slackness,NumberOfStudents - Slackness, NumberOfStudents + Slackness, NumberOfStudents ⋆ NumberOfExamsOverSlack, NumberOfStudents / NumberOfExamsOverSlack, NumberOfStudents - NumberOfExamsOverSlack, NumberOfStudents + NumberOfExamsOverSlack, NumberOfStudents ⋆ NumberOfAfterConstraints, NumberOfStudents / NumberOfAfterConstraints, NumberOfStudents - NumberOfAfterConstraints, NumberOfStudents + NumberOfAfterConstraints, NumberOfStudents ⋆ NumberOfCoincidenceConstraints, NumberOfStudents / NumberOfCoincidenceConstraints, NumberOfStudents - NumberOfCoincidenceConstraints, NumberOfStudents + NumberOfCoincidenceConstraints, NumberOfStudents ⋆ NumberOfExclusionConstraints, NumberOfStudents / NumberOfExclusionConstraints
F+BFS (12)	WeightPeriodSpread ⋆ StudentPerExam3Q, WeightPeriodSpread ⋆ TotalRoomCapacity, StudentPerExamMed + NrOfConflictMean, StudentPerExamEntropy + CliqueCalculationTime, ExamsPerStudentMax + RoomsPerExamSD, ExamPerStudentMean / PeriodPerExamMean, ExamPerStudentVarCoef ⋆ ExamLengthMax, ExamPerStudentSD ⋆ TotalExamCap/TotalRoomAndNrPeriods, ExamPerStudent1Quant / PeriodPerExamMed, ExamPerStudentMeanMax ⋆ ExamPerRoomMean, AllRoomExams / ComputationalTime, TotalRoomCapacity ⋆ WCCSD

Table A.5: The features selected by different methods for the SACP solver from the original and the extended versions of the SA_time datasets

FS method	**Selected features**
FFS (64)	NumberOfStudents, NumberOfExams, NumberOfRooms, NumberOfPeriods, NumberOfDays, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfExclusionConstraints, NumberOfRoomExclusiveConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, TimeRelatedFractionOfConstraints, RoomRelatedFractionOfConstraints, WeightPeriodSpread, StudentPerExamMax, StudentPerExamMean, StudentPerExamMed, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, ExamsPerStudentMax, ExamPerStudent5thPer, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent1Quant, ExamPerStudent3Q, ExamPerStudentEntropy, ExamPerStudentSkew, ExamPerStudentMinMax, ExamPerStudentMeanMax, NrOfConflictMax, NrOfConflictMean, NrOfConflictSD, RoomsPerExam95P, AllPeriodExam, PeriodPerExam95P, PeriodPerExamMean, ExamLengthMax, RatioNrOfConflictsOverExams, RoomCapacityMax, RoomCapacityMed, RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, TotalExamCap/TotalRoomAndNrPeriods, ExamPerRoomMax, RoomForAllExamsWithoutPenaltyOverTotal, PeriodLengthMax, ExamPerPeriodMax, ExamPerPeriodMin, AvNrOfPeriodPerDay, CC5th, CCmean, CCMinMax, WCCVarCoef, CliqueSizeMax, CliqueSizeMin, CliqueSizemean, CliqueSizeVarCoef, CliqueCalculationTime, GreedyScore
RFE (17)	NumberOfStudents, NumberOfPeriods, NumberOfDays, TimeRelatedFractionOfConstraints, FrontLoadPenalty, StudentPerExamMin, StudentPerExamVarCoef, StudentPerExam3Q, StudentPerExamMeanMax, ExamsPerStudentMax, ExamPerStudentMean, ExamPerStudentSD, ExamPerStudent3Q, ExamPerStudentMeanMax, NrOfConflict5thP, AllRoomExams, RoomsPerExam95P
BFS (13)	NumberOfStudents, NumberOfRooms, StudentPerExam1Q, StudentPerExamEntropy, ExamsPerStudentMax, ExamPerStudentMean, ExamPerStudentSD, ExamPerStudentMinMax, NrOfConflictMax, TotalExamCap/TotalRoomAndNrPeriods, RoomForAllExamsWithoutPenaltyOverTotal, CliqueSizeVarCoef, CliqueCalculationTime
F + RFE2 (26)	NumberOfStudents * NumberOfStudents, NumberOfStudents - NumberOfRooms, NumberOfStudents * NumberOfPeriods, NumberOfStudents / NumberOfDays, NumberOfStudents / NumberOfAfterConstraints, NumberOfStudents + NumberOfRoomExclusiveConstraints, NumberOfStudents - RoomRelatedFractionOfConstraints, NumberOfStudents / WeightPeriodSpread, NumberOfStudents - WeightPeriodSpread, NumberOfStudents * WeightNumberOfLargeExams, NumberOfStudents / WeightNumberOfLargeExams, NumberOfStudents - FrontLoadPenalty, NumberOfStudents * FrontLoadWeight, NumberOfStudents / FrontLoadWeight, NumberOfStudents - FrontLoadWeight, NumberOfStudents + FrontLoadWeight, NumberOfStudents * StudentPerExamMax, NumberOfStudents / StudentPerExamMax, NumberOfStudents - StudentPerExamMax, NumberOfStudents + StudentPerExamMax, NumberOfStudents * StudentPerExamMin, NumberOfStudents / StudentPerExamMin, NumberOfStudents + StudentPerExam3Q, NumberOfStudents - StudentPerExamEntropy, NumberOfStudents / StudentPerExamMinMax, NumberOfStudents + StudentPerExamMinMax
F + BFS (9)	WeightPeriodSpread * TotalRoomCapacity, StudentPerExamMed + NrOfConflict3Q, StudentPerExamEntropy / RoomForAllExamsWithoutPenalty, StudentPerExamEntropy + CliqueCalculationTime, ExamsPerStudentMax + RoomsPerExamSD, ExamPerStudentMedian / PeriodPerExamMed, ExamPerStudentSD * TotalExamCap/TotalRoomAndNrPeriods, nrOfConflicts / WCCSD, TotalRoomCapacity / CliqueSizeVarCoef

Table A.6: The features selected by different methods for the GRASP solver from the original and the extended versions of the SA_time datasets

Table A.7: The features selected by different methods for the SA solver from the original and the extended versions of the SA_time datasets

FS method	Selected features
FFS (73)	NumberOfRooms, NumberOfDays, Slackness, NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfExclusionConstraints, NumberOfRoomExclusiveConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, WeightPeriodSpread, StudentPerExamMax, StudentPerExamMean, StudentPerExamMed, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, ExamsPerStudentMax, ExamPerStudent5thPer, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent1Quant, ExamPerStudent3Q, ExamPerStudentEntropy, ExamPerStudentMinMax, ExamPerStudentMeanMax, NrOfConflictMax, NrOfConflictMean, NrOfConflictSD, RoomsPerExam95P, PeriodPerExam5Per, PeriodPerExamMean, PeriodPerExamSD, nrOfConflicts, RatioNrOfConflictsOverExams, RoomCapacityMax, RoomCapacityMin, RoomCapacityMed, RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, ExamPerRoomMax, ExamPerRoomMin, PeriodLengthMean, ExamPerPeriodMean, CCSD, WCC5th, WCCmean, WCCVarCoef, WCCSD, WCCthird, WCCMinMax, CCCalculationTime, CliqueSizeMax, CliqueSizeMin, CliqueSizeVarCoef, CliqueSizeFirstQuart, CliqueSizeThird, CliqueSizeMinMax, CliqueCalculationTime, Greedy Number of Unassigned Variables, Greedy Score, Greedy Independent Set - Min, Greedy Independent Set - Median, Greedy Independent Set MinMax , Modularity, ComputationalTime, MaxMembers, MeanMembers, MedMembers
RFE (12)	NumberOfRoomExclusiveConstraints, WeightPeriodSpread, WeightMixedDuration, FrontLoadPenalty, StudentPerExamVarCoef, StudentPerExamMeanMax, ExamsPerStudentMax, ExamPerStudentEntropy, NrOfConflictMed, NrOfConflictSkew, RoomsPerExam95P, RoomsPerExam5Per
BFS (6)	NumberOfRoomExclusiveConstraints, StudentPerExamMax, ExamPerStudentVarCoef, ExamPerStudentMinMax, TotalRoomCapacity, CliqueCalculationTime
F+RFE2 (30)	NumberOfStudents * NumberOfStudents, NumberOfStudents + NumberOfStudents, NumberOfStudents / NumberOfExams, NumberOfStudents / NumberOfRooms, NumberOfStudents - NumberOfRooms, NumberOfStudents + NumberOfRooms, NumberOfStudents * NumberOfPeriods, NumberOfStudents - NumberOfPeriods, NumberOfStudents + NumberOfPeriods, NumberOfStudents * NumberOfDays, NumberOfStudents / NumberOfDays, NumberOfStudents - NumberOfDays, NumberOfStudents + NumberOfDays, NumberOfStudents * Slackness, NumberOfStudents / Slackness, NumberOfStudents - Slackness, NumberOfStudents + Slackness, NumberOfStudents * NumberOfExamsOverSlack, NumberOfStudents / NumberOfExamsOverSlack, NumberOfStudents - NumberOfExamsOverSlack, NumberOfStudents + NumberOfExamsOverSlack, NumberOfStudents * NumberOfAfterConstraints, NumberOfStudents / NumberOfAfterConstraints, NumberOfStudents / NumberOfCoincidenceConstraints, NumberOfStudents - NumberOfCoincidenceConstraints, NumberOfStudents + NumberOfCoincidenceConstraints,NumberOfStudents * NumberOfExclusionConstraints, NumberOfStudents / NumberOfExclusionConstraints, NumberOfStudents - NumberOfExclusionConstraints, NumberOfStudents + NumberOfExclusionConstraints
F+BFS (11)	Slackness - StudentPerExamMax, WeightPeriodSpread * ExamPerStudent5thPer, WeightPeriodSpread * TotalRoomCapacity, ExamsPerStudentMax + Greedy Number of Unassigned Variables, ExamPerStudentVarCoef * PeriodLengthMean, ExamPerStudent1Quant / PeriodPerExam5Per, ExamPerStudent1Quant * ExamPerPeriodMed, ExamPerStudentMinMax * TotalExamCap/TotalRoomAndNrPeriods, RoomsPerExam95P * WCC5th, TotalRoomCapacity * Greedy Number of Used Colors Per Period, Greedy Number of Used Colors / ComputationalTime

A.2 Feature Selection Results for The Algorithm Selection Problem

Table A.8: The most selected features among all datasets for the Algorithm Selection problem

Selected features	Frequency	Frequency (%)
NumberOfCoincidenceConstraints	12	0.67%
NumberOfExamsOverSlack	11	0.61%
ExamPerPeriodVarCoef	10	0.56%
GreedyNumberOfUsedColorsPerPeriod	10	0.56%
GreedyIndependentSetVC	9	0.5%
TotalRoomCapacity	9	0.5%
GreedyScore	9	0.5%
GreedyIndependentSetSD	8	0.44%
GreedyNumberOfUsedColors	8	0.44%
TotalExamCap/TotalRoomAndNrPeriods	8	0.44%
RoomsPerExamMean	8	0.44%
StudentPerExamSD	7	0.39%
ExamPerPeriodSD	7	0.39%
NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables	6	0.33%
TotalRoomCapacityMultNrOfPeriods	6	0.33%
StudentPerExamMed	6	0.33%
RoomCapacityMin	6	0.33%
ExamPerStudentSkew	6	0.33%
ConflictGraphDensity	6	0.33%
NumberOfAfterConstraints	6	0.33%
TotalNumberOfConstraints	5	0.28%
ExamPerStudentSD	5	0.28%
CliqueCalculationTime	5	0.28%
NumberOfCoincidenceConstraints / ExamPerPeriodSD	5	0.28%
CCEntropy	5	0.28%
GreedyIndependentSetMinMax	5	0.28%
ExamPerStudentEntropy	5	0.28%
ExamPerStudentMean	5	0.28%
WeightPeriodSpread * TotalExamCap/TotalRoomAndNrPeriods	5	0.28%
WCCEntropy	5	0.28%

Table A.9: The most selected features among the original and the decomposed versions of the extended datasets for the Algorithm Selection problem

Selected features	**Frequency**	**Frequency (%)**
NumberOfCoincidenceConstraints	210	6.66%
GreedyScore	208	6.6%
NumberOfExamsOverSlack	154	4.88%
GreedyNumberOfUnassignedVariables	100	3.17%
GreedyNumberOfUsedColorsPerPeriod	95	3.01%
TotalExamCap/TotalRoomAndNrPeriods	88	2.79%
GreedyNumberOfUsedColors	85	2.7%
CliqueCalculationTime	76	2.41%
TotalRoomCapacityMultNrOfPeriods	70	2.22%
TotalRoomCapacity	65	2.06%
ComputationalTime	65	2.06%
GreedyIndependentSetVC	61	1.93%
TimeRelatedFractionOfConstraints	49	1.55%
WeightPeriodSpread	45	1.43%
NumberOfAfterConstraints	42	1.33%
GreedyIndependentSetSD	41	1.3%
NumberOfRoomExclusiveConstraints	34	1.08%
StudentPerExamMax	32	1.01%
WeightMixedDuration	31	0.98%
GreedyIndependentSetMax	31	0.98%
NumberOfExclusionConstraints	30	0.95%
StudentPerExamMean	30	0.95%
MaxMembers	29	0.92%
GreedyIndependentSet3Q	28	0.89%
NumberOfTimeRelatedConstraints	27	0.86%
Modularity	27	0.86%
WeightTwoInADay	26	0.82%
RoomCapacityMax	25	0.79%
RoomCapacityMean	25	0.79%

Table A.10: The features selected by different FS methods for the standardized and discretized versions of the SA_3class dataset

FS method	**Selected features**
S+RFE (12)	NumberOfCoincidenceConstraints, TotalNumberOfConstraints , StudentPerExamMed, StudentPerExam3Q, CCVarCoef, WCC95th, WCCSD, WCCEntropy, CliqueSizeMean, CliqueSizeVarCoef, CliqueCalculationTime, GreedyNumberOfUsedColors
S+BFS (19)	NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, ExamPerStudentEntropy, NrOfConflictMean, NrOneRoomExam, RoomsPerExamMean, PeriodPerExamSD, RoomCapacityMin, TotalExamCap/TotalRoomAndNrPeriods, ExamPerPeriodVarCoef, CC95th, CliqueSizeMin, CliqueSizeMinMax, GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyScore, GreedyIndependentSetVC, GreedyIndependentSetMinMax
Ext+S+BFS (61)	NumberOfExams/StudentPerExamEntropy, NumberOfExamsOverSlack / NumberOfCoincidenceConstraints, NumberOfExamsOverSlack * RoomRelatedFractionOfConstraints, NumberOfExamsOverSlack * StudentPerExamSkew, NumberOfExamsOverSlack − AllPeriodExamOverTotal, NumberOfExamsOverSlack + PeriodPerExamSD, NumberOfExamsOverSlack + ExamLengthSD, NumberOfExamsOverSlack * NrOfPeriodsWithPenalty, NumberOfExamsOverSlack / NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfExamsOverSlack + CCEntropy, NumberOfAfterConstraints * ExamPerStudentSkew, NumberOfCoincidenceConstraints * WeightPeriodSpread, NumberOfCoincidenceConstraints + ExamPerStudentMinMax, NumberOfCoincidenceConstraints / NrOneRoomExam, NumberOfCoincidenceConstraints / OneRoomOverTotalExams, NumberOfCoincidenceConstraints * ConflictGraphDensity, NumberOfCoincidenceConstraints / MaxRoomPenalty, NumberOfCoincidenceConstraints / ExamPerPeriodSD, NumberOfCoincidenceConstraints * NrOfPeriodsForAllExamsWithoutPenalty, NumberOfCoincidenceConstraints * NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfCoincidenceConstraints + CliqueSizeMinMax, NumberOfCoincidenceConstraints / GreedyNumberOfUsedColorsPerPeriod, NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints / GreedyScore, NumberOfCoincidenceConstraints * GreedyIndependent SetVC, NumberOfCoincidenceConstraints * GreedyIndependentSet, MeanMax, WeightPeriodSpread * TotalExamCap/TotalRoomAndNrPeriods, StudentPerExamMax / GreedyNumberOfUnassignedVariables, StudentPerExamMinMax / GreedyNumberOfUnassignedVariables, ExamPerStudentVarCoef + ExamPerPeriodSD, ExamPerStudentVarCoef − GreedyIndependentSetVC, ExamPerStudentSD / SDMembers, ExamPerStudentSkew * ExamPerRoomSD, ExamPerStudentMinMax / GreedyNumberOfUnassignedVariables, NrOfConflictSkew * TotalRoomCapacity, RoomsPerExam5Per * CCmean, PeriodPerExamVarCoef + TotalExamCap/TotalRoomAndNrPeriods, PeriodPerExamSD + TotalExamCap/TotalRoomAndNrPeriods, TotalRoomCapacity − TotalExamCap/TotalRoomAndNrPeriods, TotalRoomCapacity * Ccmed, TotalRoomCapacity + WCC95th, TotalRoomCapacity + WCCFirstQuart, TotalExamCap/TotalRoomAndNrPeriods * PeriodLengthMed, TotalExamCap/TotalRoomAndNrPeriods − WCCMeanMax, TotalExamCap/TotalRoomAndNrPeriods / CliqueSizeMin, TotalExamCap/TotalRoomAndNrPeriods − GreedyNumber of Used Colors Per Period TotalExamCap/TotalRoomAndNrPeriods − GreedyIndependentSetVC, ExamPerRoomMed / GreedyIndependentSet3thQ, ExamPerRoomSD / GreedyScore, RoomForAllExamsWithoutPenaltyOverTotal / GreedyNumberofUnassignedVariables, PeriodLengthMean + ExamPerPeriodVarCoef, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal * GreedyIndependentSetVC, CC5th / GreedyNumberOfUsedColors, CCThird * GreedyIndependentSetVC, WCCVarCoef / GreedyNumberofUsedColorsPerPeriod, WCCFirstQuart / GreedyNumberofUnassignedVariables, WCCEntropy − GreedyIndependentSetSD , WCCMinMax + GreedyNumberofUnassignedVariables, WCCMinMax − MeanMembers, CliqueSizeVarCoef / GreedyNumberofUsedColorsPerPeriod, CliqueSizeMinMax / GreedyNumberOfUnassignedVariables
D+RFE (39)	NumberOfStudents, NumberOfRooms, NumberOfExamsOverSlack, NumberOfCoincidenceConstraints, ExamPerStudentMean, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent3Q, ExamPerStudentSkew, NrOneRoomExam, RoomsPerExam95P, RoomsPerExamMean, RoomsPerExamSD, PeriodPerExam5Per, nrOfConflicts, ConflictGraphDensity, RoomCapacityMax, RoomCapacityMin, TotalRoomCapacity, TotalExamCap/TotalRoomAndNrPeriods, ExamOverRoomMeanCapacity, ExamPerPeriodVarCoef, ExamPerPeriodSD, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, CCEntropy, WCC5th, WCCEntropy, WCCMinMax, WCCMeanMax, CCCalculationTime, CliqueSizeMax, CliqueCalculationTime, GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyNumberOfUnassignedVariables, GreedyScore, GreedyIndependentSetVCGreedy, IndependentSetSD, ComputationalTime
D+BFS (21)	NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, ExamPerStudentSD, ExamPerStudentMinMax, NrOneRoomExam, RoomsPerExamMean, PeriodPerExamSD, ConflictGraphDensity, RoomCapacityMin, TotalRoomCapacity, TotalExamCap/TotalRoomAndNrPeriods, ExamPerPeriodVarCoef, ExamPerPeriodSD, CCEntropy, CliqueSizeMin , GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyScore, GreedyIndependentSetVC, GreedyIndependentSetMinMax
Ext+D+BFS (71)	NumberOfExams / GreedyIndependentSetMax, NumberOfRooms * GreedyNumberofUsedColors, Slackness * GreedyScore, Slackness / MinMembers, NumberOfExamsOverSlack / NumberOfCoincidenceConstraints, NumberOfExamsOverSlack / RoomsPerExam5Per, NumberOfExamsOverSlack / AllPeriodExamOverTotal, NumberOfExamsOverSlack + PeriodPerExamSD, NumberOfExamsOverSlack / ConflictGraphDensity, NumberOfExamsOverSlack / Ccmed, NumberOfExamsOverSlack * GreedyScore, NumberOfAfterConstraints * GreedyNumberOfUsedColors, NumberOfAfterConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints / WeightTwoInADay, NumberOfCoincidenceConstraints − StudentPerExam1Q, NumberOfCoincidenceConstraints − ExamsPerStudentMax, NumberOfCoincidenceConstraints − OneRoomOverTotalExams, NumberOfCoincidenceConstraints * AllRoomOverTotalExams, NumberOfCoincidenceConstraints / ExamLengthMean, NumberOfCoincidenceConstraints * ConflictGraphDensity, NumberOfCoincidenceConstraints / TotalExamCap/TotalRoomAndNrPeriods, NumberOfCoincidenceConstraints / ExamPerPeriodSD, NumberOfCoincidenceConstraints * NrOfPeriodsForAllExamsWithoutPenalty, NumberOfCoincidenceConstraints * NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfCoincidenceConstraints − WCC5th, NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints + GreedyIndependentSetMedian, NumberOfCoincidenceConstraints * GreedyIndependentSetVC, NumberOfCoincidenceConstraints * GreedyIndependentSetMeanMax, NumberOfRoomExclusiveConstraints + GreedyNumberofUsedColors, WeightTwoInTheRow + GreedyNumberofUsedColors, WeightPeriodSpread − GreedyNumberofUsedColors, StudentPerExam3Q − GreedyNumberofUsedColorsPerPeriod, StudentPerExamMinMax / GreedyNumberofUsedColorsPerPeriod, ExamPerStudent5thPer / TotalExamCap/TotalRoomAndNrPeriods, ExamPerStudent5thPer / ComputationalTime, ExamPerStudentVarCoef / GreedyIndependentSetVC, ExamPerStudentSD / GreedyNumberofUsedColors, ExamPerStudentMeanMax + CC5th, NrOfConflict1Q / GreedyNumberofUnassignedVariables, NrOfConflictSkew + TotalRoomCapacity NrOneRoomExam − WCCmed, RoomsPerExamSD + PeriodPerExam5Per, PeriodPerExamMean * RoomForAllExamsWithoutPenalty, PeriodPerExamSD + TotalExamCap/TotalRoomAndNrPeriods, TotalRoomCapacity + WCC95th, TotalRoomCapacity + WCCFirstQuart, TotalExamCap/TotalRoomAndNrPeriods + ExamPerPeriodVarCoef, TotalExamCap/TotalRoomAndNrPeriods / Ccmed, TotalExamCap/TotalRoomAndNrPeriods / CliqueSizeMin, TotalExamCap/TotalRoomAndNrPeriods / GreedyNumberofUnassignedVariables, TotalExamCap/TotalRoomAndNrPeriods − GreedyIndependentSetVC, ExamPerRoomMed / GreedyIndependentSet3thQ, ExamPerRoomSD / GreedyScore, RoomForAllExamsWithoutPenaltyOverTotal − GreedyNumberofUsedColorsPerPeriod, PeriodPenaltyMax + GreedyNumberofUsedColorsPerPeriod, ExamPerPeriodMin + GreedyIndependentSetVC , ExamPerPeriodMed + GreedyNumberofUsedColorsPerPeriod, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal + GreedyScore, CC95th * GreedyIndependentSetVCC, FirstQuart / GreedyNumberOfUsedColors, CCEntropy * GreedyNumberOfUsedColorsPerPeriod, WCC95th / CliqueSizemean, WCCThird / GreedyNumberofUsedColors, WCCEntropy / GreedyIndependentSetSD, WCCEntropy − GreedyIndependentSetSD, WCCMinMax / GreedyNumberofUsedColors, CCCalculationTime / GreedyNumberofUnassignedVariables, CliqueSizeMax / ComputationalTime, GreedyNumberofUsedColorsPerPeriod * Modularity, GreedyNumberofUnassignedVariables / GreedyIndependentSetVC

Table A.11: The features selected by the RFE method for the standardized extended version of the `SA_3class` dataset

FS method	Selected features
Ext+S+RFE (235)	NumberOfCoincidenceConstraints ⋆ NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfCoincidenceConstraints ⋆ CC95th, NumberOfCoincidenceConstraints + CC95th, NumberOfCoincidenceConstraints / CC5th, NumberOfCoincidenceConstraints ⋆ CCmean, NumberOfCoincidenceConstraints ⋆ Ccmed, NumberOfCoincidenceConstraints ⋆ CCVarCoef, NumberOfCoincidenceConstraints ⋆ CCSD, NumberOfCoincidenceConstraints ⋆ CCFirstQuart, NumberOfCoincidenceConstraints ⋆ CCThird, NumberOfCoincidenceConstraints ⋆ CCEntropy, NumberOfCoincidenceConstraints ⋆ CCSkew, NumberOfCoincidenceConstraints / CCMinMax, NumberOfCoincidenceConstraints ⋆ CCMeanMax, NumberOfCoincidenceConstraints ⋆ WCC95th, NumberOfCoincidenceConstraints / WCC5th, NumberOfCoincidenceConstraints ⋆ WCCmean, NumberOfCoincidenceConstraints ⋆ WCCVarCoef, NumberOfCoincidenceConstraints ⋆ WCCSD, NumberOfCoincidenceConstraints ⋆ WCCEntropy, NumberOfCoincidenceConstraints ⋆ WCCSkew, NumberOfCoincidenceConstraints / WCCMinMax, NumberOfCoincidenceConstraints ⋆ WCCMeanMax, NumberOfCoincidenceConstraints / CCCalculationTime, NumberOfCoincidenceConstraints + CliqueSizeMax, NumberOfCoincidenceConstraints - CliqueSizeMin, NumberOfCoincidenceConstraints + CliqueSizeMin, NumberOfCoincidenceConstraints ⋆ CliqueSizeMean, NumberOfCoincidenceConstraints + CliqueSizeMed, NumberOfCoincidenceConstraints ⋆ CliqueSizeVarCoef, NumberOfCoincidenceConstraints ⋆ CliqueSizeSD, NumberOfCoincidenceConstraints + CliqueSizeFirstQuart, NumberOfCoincidenceConstraints + CliqueSizeThird, NumberOfCoincidenceConstraints ⋆ CliqueSizeEntropy, NumberOfCoincidenceConstraints ⋆ CliqueSizeSkew, NumberOfCoincidenceConstraints ⋆ CliqueSizeMinMax, NumberOfCoincidenceConstraints + CliqueSizeMinMax, NumberOfCoincidenceConstraints - CliqueCalculationTime, NumberOfCoincidenceConstraints ⋆ GreedyNumberOfUsedColors, NumberOfCoincidenceConstraints / GreedyNumberOfUsedColors, NumberOfCoincidenceConstraints ⋆ GreedyNumberOfUsedColorsPerPeriod, NumberOfCoincidenceConstraints / GreedyNumberOfUsedColorsPerPeriod, NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints - GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints ⋆ GreedyScore, NumberOfCoincidenceConstraints / GreedyScore, NumberOfCoincidenceConstraints - GreedyScore, NumberOfCoincidenceConstraints / GreedyIndependentSetMin, NumberOfCoincidenceConstraints - GreedyIndependentSetMin, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetMean, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetVC, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetSD, NumberOfCoincidenceConstraints / GreedyIndependentSetSD, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetEntropy, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetSkew, NumberOfCoincidenceConstraints + ComputationalTime, NumberOfCoincidenceConstraints + MinMembers, NumberOfExclusionConstraints + StudentPerExamEntropy, NumberOfExclusionConstraints - ExamPerStudentSD, NumberOfExclusionConstraints - ExamPerStudentEntropy, NumberOfExclusionConstraints - TotalExamCap/TotalRoomAndNrPeriods, NumberOfExclusionConstraints + GreedyIndependentSetVC, NumberOfRoomExclusiveConstraints - ExamPerStudentMean, NumberOfRoomExclusiveConstraints + ExamPerStudentSD, NumberOfRoomExclusiveConstraints - ExamPerStudentEntropy, NumberOfRoomExclusiveConstraints + ExamPerStudentEntropy, NumberOfRoomExclusiveConstraints + TotalExamCap/TotalRoomAndNrPeriods, NumberOfRoomExclusiveConstraints - GreedyScore, NumberOfTimeRelatedConstraints + ExamPerStudentMean, NumberOfTimeRelatedConstraints + ExamPerStudentSD, NumberOfTimeRelatedConstraints + ExamPerStudentEntropy, NumberOfTimeRelatedConstraints - nrOfConflicts, NumberOfTimeRelatedConstraints - TotalExamCap / TotalRoomAndNrPeriods, NumberOfTimeRelatedConstraints + TotalExamCap/TotalRoomAndNrPeriods, NumberOfTimeRelatedConstraints + CliqueSizeMean, NumberOfTimeRelatedConstraints / GreedyNumberOfUsedColors, NumberOfTimeRelatedConstraints + ComputationalTime, TotalNumberOfConstraints + ExamPerStudentEntropy, TotalNumberOfConstraints - nrOfConflicts, TotalNumberOfConstraints - TotalExamCap/TotalRoomAndNrPeriods, TotalNumberOfConstraints - CliqueCalculationTime, TotalNumberOfConstraints / GreedyNumberOfUsedColors, TotalNumberOfConstraints ⋆ GreedyScore, TotalNumberOfConstraints / GreedyScore, TimeRelatedFractionOfConstraints + TotalExamCap / TotalRoomAndNrPeriods, TimeRelatedFractionOfConstraints / GreedyNumberOfUsedColorsPerPeriod, TimeRelatedFractionOfConstraints / GreedyNumberOfUnassignedVariables, RoomRelatedFractionOfConstraints + TotalExamCap/TotalRoomAndNrPeriods, RoomRelatedFractionOfConstraints / GreedyNumberOfUsedColors, RoomRelatedFractionOfConstraints / GreedyNumberOfUsedColorsPerPeriod, RoomRelatedFractionOfConstraints / GreedyScore, WeightTwoInTheRow + ExamPerStudentEntropy, WeightTwoInTheRow + TotalExamCap/TotalRoomAndNrPeriods, WeightTwoInTheRow / GreedyNumberOfUsedColorsPerPeriod, WeightTwoInTheRow / GreedyScore, WeightTwoInTheRow - ComputationalTime, WeightTwoInADay + ExamPerStudentEntropy, WeightTwoInADay + nrOfConflicts, WeightTwoInADay + TotalExamCap/TotalRoomAndNrPeriods, WeightTwoInADay / GreedyNumberOfUsedColors, WeightTwoInADay / GreedyNumberOfUsedColorsPerPeriod, WeightTwoInADay / GreedyScore, WeightPeriodSpread + ExamPerStudentEntropy, WeightPeriodSpread + NrOfConflictMean, WeightPeriodSpread + nrOfConflicts, WeightPeriodSpread - RatioNrOfConflictsOverExams, WeightPeriodSpread + RatioNrOfConflictsOverExams, WeightPeriodSpread - TotalExamCap / TotalRoomAndNrPeriods, WeightMixedDuration - ExamPerStudentMean, WeightMixedDuration - ExamPerStudentSD, WeightMixedDuration - TotalExamCap/TotalRoomAndNrPeriods, CliqueSizemean ⋆ GreedyIndependentSetSD, CliqueSizemean + ComputationalTime, CliqueSizemean + MaxMembers, CliqueSizeMean + MinMembers, CliqueSizeMean, CliqueSizeMed + Greedy Score, CliqueSizeMed ⋆ GreedyIndependentSetMedian, CliqueSizeMed ⋆ GreedyIndependentSet3thQ, CliqueSizeMed - ComputationalTime, CliqueSizeMed + ComputationalTime, CliqueSizeVarCoef ⋆ CliqueSizeMinMax, CliqueSizeVarCoef - CliqueCalculationTime , CliqueSizeVarCoef / GreedyNumberOfUsedColors, CliqueSizeVarCoef / GreedyNumberOfUsedColorsPerPeriod, CliqueSizeVarCoef / GreedyNumberOfUnassignedVariables, CliqueSizeVarCoef / GreedyScore, CliqueSizeVarCoefMeanMembers, CliqueSizeSD + CliqueSizeEntropy, CliqueSizeSD + CliqueCalculationTime, CliqueSizeSD ⋆ GreedyNumberOfUsedColors, CliqueSizeSD / GreedyNumberOfUsedColors, CliqueSizeSD + GreedyScore, CliqueSizeSD ⋆ GreedyIndependentSetMean, CliqueSizeSD + MeanMembers, CliqueSizeFirstQuart / GreedyNumberOfUnassignedVariables, CliqueSizeFirstQuart + GreedyScore, CliqueSizeFirstQuart + ComputationalTime, CliqueSizeThird - CliqueSizeEntropy, CliqueSizeThird / GreedyNumberOfUsedColors, CliqueSizeThird ⋆ GreedyNumberOfUsedColorsPerPeriod, CliqueSizeThird ⋆ GreedyScore CliqueSizeThird + ComputationalTime, CliqueSizeEntropy / GreedyNumberOfUsedColors, CliqueSizeEntropy / GreedyNumberOfUnassignedVariables, CliqueSizeEntropy + GreedyNumberOfUnassignedVariables, CliqueSizeEntropy + GreedyIndependentSetMin, CliqueSizeEntropy ⋆ GreedyIndependentSetMedian, CliqueSizeEntropy - GreedyIndependentSetVC, CliqueSizeEntropy - MinMembers, CliqueSizeEntropy + MinMembers, CliqueSizeEntropy + MeanMembers, CliqueSizeEntropy, CliqueSizeSkew / GreedyNumberOfUsedColors, CliqueSizeSkew / GreedyNumberOfUnassignedVariables, CliqueSizeSkew / GreedyScore, CliqueSizeMinMax - CliqueCalculationTime, CliqueSizeMinMax / GreedyNumberOfUsedColors, CliqueSizeMinMax / GreedyNumberOfUsedColorsPerPeriod, CliqueSizeMinMax / GreedyNumberOfUnassignedVariables, CliqueSizeMinMax / GreedyScore, CliqueSizeMinMax + GreedyScore, CliqueSizeMinMax / MinMembers, CliqueSizeMinMax - MeanMembers, CliqueSizeMeanMax / GreedyScore, CliqueCalculationTime + CliqueCalculationTime, CliqueCalculationTime / GreedyNumberOfUnassignedVariables, CliqueCalculationTime / GreedyScore, CliqueCalculationTime + GreedyIndependentSetMin, CliqueCalculationTime ⋆ GreedyIndependentSetMean, CliqueCalculationTime + GreedyIndependentSetMedian, CliqueCalculationTime - GreedyIndependentSetSD, CliqueCalculationTime - GreedyIndependentSetFirstQ, CliqueCalculationTime + GreedyIndependentSet3thQ, CliqueCalculationTime ⋆ ComputationalTime, CliqueCalculationTime + MinMembers, CliqueCalculationTime + MeanMembers, CliqueCalculationTime + MedMembers, CliqueCalculationTime, GreedyNumberOfUsedColors ⋆ GreedyScore, GreedyNumberOfUsedColors - GreedyScore, GreedyNumberOfUsedColors + GreedyScore, GreedyNumberOfUsedColors ⋆ GreedyIndependentSetMean, GreedyNumberOfUsedColors ⋆ GreedyIndependentSetMedian, GreedyNumberOfUsedColors ⋆ GreedyIndependentSetVC, GreedyNumberOfUsedColors / GreedyIndependentSetVC, GreedyNumberOfUsedColors ⋆ GreedyIndependentSetSD, GreedyNumberOfUsedColors ⋆ GreedyIndependentSetFirstQ, GreedyNumberOfUsedColors ⋆ GreedyIndependentSetEntropy, GreedyNumberOfUsedColors / GreedyIndependentSetSkew, GreedyNumberOfUsedColors / Modularity, GreedyNumberOfUsedColors ⋆ MaxMembers, GreedyNumberOfUsedColors / MinMembers, GreedyNumberOfUsedColors / MedMembers, GreedyNumberOfUsedColorsPerPeriod + GreedyNumberOfUsedColorsPerPeriod, GreedyNumberOfUsedColorsPerPeriod / GreedyScore, GreedyNumberOfUsedColorsPerPeriod + GreedyScore, GreedyNumberOfUsedColorsPerPeriod ⋆ GreedyIndependentSetMax, GreedyNumberOfUsedColorsPerPeriod / GreedyIndependentSetVC, GreedyNumberOfUsedColorsPerPeriod ⋆ GreedyIndependentSetSD, GreedyNumberOfUsedColorsPerPeriod ⋆ GreedyIndependentSetFirstQ, GreedyNumberOfUsedColorsPerPeriod ⋆ GreedyIndependentSet3thQ, GreedyNumberOfUsedColorsPerPeriod ⋆ GreedyIndependentSetEntropy, GreedyNumberOfUsedColorsPerPeriod / GreedyIndependentSetSkew, GreedyNumberOfUsedColorsPerPeriod / GreedyIndependentSetMinMax, GreedyNumberOfUsedColorsPerPeriod / Modularity, GreedyNumberOfUsedColorsPerPeriod ⋆ ComputationalTime, GreedyNumberOfUsedColorsPerPeriod ⋆ MaxMembers, GreedyNumberOfUsedColorsPerPeriod / MinMembers, GreedyNumberOfUsedColorsPerPeriod ⋆ MeanMembers, GreedyNumberOfUsedColorsPerPeriod + MeanMembers, GreedyNumberOfUsedColorsPerPeriod ⋆ SDMembers, GreedyNumberOfUsedColorsPerPeriod, GreedyNumberOfUnassignedVariables - GreedyScore, GreedyNumberOfUnassignedVariables + ComputationalTime, GreedyScore + GreedyScore, GreedyScore ⋆ GreedyIndependentSetMax, GreedyScore - GreedyIndependentSetMax GreedyScore / GreedyIndependentSetMin, GreedyScore - GreedyIndependentSetMin, GreedyScore + GreedyIndependentSetMin, GreedyScore ⋆ GreedyIndependentSetMean, GreedyScore ⋆ GreedyIndependentSetMedian, GreedyScore - GreedyIndependentSetMedian, GreedyScore + GreedyIndependentSetMedian, GreedyScore ⋆ GreedyIndependentSetVC, GreedyScore / GreedyIndependentSetVC, GreedyScore + GreedyIndependentSetVC, GreedyScore ⋆ GreedyIndependentSetSD, GreedyScore ⋆ GreedyIndependentSetFirstQ, GreedyScore / GreedyIndependentSetFirstQ, GreedyScore - GreedyIndependentSetFirstQ, GreedyScore ⋆ GreedyIndependentSet3thQ, GreedyScore - GreedyIndependentSet3thQ, GreedyScore ⋆ GreedyIndependentSetEntropy

Table A.12: The features selected by the RFE method for the discretized version of the `SA_3class` dataset

FS method	Selected features
Ext+D+RFE (191)	WeightTwoInADay + TotalExamCap / TotalRoomAndNrPeriods, WeightTwoInADay + WCCThird, WeightTwoInADay - WCCEntropy, WeightTwoInADay - CCCalculationTime, WeightTwoInADay + CCCalculationTime, WeightTwoInADay - CliqueCalculationTime, WeightTwoInADay + CliqueCalculationTime, WeightTwoInADay / GreedyNumberOfUsedColors, WeightTwoInADay + GreedyNumberOfUsedColors, WeightTwoInADay - GreedyNumberOfUsedColorsPerPeriod, WeightTwoInADay + GreedyNumberOfUsedColorsPerPeriod, WeightTwoInADay * GreedyNumberOfUnassignedVariables, WeightTwoInADay - GreedyScore, WeightTwoInADay + GreedyScore, WeightTwoInADay - ComputationalTime, WeightTwoInADay + ComputationalTime, WeightTwoInADay + MeanMembers, WeightPeriodSpread - NrOfConflictMean, WeightPeriodSpread + NrOfConflictMean, WeightPeriodSpread + NrOfConflictSD, WeightPeriodSpread - nrOfConflicts, WeightPeriodSpread + nrOfConflicts, WeightPeriodSpread - RatioNrOfConflictsOverExams, WeightPeriodSpread + RatioNrOfConflictsOverExams, WeightPeriodSpread * TotalRoomCapacity, WeightPeriodSpread * TotalExamCap / TotalRoomAndNrPeriods, WeightPeriodSpread - TotalExamCap /TotalRoomAndNrPeriods, WeightPeriodSpread + TotalExamCap / TotalRoomAndNrPeriods, WeightPeriodSpread + WCCmed, WeightPeriodSpread + CCCalculationTime, WeightPeriodSpread - CliqueCalculationTime, WeightPeriodSpread + CliqueCalculationTime, WeightPeriodSpread * GreedyNumberOfUsedColors, WeightPeriodSpread / GreedyNumberOfUsedColors, WeightPeriodSpread - GreedyNumberOfUsedColors, WeightPeriodSpread * GreedyNumberOfUsedColorsPerPeriod, WeightPeriodSpread / GreedyNumberOfUsedColorsPerPeriod, WeightPeriodSpread - GreedyNumberOfUsedColorsPerPeriod, WeightPeriodSpread * GreedyNumber OfUnassignedVariables, WeightPeriodSpread / GreedyNumberOfUnassignedVariables, WeightPeriodSpread * GreedyScore, WeightPeriodSpread / GreedyScore, WeightPeriodSpread - GreedyScore, WeightPeriodSpread + GreedyScore, WeightPeriodSpread - ComputationalTime, WeightPeriodSpread + ComputationalTime, WeightMixedDuration - StudentPerExam3Q, WeightMixedDuration - NrOfConflictMean, WeightMixedDuration + NrOfConflictMean, WeightMixedDuration / NrOfConflictMed, WeightMixedDuration - NrOfConflictSD, WeightMixedDuration + NrOfConflictSD, WeightMixedDuration - nrOfConflicts, WeightMixedDuration + nrOfConflicts, WeightMixedDuration - RatioNrOfConflictsOverExams, WeightMixedDuration + RatioNrOfConflictsOverExams, WeightMixedDuration - TotalExamCap/TotalRoomAndNrPeriods, WeightMixedDuration + TotalExamCap / TotalRoomAndNrPeriods, WeightMixedDuration + WCCmed, WeightMixedDuration + WCCEntropy, WeightMixedDuration - CCCalculationTime, WeightMixedDuration + CCCalculationTime, WeightMixedDuration - CliqueCalculationTime, WeightMixedDuration + CliqueCalculationTime, WeightMixedDuration * Greedy Number of Used Colors WeightMixedDuration + Greedy Number of Used Colors WeightMixedDuration + Greedy Number of Used Colors Per Period WeightMixedDuration * Greedy NumberOfUnassignedVariables, WeightMixedDuration * GreedyScore, WeightMixedDuration + GreedyScore, WeightMixedDuration - ComputationalTime, WeightMixedDuration + ComputationalTime, WeightMixedDuration - MedMembers, WeightNumberOfLargeExams + NrOfConflictMax, WeightNumberOfLargeExams - NrOfConflictMean, WeightNumberOfLargeExams - NrOfConflictSD, WeightNumberOfLargeExams - NrOfConflictEntropy, WeightNumberOfLargeExams + NrOfConflictEntropy, WeightNumberOfLargeExams - TotalExamCap / TotalRoomAndNrPeriods, WeightNumberOfLargeExams - WCCEntropy, WeightNumberOfLargeExams - CliqueCalculationTime, WeightNumberOfLargeExams - GreedyScore, FrontLoadPenalty - NrOfConflictMean, FrontLoadPenalty + TotalExamCap/TotalRoomAndNrPeriods, FrontLoadPenalty - CliqueCalculationTime, FrontLoadWeight + GreedyNumberOfUsedColors, StudentPerExamMax / GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod / GreedyIndependentSetVC, GreedyNumberOfUsedColorsPerPeriod + GreedyIndependentSetFirstQ, GreedyNumberOfUsedColorsPerPeriod * GreedyIndependentSet3thQ GreedyNumberOfUsedColorsPerPeriod / GreedyIndependentSetSkew GreedyNumberOfUsedColorsPerPeriod / GreedyIndependentSetMinMax GreedyNumberOfUsedColorsPerPeriod * GreedyIndependentSetMeanMax GreedyNumberOfUsedColorsPerPeriod * Modularity, GreedyNumberOfUsedColorsPerPeriod / Modularity, GreedyNumberOfUsedColorsPerPeriod * ComputationalTime, GreedyNumberOfUsedColorsPerPeriod - ComputationalTime, GreedyNumberOfUsedColorsPerPeriod * MaxMembers, GreedyNumberOfUsedColorsPerPeriod / MaxMembers, GreedyNumberOfUsedColorsPerPeriod - MaxMembers, GreedyNumberOfUsedColorsPerPeriod / MinMembers, GreedyNumberOfUsedColorsPerPeriod - MinMembers, GreedyNumberOfUsedColorsPerPeriod + MinMembers, GreedyNumberOfUsedColorsPerPeriod * MeanMembers, GreedyNumberOfUsedColorsPerPeriod / MeanMembers, GreedyNumberOfUsedColorsPerPeriod / MedMembers, GreedyNumberOfUsedColorsPerPeriod - MedMembers, GreedyNumberOfUsedColorsPerPeriod + MedMembers, GreedyNumberOfUsedColorsPerPeriod * VarCoefMembers, GreedyNumberOfUsedColorsPerPeriod * SDMembers, GreedyNumberOfUsedColorsPerPeriod / SDMembers, GreedyNumberOfUsedColorsPerPeriod, GreedyNumberOfUnassignedVariables * GreedyNumberOfUnassignedVariables, GreedyNumberOfUnassignedVariables + GreedyNumberOfUnassignedVariables, GreedyNumberOfUnassignedVariables / GreedyScore, GreedyNumberOfUnassignedVariables - GreedyScore, GreedyNumberOfUnassignedVariables + GreedyScore, GreedyNumberOfUnassignedVariables * GreedyIndependentSetMax, GreedyNumberOfUnassignedVariables / GreedyIndependentSetMin, GreedyNumberOfUnassignedVariables - GreedyIndependentSetMin, GreedyNumberOfUnassignedVariables * GreedyIndependentSetMean, GreedyNumberOfUnassignedVariables / GreedyIndependentSetMean, GreedyNumberOfUnassignedVariables * GreedyIndependentSetMedian, GreedyNumberOfUnassignedVariables / GreedyIndependentSetMedian, GreedyNumberOfUnassignedVariables * GreedyIndependentSetVC, GreedyNumberOfUnassignedVariables / GreedyIndependentSetVC, GreedyNumberOfUnassignedVariables * GreedyIndependentSetSD, GreedyNumberOfUnassignedVariables / GreedyIndependentSetSD, GreedyNumberOfUnassignedVariables * GreedyIndependentSet3thQ, GreedyNumberOfUnassignedVariables / GreedyIndependentSet3thQ, GreedyNumberOfUnassignedVariables * GreedyIndependentSetEntropy, GreedyNumberOfUnassignedVariables * GreedyIndependentSetSkew, GreedyNumberOfUnassignedVariables / GreedyIndependentSetSkew, GreedyNumberOfUnassignedVariables * GreedyIndependentSetMinMax, GreedyNumberOfUnassignedVariables * GreedyIndependentSetMeanMax, GreedyNumberOfUnassignedVariables / GreedyIndependentSetMeanMax, GreedyNumberOfUnassignedVariables * Modularity, GreedyNumberOfUnassignedVariables / Modularity, GreedyNumberOfUnassignedVariables * NrOfCommunities, GreedyNumberOfUnassignedVariables / NrOfCommunities, GreedyNumberOfUnassignedVariables * ComputationalTime, GreedyNumberOfUnassignedVariables / ComputationalTime, GreedyNumberOfUnassignedVariables - ComputationalTime, GreedyNumberOfUnassignedVariables + ComputationalTime, GreedyNumberOfUnassignedVariables * MaxMembers, GreedyNumberOfUnassignedVariables - MaxMembers, GreedyNumberOfUnassignedVariables + MaxMembers, GreedyNumberOfUnassignedVariables - MinMembers, GreedyNumberOfUnassignedVariables * MeanMembers, GreedyNumberOfUnassignedVariables + MeanMembers, GreedyNumberOfUnassignedVariables - MedMembers, GreedyNumberOfUnassignedVariables + MedMembers, GreedyNumberOfUnassignedVariables * VarCoefMembers, GreedyNumberOfUnassignedVariables / VarCoefMembers, GreedyNumberOfUnassignedVariables * SDMembersGreedyNumberOfUnassignedVariables, GreedyScore * GreedyScore, GreedyScore + GreedyScore, GreedyScore + GreedyIndependentSetMin, GreedyScore * GreedyIndependentSetMean, GreedyScore / GreedyIndependentSetMean, GreedyScore - GreedyIndependentSetMedian, GreedyScore + GreedyIndependentSetMedian, GreedyScore / GreedyIndependentSetVC, GreedyScore * GreedyIndependentSetSD, GreedyScore / GreedyIndependentSetSD, GreedyScore * GreedyIndependentSetFirstQ, GreedyScore - GreedyIndependentSetFirstQ, GreedyScore / GreedyIndependentSet3thQ, GreedyScore - GreedyIndependentSet3thQ, GreedyScore + GreedyIndependentSet3thQ, GreedyScore * GreedyIndependentSetSkew, GreedyScore / GreedyIndependentSetMinMax, GreedyScore * Modularity, GreedyScore * NrOfCommunities, GreedyScore * ComputationalTime, GreedyScore / ComputationalTime, GreedyScore * MaxMembers, GreedyScore / MaxMembers, GreedyScore - MaxMembers, GreedyScore - MinMembers, GreedyScore + MinMembers, GreedyScore * MeanMembers, GreedyScore / MeanMembers, GreedyScore / MedMembers, GreedyScore - MedMembers, GreedyScore + MedMembers, Greedy Score * SDMembers, GreedyScore / SDMembers, GreedyScore, GreedyIndependentSetMax - ComputationalTime

Table A.13: The features selected by different method on the standartized version of the `SA_2class` dataset

FS method	**Selected features**
S+RFE (55)	NumberOfExams, NumberOfPeriods, NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, RoomRelatedFractionOfConstraints, StudentPerExamMed, StudentPerExamSD, StudentPerExam3Q, StudentPerExamEntropy, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudentEntropy, ExamPerStudentSkew, ExamPerStudentMinMax, RoomsPerExamMean, RoomsPerExamSD, AllPeriodExamOverTotal, PeriodPerExam95P, PeriodPerExam5Per, PeriodPerExamMean, PeriodPerExamVarCoef, PeriodPerExamSD, ConflictGraphDensity, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, TotalExamCap / TotalRoomAndNrPeriods, ExamOverRoomMeanCapacity, ExamPerRoomMax, ExamPerRoomMin, ExamPerRoomMean, ExamPerRoomMed, ExamPerRoomSD, ExamPerPeriodMean, ExamPerPeriodMed, ExamPerPeriodVarCoef, ExamPerPeriodSD, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, CCmean, CCSD, CCSkew, WCCEntropy, CliqueSizeVarCoef, CliqueCalculationTime, GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyNumberOfUnassignedVariables, GreedyScore, GreedyIndependentSetMax, GreedyIndependentSetMean, GreedyIndependentSetVC, GreedyIndependentSetSD, GreedyIndependentSet3thQ
S+BFS(15)	NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, StudentPerExamSD, ExamPerStudent3Q, TotalRoomCapacity, TotalExamCap / TotalRoomAndNrPeriods, ExamPerPeriodVarCoef, ExamPerPeriodSD, CC95th, CliqueSizeMin, GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyIndependentSetVC, GreedyIndependentSetSD
Ext+S+RFE (27)	NumberOfExamsOverSlack / PeriodLengthMed, NumberOfExamsOverSlack * NrOfPeriodsWithPenalty, NumberOfExamsOverSlack + ExamPerPeriodVarCoef, NumberOfExamsOverSlack + ExamPerPeriodSD, NumberOfExamsOverSlack / NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfExamsOverSlack - NrOfPeriodsForAllExamsWithoutPenaltyOverTota, NumberOfExamsOverSlack / CC95th, NumberOfExamsOverSlack / CCmean, NumberOfExamsOverSlack / Ccmed, NumberOfExamsOverSlack / WCCmean, NumberOfCoincidenceConstraints * WeightPeriodSpread, GreedyScore / Greedy IndependentSetMinMax, GreedyScore / Modularity, GreedyScore / NrOfCommunities, GreedyScore * ComputationalTime, GreedyScore / ComputationalTime, GreedyScore / MaxMembers, GreedyScore + MinMembers, GreedyScore + MedMembers, GreedyScore / VarCoefMembers, GreedyIndependentSetVC * GreedyIndependentSetSD, GreedyIndependentSetVC - GreedyIndependentSet3thQ, GreedyIndependentSetVC / ComputationalTime, GreedyIndependentSetVC - MaxMembers, GreedyIndependentSetVC - MedMembers, GreedyIndependentSetSD / ComputationalTime, GreedyIndependentSetSD - ComputationalTime
Ext+S+BFS (100)	NumberOfStudents / TotalRoomCapacity, NumberOfStudents - TotalRoomCapacity, NumberOfExams / NumberOfCoincidenceConstraints, NumberOfExams / GreedyIndependentSetMax, NumberOfExamsOverSlack / NumberOfCoincidenceConstraints, NumberOfExamsOverSlack / NumberOfTimeRelatedConstraints, NumberOfExamsOverSlack / StudentPerExamMinMax, NumberOfExamsOverSlack / RoomsPerExam5Per, NumberOfExamsOverSlack - AllPeriodExamOverTotal, NumberOfExamsOverSlack + PeriodPerExamSD, NumberOfExamsOverSlack + ExamLengthSD, NumberOfExamsOverSlack / ConflictGraphDensity, NumberOfExamsOverSlack * NrOfPeriodsWithPenalty, NumberOfExamsOverSlack + ExamPerPeriodVarCoef, NumberOfExamsOverSlack / NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfExamsOverSlack + WCCSkew, NumberOfAfterConstraints / ExamPerStudent5thPer, NumberOfAfterConstraints * ExamPerStudentSkew, NumberOfAfterConstraints * PeriodPerExamSD, NumberOfAfterConstraints * ExamLengthVarCoef, NumberOfAfterConstraints * CliqueSizeVarCoef, NumberOfAfterConstraints * GreedyIndependentSetSkew, NumberOfCoincidenceConstraints * ExamPerStudentMedian, NumberOfCoincidenceConstraints + ExamPerStudentMinMax, NumberOfCoincidenceConstraints - NrOfConflictMax, NumberOfCoincidenceConstraints - OneRoomOverTotalExams, NumberOfCoincidenceConstraints + OneRoomOverTotalExams, NumberOfCoincidenceConstraints + PeriodPerExamMed, NumberOfCoincidenceConstraints / ExamLengthSD, NumberOfCoincidenceConstraints * ConflictGraphDensity, NumberOfCoincidenceConstraints / TotalExamCap / TotalRoomAndNrPeriods, NumberOfCoincidenceConstraints * PeriodLengthMin, NumberOfCoincidenceConstraints / ExamPerPeriodSD, NumberOfCoincidenceConstraints / GreedyNumberOfUsedColorsPerPeriod, NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints - GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints - GreedyIndependentSetMedian, NumberOfCoincidenceConstraints * GreedyIndependentSetMeanMax, TimeRelatedFractionOfConstraints * PeriodPerExam5Per, TimeRelatedFractionOfConstraints - ExamPerRoomSD, RoomRelatedFractionOfConstraints + ExamPerRoomSD, WeightTwoInADay * ExamPerPeriodSD, WeightPeriodSpread * TotalExamCap /TotalRoomAndNrPeriods, WeightPeriodSpread / CliqueSizeSD, StudentPerExamMax + CliqueSizeEntropy, StudentPerExamMin / ExamPerPeriodSD, StudentPerExamSkew - ExamPerStudentVarCoef, ExamsPerStudentMax / GreedyNumberOfUnassignedVariables, ExamPerStudent5thPer / TotalExamCap / TotalRoomAndNrPeriods, ExamPerStudentMean / GreedyNumberOfUnassignedVariables, ExamPerStudentVarCoef * NrOfConflict3Q, ExamPerStudent1Quant / NrOfConflictSD, ExamPerStudentSkew * ExamPerPeriodVarCoef, ExamPerStudentSkew / GreedyNumberOfUsedColorsPerPeriod, ExamPerStudentSkew * MaxMembers, ExamPerStudentMinMax * NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, ExamPerStudentMinMax / GreedyNumberOfUnassignedVariables, ExamPerStudentMinMax * GreedyIndependentSetSD, ExamPerStudentMeanMax * GreedyIndependentSetVC, NrOfConflictMean / GreedyNumberOfUnassigned Variables, NrOfConflictVarCoef + TotalExamCap / TotalRoomAndNrPeriods, NrOfConflictSkew * TotalRoomCapacity, NrOfConflictSkew + TotalRoomCapacity, OneRoomOverTotalExams + GreedyNumberOfUnassignedVariables, AllRoomExams / GreedyIndependentSetMax, AllRoomExams / MinMembers, RoomsPerExam5Per * CCmean, RoomsPerExam5Per * WCCMinMax, RoomsPerExamVarCoef * NrOfPeriodsWithPenalty, PeriodPerExam5Per * GreedyIndependentSetVC, PeriodPerExamMean * RoomForAllExamsWithoutPenalty, PeriodPerExamVarCoef + TotalExamCap / TotalRoomAndNrPeriods, PeriodPerExamVarCoef + GreedyIndependentSetVC, PeriodPerExamSD + TotalExamCap /TotalRoomAndNrPeriods, ExamLengthMax / TotalRoomCapacity, ExamLengthMean + ExamPerPeriodSD, ConflictGraphDensity + NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, RoomCapacityMed * CliqueSizemean, RoomCapacityMean * WCCmean, TotalRoomCapacity / ExamPerRoomSD, TotalRoomCapacity - ExamPerRoomSD, TotalRoomCapacity + WCC95th, TotalExamCap /TotalRoomAndNrPeriods * RoomForAllExamsWithoutPenaltyOverTotal, TotalExamCap/TotalRoomAndNrPeriods * NrOfPeriodsWithPenalty, TotalExamCap / TotalRoomAndNrPeriods + ExamPerPeriodVarCoef, TotalExamCap / TotalRoomAndNrPeriods / Ccmed, TotalExamCap/TotalRoomAndNrPeriods / CCFirstQuart, TotalExamCap / TotalRoomAndNrPeriods - CCEntropy, TotalExamCap / TotalRoomAndNrPeriods / GreedyNumberOfUsedColors, TotalExamCap /TotalRoomAndNrPeriods / GreedyNumberOfUsedColorsPerPeriod, MaxRoomPenalty / GreedyNumberOfUsedColors, ExamPerRoomMean / CliqueSizemean, ExamPerPeriodVarCoef / CC95th, ExamPerPeriodVarCoef + CCEntropy, CC5th / GreedyIndependentSetMinMax, CCThird * GreedyIndependentSetVC, WCCMinMax - ComputationalTime, CliqueSizeFirstQuart / GreedyNumberOfUnassignedVariables, CliqueSizeThird / GreedyIndependentSet, MinMax, GreedyNumberOfUnassignedVariables / GreedyIndependentSetVC

Table A.14: The features selected by different method on the discretized version of the `SA_2class` dataset

FS method	Selected features
D + RFE (109)	NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, NumberOfExclusionConstraints, NumberOfRoomExclusiveConstraints, NumberOfTimeRelatedConstraints, TotalNumberOfConstraints, TimeRelatedFractionOfConstraints, RoomRelatedFractionOfConstraints, WeightTwoInTheRow, WeightTwoInADay, WeightPeriodSpread, WeightMixedDuration, WeightNumberOfLargeExams, FrontLoadPenalty, FrontLoadWeight, StudentPerExamMax, StudentPerExamMin, StudentPerExamMean, StudentPerExamMed, StudentPerExamVarCoef, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, StudentPerExamSkew, StudentPerExamMinMax, StudentPerExamMeanMax, ExamsPerStudentMax, ExamPerStudent5thPer, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentVarCoef, ExamPerStudentSD, ExamPerStudent1Quant, ExamPerStudent3Q, ExamPerStudentEntropy, ExamPerStudentSkew, ExamPerStudentMinMax, ExamPerStudentMeanMax, NrOfConflictMax, NrOfConflict5thP, NrOfConflictMean, NrOfConflictMed, NrOfConflictVarCoef, NrOfConflictSD, NrOfConflict1Q, NrOfConflict3Q, NrOfConflictSkew, NrOfConflictMinMax, RoomsPerExam5Per, AllPeriodExamOverTotal, PeriodPerExam5Per PeriodPerExamVarCoef, PeriodPerExamSD, ConflictGraphDensity, TotalExamCap/TotalRoomAndNrPeriods, ExamPerRoomMed, ExamPerRoomVarCoef, ExamPerRoomSD, RoomForAllExamsWithoutPenalty, RoomForAllExamsWithoutPenaltyOverTotal, PeriodLengthMax, PeriodLengthMin, PeriodLengthMed, PeriodLengthMean, PeriodPenaltyMax, NrOfPeriodsWithPenalty, ExamPerPeriodMax, ExamPerPeriodMin, ExamPerPeriodMean, ExamPerPeriodMed, ExamPerPeriodVarCoef, ExamPerPeriodSD, NrOfPeriodsForAllExamsWithoutPenalty, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, AvNrOfPeriodPerDay, CC95th, CC5th, CCmean, Ccmed, CCVarCoef, CCSD, CCFirstQuart, CCThird CCEntropy, CCSkew, CCMinMax, CCMeanMax, WCC95th, WCC5th, WCCmean, WCCmed, WCCVarCoef, WCCSD, WCCFirstQuart, WCCEntropy, WCCSkew, WCCMinMax, WCCMeanMax, CliqueSizeVarCoef, CliqueSizeThird, GreedyNumberOfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyNumberOfUnassignedVariables, GreedyScore, GreedyIndependentSetMedian, GreedyIndependentSetVC, GreedyIndependentSetSD
D+BFS (13)	NumberOfExamsOverSlack, NumberOfAfterConstraints, NumberOfCoincidenceConstraints, StudentPerExamSD, ExamPerStudent3Q, TotalRoomCapacity, TotalExamCap / TotalRoomAndNrPeriods, ExamPerPeriodVarCoef, ExamPerPeriodSD, GreedyNumber OfUsedColors, GreedyNumberOfUsedColorsPerPeriod, GreedyIndependentSetVC, GreedyIndependentSetSD
Ext+D+RFE (78)	NumberOfCoincidenceConstraints / ExamOverRoomMaxCapacity, NumberOfCoincidenceConstraints / ExamOverRoomMeanCapacity, NumberOfCoincidenceConstraints / NrRoomsWithPenalty, NumberOfCoincidenceConstraints / ExamPerRoomMax, NumberOfCoincidenceConstraints / ExamPerRoomMin, NumberOfCoincidenceConstraints / ExamPerRoomMean, NumberOfCoincidenceConstraints ⋆ RoomForAllExamsWithoutPenalty, NumberOfCoincidenceConstraints / PeriodLengthMax, NumberOfCoincidenceConstraints − PeriodLengthMed, NumberOfCoincidenceConstraints / NrOfPeriodsWithPenalty, NumberOfCoincidenceConstraints / ExamPerPeriodMax, NumberOfCoincidenceConstraints / ExamPerPeriodVarCoef, NumberOfCoincidenceConstraints / ExamPerPeriodSD, NumberOfCoincidenceConstraints + CC95th, NumberOfCoincidenceConstraints ⋆ CCSD, NumberOfCoincidenceConstraints / CCEntropy, NumberOfCoincidenceConstraints / CliqueSizeVarCoef, NumberOfCoincidenceConstraints − CliqueSizeFirstQuart, NumberOfCoincidenceConstraints + CliqueSizeFirstQuart, NumberOfCoincidenceConstraints + CliqueSizeThird, NumberOfCoincidenceConstraints − GreedyNumberOfUsedColors, NumberOfCoincidenceConstraints + GreedyNumberOfUsedColorsPerPeriod, NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints / GreedyScore, NumberOfCoincidenceConstraints / GreedyIndependentSetMax, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetVC, WeightPeriodSpread ⋆ TotalExamCap / TotalRoomAndNrPeriods, FrontLoadPenalty ⋆ TotalExamCap / TotalRoomAndNrPeriods, StudentPerExamMed − RoomForAllExamsWithoutPenaltyOverTotal, StudentPerExamMed − GreedyNumberOfUsedColorsPerPeriod, StudentPerExamEntropy ⋆ RoomsPerExamSD, ExamsPerStudentMax / TotalExamCap / TotalRoomAndNrPeriods, RoomsPerExam95P + GreedyNumberOfUsedColors, RoomsPerExamSD − PeriodPerExamMean, RoomCapacityMean − TotalExamCap / TotalRoomAndNrPeriods, TotalRoomCapacity / TotalExamCap/TotalRoomAndNrPeriods, TotalExamCap / TotalRoomAndNrPeriods + ExamPerPeriodVarCoef, TotalExamCap / TotalRoomAndNrPeriods + GreedyIndependentSetMinMax, ExamPerRoomMax ⋆ GreedyNumberOfUsedColors, ExamPerRoomMax / GreedyIndependentSetMedian, ExamPerRoomMin / GreedyIndependentSetMax, ExamPerRoomMed ⋆ GreedyNumberOfUsedColors, ExamPerRoomMed / GreedyIndependentSet3thQ ExamPerRoomSD / GreedyScore, ExamPerRoomSD − GreedyScore, RoomForAllExamsWithoutPenalty ⋆ GreedyScore, RoomForAllExamsWithoutPenaltyOverTotal / GreedyNumberOfUsedColors, RoomForAllExamsWithoutPenaltyOverTotal / GreedyNumberOfUsedColorsPerPeriod, RoomForAllExamsWithoutPenaltyOverTotal / GreedyScore, PeriodLengthMin − GreedyNumberOfUsedColors, PeriodLengthMed + GreedyNumberOfUsedColors, ExamPerPeriodMax ⋆ GreedyNumberOfUsedColors, ExamPerPeriodMax / GreedyIndependentSetMax, ExamPerPeriodMed / GreedyNumberOfUnassignedVariables, ExamPerPeriodSD / GreedyIndependentSetSD, ExamPerPeriodSD −GreedyIndependentSetSD, NrOfPeriodsForAllExamsWithoutPenalty − GreedyNumberOfUsedColorsPerPeriod, AvNrOfPeriodPerDay ⋆ GreedyNumberOfUsedColors, CC95th ⋆ GreedyNumberOfUsedColors, CC95th − GreedyNumberOfUsedColors, CC95th − GreedyNumberOfUsedColorsPerPeriod, CC95th + GreedyNumberOfUsedColorsPerPeriod, CC95th + GreedyScore, CC95th ⋆ GreedyIndependentSetVC, CCVarCoef ⋆ GreedyNumberOfUsedColors, CCSD ⋆ GreedyNumberOfUsedColors, CCFirstQuart / GreedyNumberOfUsedColors, CCskew ⋆ GreedyScore, WCC95th ⋆ GreedyNumberOfUsedColors, WCC5th / GreedyNumberOfUsedColors, WCC5th − GreedyNumberOfUnassignedVariables, WCCmean ⋆ GreedyScore, WCCmed ⋆ GreedyScore, WCCVarCoef / GreedyNumberOfUsedColorsPerPeriod, WCCVarCoef ⋆ GreedyNumberOfUnassignedVariables, WCCSkew ⋆ GreedyNumberOfUsedColorsPerPeriod, CliqueSizeMax − GreedyNumberOfUsedColors, CliqueSizeMax + GreedyNumberOfUsedColorsPerPeriod
Ext+D+BFS (68)	NumberOfStudents / TotalRoomCapacity, NumberOfStudents − TotalRoomCapacity, NumberOfExams / NumberOfCoincidenceConstraints, NumberOfRooms + GreedyNumberOfUsedColors NumberOfDays − GreedyNumberOfUsedColors, NumberOfExamsOverSlack / NumberOfCoincidenceConstraints, NumberOfExamsOverSlack / NumberOfTimeRelatedConstraints, NumberOfExamsOverSlack / RoomsPerExam5Per , NumberOfExamsOverSlack − AllPeriodExamOverTotal, NumberOfExamsOverSlack + PeriodPerExamSD, NumberOfExamsOverSlack / ConflictGraphDensity, NumberOfExamsOverSlack ⋆ NrOfPeriodsWithPenalty, NumberOfExamsOverSlack + ExamPerPeriodVarCoef, NumberOfExamsOverSlack / NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, NumberOfExamsOverSlack + VarCoefMembers, NumberOfAfterConstraints ⋆ GreedyNumberOfUsedColors, NumberOfCoincidenceConstraints − NrOfConflictMax, NumberOfCoincidenceConstraints / NrOneRoomExam, NumberOfCoincidenceConstraints / ExamLengthSD , NumberOfCoincidenceConstraints ⋆ ConflictGraphDensity, NumberOfCoincidenceConstraints / TotalExamCap / TotalRoomAndNrPeriods, NumberOfCoincidenceConstraints / MaxRoomPenalty, NumberOfCoincidenceConstraints ⋆ PeriodLengthMin, NumberOfCoincidenceConstraints / ExamPerPeriodSD, NumberOfCoincidenceConstraints ⋆ GreedyNumberOfUsedColorsPerPeriod, NumberOfCoincidenceConstraints / GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints − GreedyNumberOfUnassignedVariables, NumberOfCoincidenceConstraints − GreedyIndependentSet3thQ, NumberOfCoincidenceConstraints ⋆ GreedyIndependentSetMeanMax, NumberOfCoincidenceConstraints / MaxMembers, WeightPeriodSpread ⋆ TotalExamCap / TotalRoomAndNrPeriods, StudentPerExamSD + CliqueSizeSD, StudentPerExamEntropy ⋆ RoomsPerExamSD, ExamsPerStudentMax / GreedyNumberOfUnassigned Variables, ExamPerStudent5thPer / TotalExamCap / TotalRoomAndNrPeriods, ExamPerStudent5thPer / ExamPerPeriodSD, ExamPerStudentMean ⋆ NrOfConflictSkew, ExamPerStudentMean / GreedyNumberOfUnassignedVariables, ExamPerStudentVarCoef + TotalExamCap / TotalRoomAndNrPeriods, ExamPerStudentSkew / GreedyNumberOfUsedColorsPerPeriod, NrOfConflictMed − GreedyNumberOfUsedColors, NrOfConflictVarCoef + TotalExamCap / TotalRoomAndNrPeriods, NrOfConflictSkew ⋆ TotalRoomCapacity, NrOfConflictSkew + TotalRoomCapacity, NrOfConflictMinMax + TotalRoomCapacity, RoomsPerExam95P + GreedyNumberOfUsedColors, RoomsPerExam5Per / TotalExamCap / TotalRoomAndNrPeriods, RoomsPerExamSD − PeriodPerExamMean, AllPeriodExamOverTotal ⋆ GreedyIndependentSetVC, PeriodPerExamSD + TotalExamCap /TotalRoomAndNrPeriods, TotalRoomCapacity / ExamPerRoomMean, TotalRoomCapacity / ExamPerRoomSD, TotalRoomCapacity − ExamPerRoomSD, TotalRoomCapacity + Ccmed, TotalRoomCapacity + WCC95th, TotalExamCap / TotalRoomAndNrPeriods + ExamPerPeriodVarCoef, TotalExamCap / TotalRoomAndNrPeriods / Ccmed, TotalExamCap / TotalRoomAndNrPeriods / CCFirstQuart, TotalExamCap / TotalRoomAndNrPeriods / GreedyNumberOfUsedColors, ExamPerRoomMed / GreedyIndependentSetMax, ExamPerPeriodVarCoef / CC95th, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal ⋆ GreedyIndependentSetVC, WCCSD − Greedy IndependentSetVC, WCCMinMax − GreedyNumberofUsedColors, CliqueSizeFirstQuart / GreedyNumberofUnassignedVariables, CliqueSizeEntropy − GreedyIndependentSetVC GreedyNumberofUsedColors + GreedyIndependentSetFirstQ, GreedyNumberofUnassignedVariables / GreedyIndependentSetVC

Table A.15: The features selected by different method on the discretized version of the `ITC_2class` dataset

FS method	Selected features
S + RFE (34)	NumberOfStudents, Slackness, NumberOfExamsOverSlack, NumberOfCoincidenceConstraints, WeightPeriodSpread, StudentPerExamMean, StudentPerExamMed, StudentPerExamVarCoef, StudentPerExamSD, StudentPerExamEntropy, ExamPerStudentMean, ExamPerStudentSD, ExamPerStudentEntropy, ExamPerStudentSkew, NrOfConflictSD, RoomsPerExamMean, PeriodPerExamMean, PeriodPerExamMed, PeriodPerExamVarCoef, ConflictGraphDensity, RoomCapacityMax, RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, TotalExamCap/TotalRoomAndNrPeriods, ExamOverRoomMeanCapacity, MaxRoomPenalty, NrRoomsWithPenalty, ExamPerPeriodMean, ExamPerPeriodVarCoef, CliqueSizeSD, GreedyIndependentSetMean, GreedyIndependentSetVC, GreedyIndependentSetSD
S+BFS (31)	NumberOfStudents, Slackness, NumberOfExamsOverSlack, NumberOfCoincidenceConstraints, TotalNumberOfConstraints, FrontLoadPenalty, StudentPerExamMin, StudentPerExamMed, StudentPerExamVarCoef, StudentPerExamSD, ExamPerStudent5thPer, ExamPerStudentMean, NrOfConflictSkew, NrOfConflictMinMax, RoomsPerExamMean, PeriodPerExamMed, RoomCapacityMin, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, ExamPerRoomVarCoef, RoomForAllExamsWithoutPenalty, ExamPerPeriodMed, ExamPerPeriodVarCoef, CCSD CCThird, CCEntropy, CCMeanMax, GreedyScore, GreedyIndependentSetSD, GreedyIndependentSetMinMax, GreedyIndependentSetMeanMax
S+Ext+BFS (30)	NumberOfExamsOverSlack / StudentPerExamEntropy, NumberOfExamsOverSlack / CliqueSizeSD, NumberOfCoincidenceConstraints ⋆ RoomCapacityMax, NumberOfCoincidenceConstraints ⋆ RoomCapacityMean, NumberOfCoincidenceConstraints ⋆ MeanMembers, StudentPerExamMean + PeriodPerExamMean, StudentPerExamMean + PeriodPerExamMed, StudentPerExamMean − ExamLengthMin, StudentPerExamMean − MaxRoomPenalty, StudentPerExamMean ⋆ RoomForAllExamsWithoutPenalty, StudentPerExamSD − MaxRoomPenalty, StudentPerExam3Q − MaxRoomPenalty, ExamsPerStudentMax / TotalRoomCapacityMultNrOfPeriods, ExamPerStudentMean + RoomsPerExam5Per, ExamPerStudentMean − NrRoomsWithPenalty, ExamPerStudentSD + RoomForAllExamsWithoutPenaltyOverTotal, ExamPerStudentEntropy / TotalExamCap/TotalRoomAndNrPeriods, ExamPerStudentSkew ⋆ TotalExamCap/TotalRoomAndNrPeriods, ExamPerStudentMinMax ⋆ TotalRoomCapacityMultNrOfPeriods, NrOfConflictEntropy ⋆ TotalRoomCapacity, RoomsPerExamSD ⋆ GreedyScore, PeriodPerExamMean ⋆ GreedyIndependentSetMax, PeriodPerExamMean ⋆ GreedyIndependentSet3thQ, PeriodPerExamMed ⋆ GreedyIndependentSetMax, PeriodPerExamMed ⋆ GreedyIndependentSetSD, TotalRoomCapacity / GreedyNumberofUnassignedVariables, NrRoomsWithPenalty / GreedyScore, NrOfPeriodsForAllExamsWithoutPenalty ⋆ GreedyIndependentSetSD, WCCMeanMax − GreedyIndependentSetMeanMax, CliqueSizeThird + GreedyIndependentSetSD
D+RFE (51)	NumberOfRooms, Slackness, NumberOfExamsOverSlack, NumberOfCoincidenceConstraints, StudentPerExamMax, StudentPerExamMean, StudentPerExamSD, StudentPerExam1Q, StudentPerExam3Q, StudentPerExamEntropy, StudentPerExamMinMax, ExamPerStudentMean, ExamPerStudentMedian, ExamPerStudentEntropy, ExamPerStudentSkew, NrOfConflictMean, NrOfConflictSD, NrOfConflict3Q, RoomsPerExamMean, RoomsPerExamVarCoef, PeriodPerExam5Per, PeriodPerExamMean, PeriodPerExamMed, RatioNrOfConflictsOverExams, ConflictGraphDensity, RoomCapacityMin, RoomCapacityMed, RoomCapacityMean, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, ExamOverRoomMaxCapacity, RoomForAllExamsWithoutPenaltyOverTotal, ExamPerPeriodMean, ExamPerPeriodSD, NrOfPeriodsForAllExamsWithoutPenalty, NrOfPeriodsForAllExamsWithoutPenaltyOverTotal, WCC95th, WCCentropy, WCCMeanMax, CliqueSizeMax, CliqueSizeThird, GreedyNumberofUsedColorsPerPeriod, GreedyIndependentSetMax, GreedyIndependentSetMean, GreedyIndependentSetVC, GreedyIndependentSetSD, GreedyIndependentSetFirstQ, GreedyIndependentSet3thQ, GreedyIndependentSetEntropy, GreedyIndependentSetMinMax, MaxMembers
D+BFS (27)	Slackness, NumberOfExamsOverSlack, NumberOfCoincidenceConstraints, TotalNumberOfConstraints, StudentPerExamMean, StudentPerExamMed, StudentPerExamVarCoef, ExamPerStudent5thPer, ExamPerStudentSkew, NrOfConflictSkew, RoomsPerExam5Per, RoomsPerExamMean, RoomCapacityMin, TotalRoomCapacity, TotalRoomCapacityMultNrOfPeriods, ExamPerRoomVarCoef, ExamPerPeriodMed, ExamPerPeriodVarCoef, CCSD, CCThird, CCEntropy, WCCSD, CliqueSizeMin, GreedyScore, GreedyIndependentSetMax, GreedyIndependentSetMinMax, GreedyIndependentSetMeanMax
D+Ext+BFS (14)	NumberOfDays ⋆ GreedyIndependentSetMax, NumberOfExamsOverSlack − CliqueSizeSD, NumberOfCoincidenceConstraints ⋆ RoomCapacityMean, StudentPerExamMean + PeriodPerExamMed, StudentPerExam3Q − ExamLengthMin, StudentPerExam3Q − MaxRoomPenalty, ExamPerStudentSD + RoomForAllExamsWithoutPenaltyOverTotal, ExamPerStudentEntropy / TotalExamCap/TotalRoomAndNrPeriods, ExamPerStudentMinMax ⋆ TotalRoomCapacityMultNrOfPeriods, PeriodPerExamMean ⋆ GreedyIndependentSet3thQ, PeriodPerExamMed ⋆ GreedyIndependentSetSD , TotalRoomCapacity / MaxRoomPenalty, TotalRoomCapacity / CCThird, TotalRoomCapacityMultNrOfPeriods ⋆ RoomForAllExamsWithoutPenalty

APPENDIX B

Regression And Classification Results

In this appendix, we provide the results obtained by various regression and classification methods on the datasets. The results on the training datasets have been obtained by running 10-fold cross-validation 10 times using the estimator with the best hyperparameter settings and recording a median value across the different cross-validation runs. The best parameter settings for each estimator for the dataset can be found in **??**. For testing the statistical significance between the results of various estimators the Welch's statistical test [Wel47] has been used. The threshold value used for the tests is p=0.01. The best results and the results that are not statistically significant from the best method are marked by bold.

B.1 Detailed Results For The Algorithm Performance Prediction Problem

Solver	Prepro cessing	Method	CC	RMSE log	MAE log	MedAElog	MAPE	MedAPE
`SACP`	O	LR	96.31	0.651	0.396	0.276	0.376	0.188
		Lasso	97.27	0.548	0.402	0.316	0.305	0.212
		Ridge	97.54	0.522	0.371	0.275	0.286	0.187
		ENet	97.33	0.54	0.392	0.3	0.297	0.2
		KNN	87.07	1.13	0.905	0.756	0.82	0.484
		SVR	38.06	1.691	1.44	1.402	1.382	0.771
		RF	92.99	0.865	0.678	0.559	0.555	0.374
		GBRF	**98.03**	**0.469**	**0.349**	**0.272**	**0.257**	**0.187**
		MLP	45.21	1.504	1.271	1.195	1.184	0.696
`SACP`	S	LR	96.11	0.667	0.434	0.310	0.52	0.208
		Lasso	97.76	0.497	0.358	0.277	0.27	0.18
		Ridge	97.71	0.503	0.36	0.281	0.274	0.186
		Enet	97.77	0.496	0.354	0.275	0.269	0.184
		KNN	93.74	0.836	0.67	0.589	0.491	0.369
		SVR	**98.4**	**0.422**	**0.3**	**0.227**	**0.228**	**0.152**
		RF	93	0.864	0.678	0.558	0.555	0.375
		GBRF	98	0.473	0.352	0.271	0.259	0.188
		MLP	96.56	0.439	0.33	0.253	0.245	0.178
`GRASP`	O	LR	97.11	0.58	0.422	0.325	0.326	0.218
		Lasso	97.15	0.57	0.424	0.337	0.326	0.223
		Ridge	**97.27**	**0.562**	**0.411**	**0.311**	**0.317**	**0.205**
		ENet	97.2	0.567	0.417	0.325	0.322	0.212
		KNN	86	1.194	0.932	0.787	0.922	0.495
		SVR	36.68	1.73	1.455	1.410	1.288	0.813
		RF	92.26	0.923	0.719	0.615	0.610	0.397
		GBRF	97.1	0.578	0.428	0.338	0.33	0.23
		MLP	51.02	1.475	1.234	1.119	1.133	0.714
`GRASP`	S	LR	66.34	5.636	0.675	0.323	0.328	0.213
		Lasso	97.58	0.526	0.381	0.28	0.294	0.193
		Ridge	97.58	0.541	0.393	0.294	0.304	0.2
		ENet	97.58	0.526	0.381	0.281	0.294	0.192
		KNN	93.5	0.915	0.741	0.656	0.511	0.399
		SVR	**97.9**	**0.493**	**0.34**	**0.244**	**0.271**	**0.171**
		RF	92.25	0.923	0.718	0.614	0.61	0.397
		GBRF	97.15	0.572	0.424	0.336	0.327	0.22
		MLP	94.88	0.544	0.401	0.309	0.305	0.218

Table B.1: Quantitative comparison of the regression models with the optimal parameters for the original and the standardized versions of the `ITC2007_time` training dataset

Solver	Prepro cessing	Method	CC	RMSE log	MAE log	MedAElog	MAPE	MedAPE
SACP	O	LR	96.91	0.594	0.387	0.275	0.316	0.189
		Lasso	97.28	0.549	0.404	0.309	0.307	0.208
		Ridge	97.61	0.518	0.369	0.276	0.283	0.183
		ENet	97.37	0.542	0.392	0.3	0.297	0.199
		KNN	87.16	1.131	0.906	0.767	0.823	0.476
		SVR	38.1	1,7	1,44	1,39	1.29	0.763
		RF	93,41	0.844	0.659	0.548	0.54	0.364
		GBRF	**98**	**0.474**	**0.353**	**0.275**	**0.26**	**0.189**
		MLP	48.18	1.48	1.251	1.164	1.168	0.69
SACP	S	LR	96.85	0.601	0.393	0.28	0.322	0.191
		Lasso	97.81	0.494	0.355	0.309	0.269	0.182
		Ridge	97.76	0.5	0.358	0.271	0.272	0.184
		ENet	97.82	0.493	0.351	0.27	0.266	0.179
		KNN	93.76	0.839	0.672	0.576	0.49	0.374
		SVR	**98.35**	**0.43**	**0.304**	**0.226**	**0.23**	**0.156**
		RF	93,4	0,845	0.659	0.546	0.54	0.363
		GBRF	98.99	0.476	0.352	0.272	0.26	0.188
		MLP	96.59	0.438	0.319	0.24	0.237	0.161
GRASP	O	LR	95.98	0.702	0.415	0.303	0.320	0.2
		Lasso	97.43	0.544	0.41	0.317	0.306	0.213
		Ridge	97.38	0.55	0.413	0.321	0.31	0.22
		ENet	97.46	0.541	0.406	0.313	0.303	0.216
		KNN	87.26	1.148	0.912	0.76	0.838	0.5
		SVR	38.38	1,731	1.474	1.448	1.317	0.778
		RF	93.24	0.87	0.683	0.57	0.559	0.375
		GBRF	**97.86**	**0.499**	**0.372**	**0.289**	**0.276**	**0.197**
		MLP	52.16	1.472	1.235	1.161	1.171	0.68
GRASP	S	LR	94.92	0.806	0.443	0.327	0.352	0.218
		Lasso	97.77	0.508	0.376	0.283	0.28	0.197
		Ridge	97.72	0.513	0.378	0.293	0.283	0.195
		ENet	97.78	0.507	0.371	0.279	0.278	0.385
		KNN	93.8	0.863	0.699	0.603	0.499	0.385
		SVR	**98.29**	**0.445**	**0.322**	**0.243**	**0.242**	**0.163**
		RF	93.26	0.87	0.683	0.57	0.558	0.374
		GBRF	97.87	0.497	0.372	0.285	0.275	0.196
		MLP	96.31	0.465	0.342	0.26	0.252	0.176
SA	O	LR	92.19	0.954	0.669	0.47	0.604	0.311
		Lasso	92.93	0.894	0.663	0.485	0.552	0.314
		Ridge	92.85	0.9	0.667	0.497	0.558	0.319
		ENet	92.82	0.902	0.668	0.5	0.561	0.315
		KNN	83.84	1.281	1.019	0.851	1.01	0.526
		SVR	38.33	1.75	1.49	1.456	1.403	0.769
		RF	90.73	1.021	0.782	0.621	0.679	0.408
		GBRF	**95.12**	**0.75**	**0.525**	**0.363**	**0.414**	**0.248**
		MLP	44.99	1.555	1.298	1.189	1.279	0.7
SA	S	LR	91.99	0.968	0.68	0.486	0.613	0.314
		Lasso	93.01	0.887	0.634	0.451	0.543	0.295
		Ridge	93.11	0.884	0.633	0.465	0.544	0.294
		ENet	93.11	0.884	0.627	0.456	0.542	0.295
		KNN	90.58	1.041	0.804	0.65	0.616	0.411
		SVR	**95.5**	**0.722**	**0.474**	**0.288**	**0.386**	**0.193**
		RF	90.72	1.022	0.782	0.615	0.68	0.406
		GBRF	95.08	0.753	0.527	0.364	0.416	0.246
		MLP	89.58	0.782	0.54	0.38	0.456	0.26

Table B.2: Quantitative comparison of the regression models with the optimal parameters for the original and the standardized versions of the SA_time training dataset

Solver	Preprocessing	Method	CC	RMSE log	MAE log	MedAElog	MAPE	MedAPE
SACP (Orig)	F	ENet	97.71	0.503	0.36	0.273	0.269	0.183
		SVR	98.57	0.4	0.283	0.195	0.206	0.136
		GBR	98.15	0.457	0.335	0.254	0.246	0.174
		MLP	**97.51**	**0.373**	**0.259**	**0.186**	**0.186**	**0.128**
	F + RFE	ENet	95.38	0.505	0.361	0.27	0.27	0.27
		SVR	95.53	0.498	0.362	0.273	0.269	0.273
		GBR	96.48	0.446	0.329	0.251	0.241	0.251
		MLP	**97.42**	**0.381**	**0.268**	**0.187**	**0.128**	**0.191**
	BDS	ENet	88.39	0.786	0.616	0.513	0.499	0.513
		SVR	**89.38**	**0.755**	**0.579**	**0.458**	**0.486**	**0.458**
		GBR	88.49	0.783	0.606	0.498	0.497	0.498
		MLP	73.8	1.292	1.0	0.812	0.507	0.936
SACP (Ext)	F+RFE	ENet	96.59	0.435	0.32	0.25	0.229	0.25
		SVR	**97.78**	**0.353**	**0.241**	**0.166**	**0.168**	**0.166**
		GBR	97.22	0.394	0.284	0.211	0.201	0.211
		MLP	97.01	0.41	0.292	0.206	0.145	0.209
	F+RFE 2	ENet	95.4	0.504	0.387	0.315	0.279	0.315
		SVR	**97.47**	**0.377**	**0.246**	**0.16**	**0.168**	**0.16**
		GBR	97.09	0.404	0.285	0.207	0.201	0.207
		MLP	96.45	0.449	0.289	0.175	0.121	0.215
	F+BDS	ENet	93.39	0.601	0.463	0.374	0.353	0.374
		SVR	**96.15**	**0.463**	**0.324**	**0.221**	**0.231**	**0.221**
		GBR	94.86	0.532	0.395	0.301	0.289	0.301
		MLP	93.33	0.617	0.411	0.264	0.18	0.29
GRASP (Orig)	F	ENet	97.68	0.516	0.376	0.294	0.29	0.2
		SVR	**98.16**	**0.461**	**0.324**	**0.238**	**0.243**	**0.164**
		GBR	97.45	0.543	0.399	0.3	0.303	0.206
		MLP	96.04	0.48	0.342	**0.23**	0.252	**0.165**
	F+RFE	ENet	95.35	0.516	0.376	0.294	0.289	0.294
		SVR	95.52	0.507	0.36	0.254	0.266	0.254
		GBR	94.54	0.559	0.414	0.321	0.314	0.321
		MLP	**95.8**	**0.495**	**0.353**	**0.256**	**0.177**	**0.259**
	BDS	ENet	88.26	0.805	0.625	0.512	0.519	0.512
		SVR	**88.9**	**0.785**	**0.593**	**0.465**	**0.522**	**0.465**
		GBR	87.54	0.828	0.635	0.522	0.544	0.522
		MLP	74.57	1.281	0.958	0.746	0.48	0.903
GRASP (Ext)	F+RFE	ENet	96.25	0.465	0.337	0.258	0.245	0.258
		SVR	**97.17**	**0.405**	**0.272**	**0.199**	**0.192**	**0.199**
		GBR	96.42	0.456	0.33	0.246	0.238	0.246
		MLP	96.6	0.446	0.301	0.208	0.145	0.217
	F+RFE2	ENet	95.1	0.53	0.401	0.323	0.294	0.323
		SVR	**97.29**	**0.396**	**0.271**	**0.174**	**0.187**	**0.174**
		GBR	96.32	0.461	0.33	0.236	0.238	0.236
		MLP	96.13	0.48	0.322	0.215	0.149	0.232
	F+BDS	ENet	93.43	0.61	0.469	0.381	0.362	0.381
		SVR	**94.62**	**0.555**	**0.411**	**0.307**	**0.316**	**0.307**
		GBR	94.58	**0.556**	0.417	0.327	0.311	0.327
		MLP	93.52	0.625	0.432	0.294	0.403	0.335

Table B.3: Quantitative comparison of the regression models with the optimal parameters on the reduced versions of the standardized and the extended `ITC2007_time` train dataset

Solver	Preprocessing	Method	CC	RMSE log	MAE log	MedAElog	MAPE	MedAPE
`SACP` (Orig)	F	ENet	97.69	0.507	0.371	0.283	0.278	0.188
		SVR	98.61	0.395	0.281	0.207	0.205	0.143
		GBR	97.99	0.477	0.352	0.266	0.259	0.183
		MLP	**97.51**	**0.377**	**0.259**	**0.16**	**0.186**	**0.111**
	F+RFE	ENet	95.3	0.512	0.376	0.288	0.282	0.288
		SVR	80.23	1.091	0.847	0.652	0.714	0.652
		GBRF	96.33	0.455	0.336	0.26	0.246	0.26
		MLP	**97.64**	**0.366**	**0.261**	**0.189**	**0.128**	**0.186**
	BDS	ENet	88.34	0.791	0.62	0.506	0.509	0.506
		SVR	**88.98**	**0.771**	**0.593**	**0.473**	**0.498**	**0.473**
		GBR	87.98	0.802	0.62	0.52	0.512	0.52
		MLP	73.81	1.275	0.97	0.747	0.48	0.915
`SACP` (Ext)	F+RFE	ENet	96.28	0.456	0.342	0.272	0.245	0.272
		SVR	**97.92**	**0.343**	**0.226**	**0.148**	**0.156**	**0.148**
		GBR	96.78	0.426	0.302	0.221	0.217	0.221
		MLP	97.09	0.408	0.264	0.167	0.118	0.192
	F+RFE2	ENet	96.28	0.456	0.342	0.272	0.245	0.272
		SVR	**97.92**	**0.343**	**0.226**	**0.148**	**0.156**	**0.148**
		GBR	97.41	0.383	0.269	0.192	0.189	0.192
		MLP	96.74	0.434	0.282	0.173	0.118	0.205
	F+BDS	ENet	94.99	0.528	0.406	0.327	0.307	0.327
		SVR	**97.46**	**0.379**	**0.263**	**0.178**	**0.187**	**0.178**
		GBR	95.65	0.493	0.356	0.261	0.258	0.261
		MLP	94.48	0.564	0.36	0.234	0.161	0.275
`GRASP` (Orig)	F	ENet	97.77	0.507	0.373	0.285	0.278	0.189
		SVR	**98.52**	**0.411**	**0.292**	**0.215**	**0.212**	**0.148**
		GBR	97.82	0.505	0.378	0.29	0.277	0.2
		MLP	97.06	0.415	0.293	**0.21**	0.208	**0.146**
	F+RFE	ENet	95.38	0.517	0.382	0.286	0.283	0.286
		SVR	96.4	0.458	0.337	0.259	0.24	0.259
		GBR	96.01	0.485	0.358	0.269	0.261	0.269
		MLP	**97.03**	**0.417**	**0.298**	**0.211**	**0.145**	**0.215**
	BDS	ENet	88.47	0.802	0.629	0.507	0.506	0.507
		SVR	**89.26**	**0.776**	**0.605**	**0.499**	**0.485**	**0.499**
		GBR	88.03	0.816	0.631	0.513	0.52	0.513
		MLP	74.46	1.3	1.009	0.8	0.515	0.976
`GRASP` (Ext)	F+RFE	ENet	96.18	0.471	0.353	0.274	0.254	0.274
		SVR	96	0.502	0.392	0.332	0.271	0.332
		GBR	96.77	0.436	0.312	0.229	0.221	0.229
		MLP	**97.3**	**0.4**	**0.269**	**0.185**	**0.127**	**0.191**
	F+RFE2	ENet	95.38	0.517	0.395	0.32	0.287	0.32
		SVR	**97.64**	**0.371**	**0.244**	**0.16**	**0.17**	**0.16**
		GBR	96.87	0.428	0.306	0.224	0.217	0.224
		MLP	95.92	0.497	0.325	0.202	0.142	0.246
	F+BDS	ENet	93.02	0.631	0.483	0.394	0.371	0.394
		SVR	**94.83**	**0.546**	**0.402**	**0.291**	**0.297**	**0.291**
		GBR	93.77	0.598	0.444	0.348	0.333	0.348
		MLP	90.64	0.753	0.538	0.391	0.267	0.415
`SA` (Orig)	F	ENet	93.84	0.836	0.602	0.431	0.494	0.286
		SVR	93.98	0.83	0.475	0.489	0.317	**0.136**
		GBR	**95.59**	**0.716**	**0.497**	**0.342**	**0.385**	**0.234**
		MLP	90.51	0.753	0.486	0.303	0.389	0.206
	F+RFE	ENet	87.68	0.836	0.602	0.431	0.494	0.431
		SVR	70.22	1.313	1.058	0.896	0.93	0.896
		GBR	**91.28**	**0.713**	**0.498**	**0.35**	**0.383**	**0.35**
		MLP	90.38	0.764	**0.492**	0.298	**0.205**	0.411
	BDS	ENet	72.12	1.205	0.913	0.714	0.765	0.714
		SVR	**76.25**	**1.129**	**0.834**	**0.632**	**0.705**	**0.632**
		GBR	76.23	1.131	0.858	0.662	0.753	0.662
		MLP	54.6	1.838	1.348	1.027	0.625	2.659
`SA`	F+RFE	ENet	87.7	0.84	0.598	0.435	0.468	0.435
		SVR	**94.17**	**0.588**	**0.346**	**0.177**	**0.242**	**0.177**
		GBR	92.88	0.645	0.422	0.269	0.32	0.269
		MLP	88.32	0.852	0.493	0.258	0.176	0.394
	F+RFE2	ENet	86.42	0.881	0.636	0.486	0.513	0.486
		SVR	**93.75**	**0.609**	**0.355**	**0.186**	**0.254**	**0.186**
		GBR	93.15	0.633	0.402	0.232	0.299	0.232
		MLP	88.31	0.861	0.497	0.232	0.163	0.489
	F+BDS	ENet	86.11	0.884	0.655	0.477	0.537	0.477
		SVR	**90.14**	**0.755**	**0.517**	**0.347**	**0.393**	**0.347**
		GBR	89.49	0.776	0.55	0.389	0.432	0.389
		MLP	81.22	1.103	0.708	0.431	0.293	0.657

Table B.4: Quantitative comparison of the regression models with the optimal parameters based on the reduced versions of the standardized and the extended `SA_time` training dataset

Solver	Prepro cessing	Method	CC	RMSE log	MAE log	MedAElog	MAPE	MedAPE
SACP_ITC	Orig	**SVR**	**95.96**	**0.45**	0.33	0.25	0.26	0.17
		GBR	94.41	0.53	0.41	0.32	0.31	0.23
	RFE	SVR	81.09	0.98	0.78	0.68	0.68	0.45
		GBR	**84.29**	**0.86**	0.69	0.58	0.61	0.4
	BFS	**SVR**	**87.03**	**0.8**	0.62	0.52	0.54	0.33
		GBR	86.38	0.81	0.64	0.53	0.56	0.35
SACP_ITC	F+RFE	**SVR**	**48.95**	**1.43**	1.19	1.09	1.24	0.67
(Ext)		GBR	46.38	1.46	1.21	1.07	1.28	0.67
	F+BFS	**SVR**	**95.34**	**0.49**	0.34	0.24	0.27	0.17
		GBR	93.93	0.56	0.42	0.33	0.32	0.22
GRASP_ITC	Orig	**SVR**	95.78	0.45	0.33	0.26	0.26	0.17
		GBR	93.33	0.56	0.43	0.34	0.33	0.24
	RFE	**SVR**	**87.93**	**0.75**	0.6	0.51	0.49	0.34
		GBR	85.68	0.81	0.64	0.55	0.54	0.36
	BFS	**SVR**	**88.74**	**0.73**	0.57	0.45	0.49	0.31
		GBR	86.69	0.79	0.63	0.52	0.52	0.35
GRASP_ITC	F+RFE	**SVR**	**80.64**	**0.93**	0.71	0.55	0.62	0.39
(Ext)		GBR	79.13	0.96	0.74	0.57	0.65	0.39
	F+BFS	**SVR**	**93.65**	**0.56**	0.43	0.34	0.33	0.24
		GBR	93.2	0.57	0.44	0.35	0.33	0.26
SACP_SA	Orig	**SVR**	**96**	**0.45**	0.33	0.26	0.26	0.17
		GBR	93.77	0.56	0.41	0.31	0.31	0.22
	RFE	SVR	78.62	1.01	0.8	0.65	0.72	0.43
		GBR	**81.49**	**0.93**	0.73	0.61	0.65	0.42
	BFS	**SVR**	**86.2**	**0.82**	0.63	0.51	0.57	0.33
		GBR	85.52	0.84	0.64	0.55	0.58	0.35
SACP_SA	F+RFE	SVR	82.31	0.91	0.7	0.6	0.65	0.41
(Ext)		**GBR**	**84.31**	**0.86**	0.68	0.6	0.56	0.38
	F+BFS	**SVR**	**96.89**	**0.4**	0.28	0.21	0.2	0.14
		GBR	93.81	0.55	0.4	0.3	0.31	0.21
GRASP_SA	Orig	**SVR**	**96.28**	**0.44**	0.33	0.25	0.24	0.16
		GBR	93.49	0.59	0.44	0.32	0.33	0.21
	RFE	SVR	82.66	0.94	0.75	0.64	0.65	0.42
		GBR	**85.41**	**0.85**	0.7	0.64	0.56	0.41
	BFS	**SVR**	**87.5**	**0.8**	0.62	0.49	0.51	0.33
		GBR	86.55	0.83	0.65	0.53	0.55	0.37
GRASP_SA	F+RFE	**SVR**	**79.89**	**0.9**	0.78	0.68	0.61	0.43
(Ext)		GBR	78.06	1.03	0.83	0.69	0.68	0.44
	F+BFS	**SVR**	**93.6**	**0.58**	0.43	0.33	0.32	0.23
		GBR	91.67	0.66	0.5	0.4	0.38	0.27
SA_SA	Orig	**SVR**	**89.65**	**0.73**	0.48	0.32	0.38	0.2
		GBR	88.86	0.74	0.51	0.35	0.39	0.23
	RFE	**SVR**	**73.19**	**1.14**	0.86	0.66	0.73	0.43
		GBR	72.62	**1.14**	0.86	0.7	0.72	0.46
	BFS	**SVR**	73.49	**1.11**	0.82	0.61	0.65	0.39
		GBR	**73.72**	**1.1**	0.86	0.67	0.73	0.44
SA_SA	F+RFE	**SVR**	**33.29**	**1.57**	1.31	1.21	1.29	0.71
(Ext)		GBR	25.49	1.67	1.37	1.22	1.51	0.72
	F+BFS	**SVR**	**87.89**	**0.78**	0.53	0.35	0.4	0.23
		GBR	87.88	0.8	0.55	0.38	0.45	0.25

Table B.5: Quantitative comparison of the regression models with the optimal parameters based on the test dataset

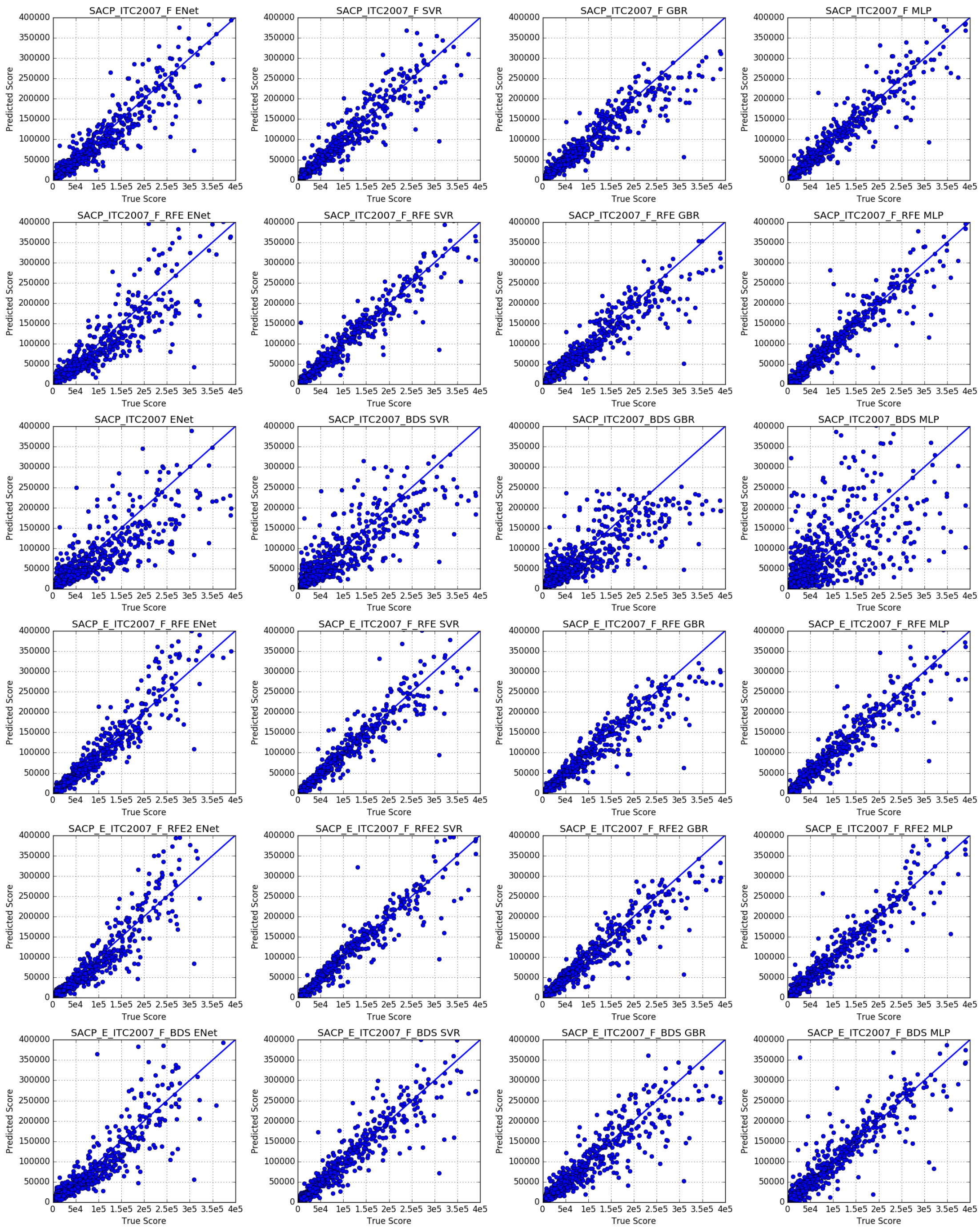

Figure B.1: Quantitative comparison of different regression models with optimal parameters on the reduced standardized versions of ITC2007time dataset for the SACP solver

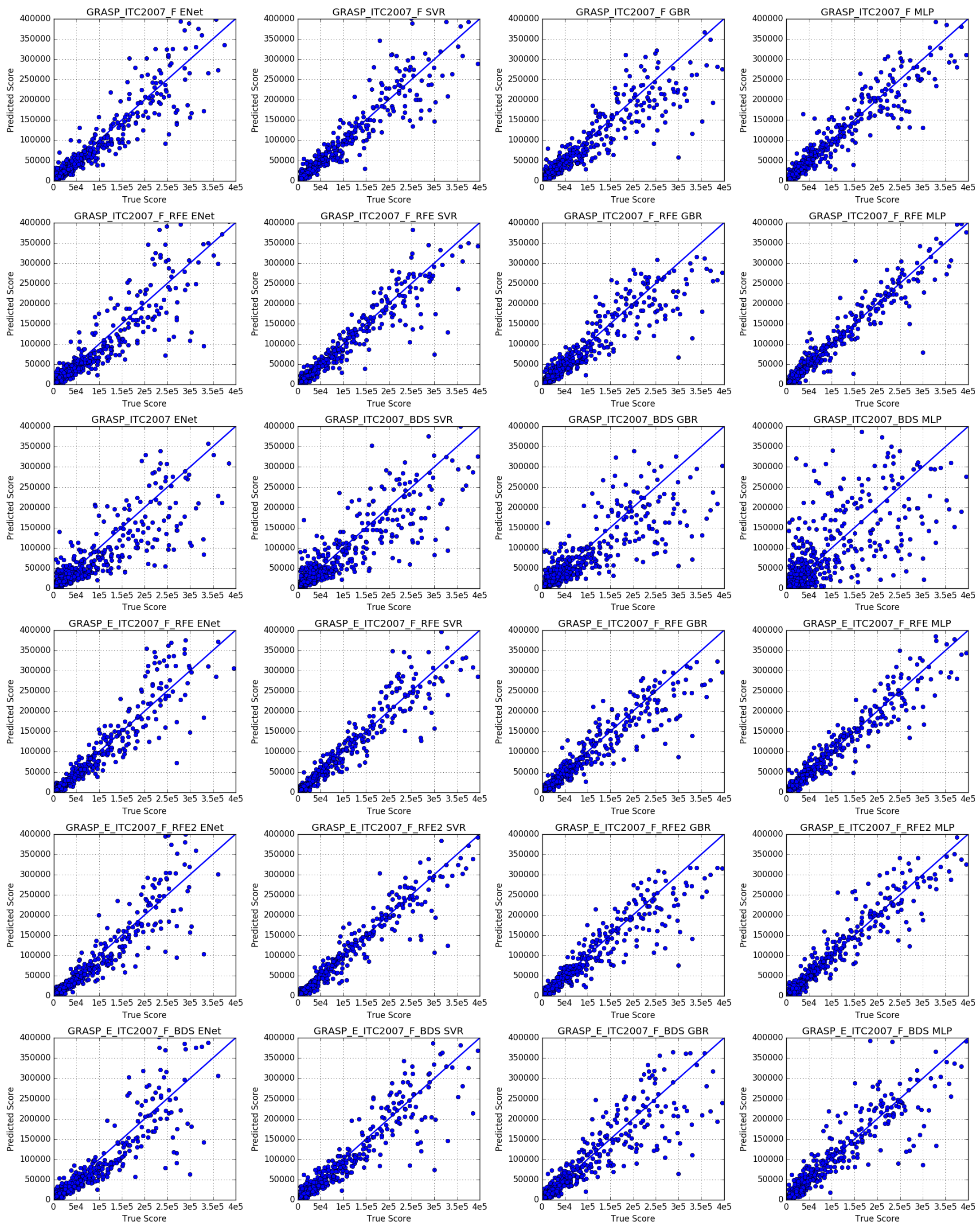

Figure B.2: Quantitative comparison of different regression models with optimal parameters on the reduced standardized versions of ITC2007time dataset for the GRASP solver

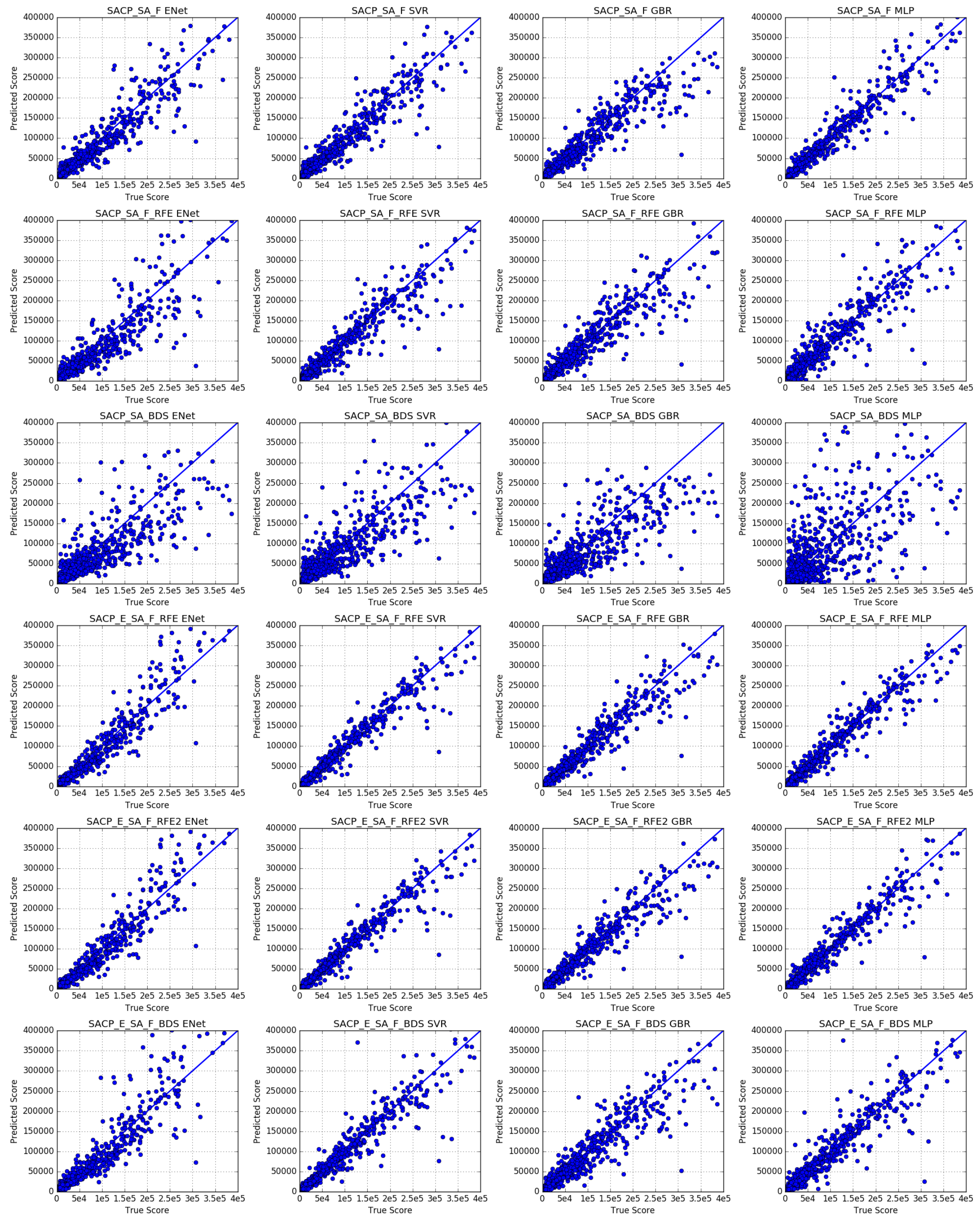

Figure B.3: Quantitative comparison of different regression models with optimal parameters on the reduced standardized versions of SA time dataset for the SACP solver

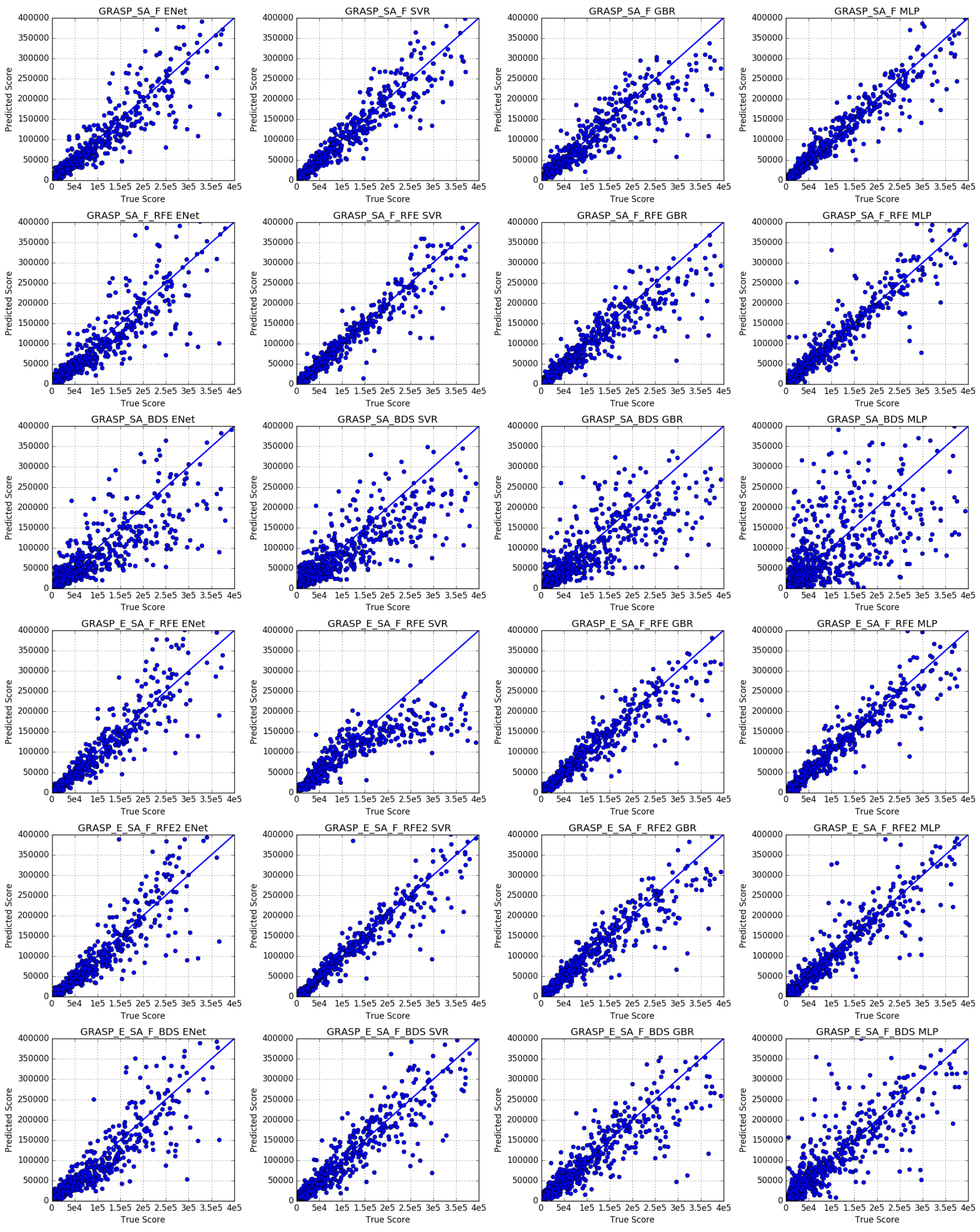

Figure B.4: Quantitative comparison of different regression models with optimal parameters on the reduced standardized versions of ITC2007time dataset for the GRASP solver

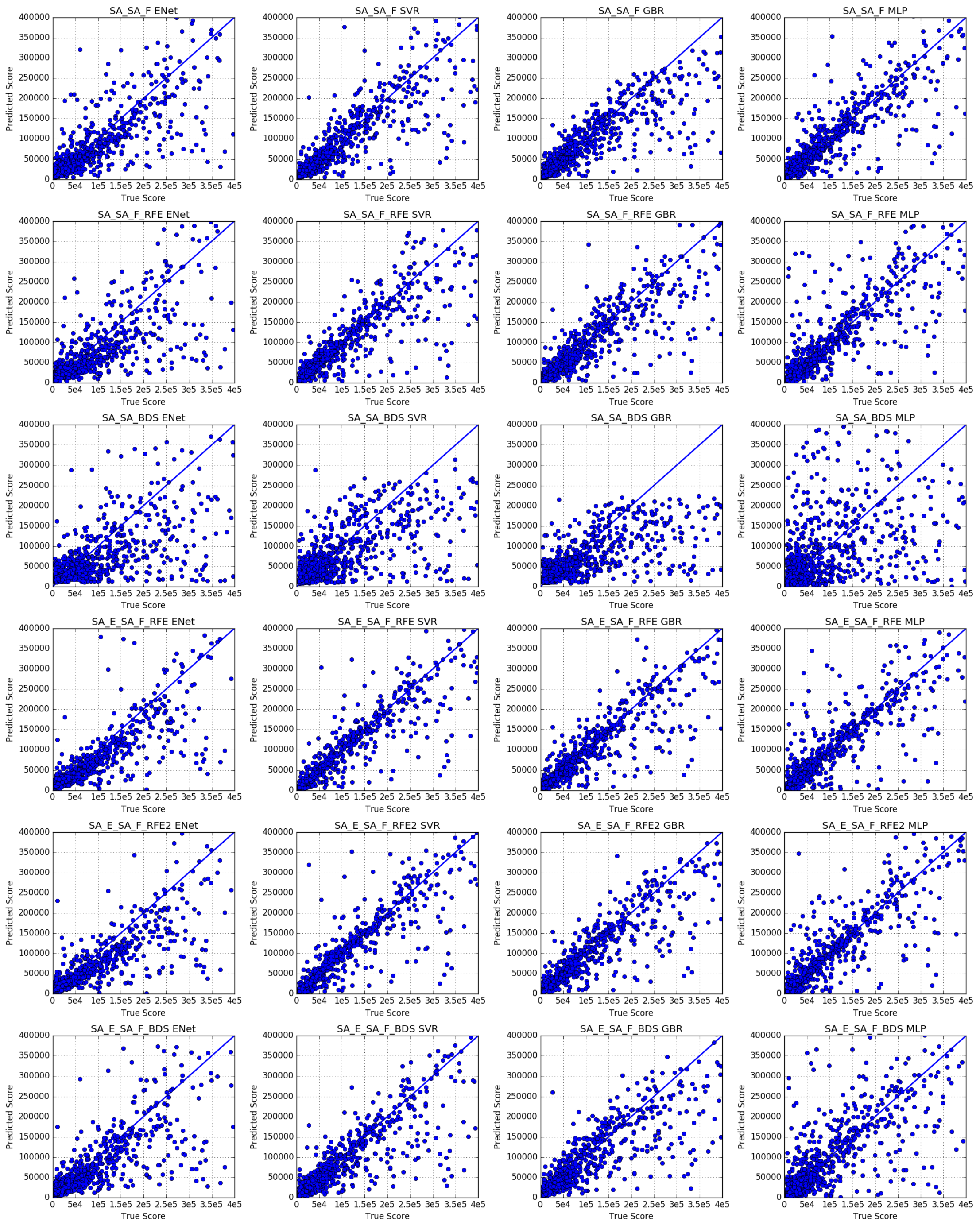

Figure B.5: Quantitative comparison of different regression models with optimal parameters on the reduced standardized versions of SA time dataset for the SA solver

B.2 Detailed Results for the Algorithm Selection Problem

Prepro cessing	Method	Accuracy	Log loss	Precision (macro)	Precision (micro)	Recall (macro)	Recall (micro)	F1 (macro)	F1 (micro)
O	Naive Bayes	49.44	12.48	49.19	49.44	54.16	49.44	49.44	43.83
	KNN	67.18	0.87	44.44	67.18	38.44	67.18	67.18	35.93
	SVC	63.94	0.85	21.31	63.94	33.33	63.94	63.94	26
	RF	64.51	0.82	55.24	64.51	61.39	64.51	64.51	54.43
	GBC	**73.24**	**0.65**	56.46	73.24	50.2	73.24	73.24	51.09
	MLP	65.92	0.77	41.32	65.92	36.99	65.92	65.92	33.76
S	Naïve Bayes	49.44	12.48	49.19	49.44	54.16	49.44	49.44	43.83
	KNN	64.93	0.76	39.05	64.93	35.41	64.93	64.93	30.9
	SVC	71.97	0.65	58.25	71.97	58.39	71.97	71.97	58.28
	RF	66.76	0.81	57.58	66.76	63.64	66.76	66.76	57.04
	GBC	**75.23**	**0.64**	63.58	74.23	51.28	74.23	74.23	52.85
	MLP	63.94	0.85	21.31	63.94	33.33	63.94	63.94	26
D	Naïve Bayes	55.07	11.54	53.56	55.07	56.76	55.07	55.07	45.88
	KNN	68.87	0.7	61.64	68.87	46.2	68.87	68.87	46
	SVC	68.59	0.62	58.67	68.59	65.24	68.59	68.59	59.84
	RF	65.49	0.8	56.43	65.49	64.2	65.49	65.49	56.37
	GBC	**75.94**	**0.61**	62.24	73.94	56.26	73.94	73.94	58.31
	MLP	73.66	0.63	62.26	73.66	55.3	73.66	73.66	56.73
S+RFE	SVC	28.87	0.86	33.18	28.87	33.3	28.87	28.87	26.34
	RF	67.46	0.71	45.42	67.46	38.19	67.46	67.46	35.3
	GBC	**73.52**	**0.66**	70.72	73.52	52.06	73.52	73.52	54.15
	MLP	67.46	0.74	43.84	67.46	41.31	67.46	67.46	40.38
S+BFS	SVC	34.51	0.86	33.28	34.51	33.4	34.51	34.51	29.48
	RF	70.85	0.68	45.82	70.85	43.66	70.85	70.85	42.65
	GBC	71.83	**0.64**	62.46	71.83	49.83	71.83	71.83	51.55
	MLP	**73.24**	**0.64**	63.51	73.24	50.51	73.24	73.24	51.14
Ext+S+RFE	SVC	63.94	0.85	21.31	63.94	33.33	63.94	63.94	26
	RF	72.39	0.65	45.97	72.39	46.32	72.39	72.39	45.27
	GBC	**76.34**	**0.6**	66.05	76.34	54.99	76.34	76.34	56.69
	MLP	70.7	0.74	58.22	70.7	51.59	70.7	70.7	53.52
Ext+S+BFS	SVC	63.94	0.85	21.31	63.94	33.33	63.94	63.94	26
	RF	71.55	0.67	46.02	71.55	44.81	71.55	71.55	43.87
	GBC	**76.48**	**0.6**	65.41	76.48	54.69	76.48	76.48	56.22
	MLP	69.44	0.76	43.45	69.44	44.29	69.44	69.44	43.14
D+RFE	SVC	67.89	0.61	58.83	67.89	66.82	67.89	67.89	59.27
	RF	68.03	0.69	40.31	68.03	41.41	68.03	68.03	39.79
	GBC	**73.38**	**0.6**	62.13	73.38	58.82	73.38	73.38	60.19
	MLP	71.27	1.16	58.18	71.27	54.39	71.27	71.27	55.78
D+BFS	**SVC**	70.85	**0.6**	61.93	70.85	71.06	70.85	70.85	63.44
	RF	67.04	0.69	39.15	67.04	40.31	67.04	67.04	38.54
	GBC	**73.52**	**0.6**	61.55	73.52	57.47	73.52	73.52	59.06
	MLP	73.24	0.61	60.11	73.24	53.54	73.24	73.24	55.05
Ext+D+RFE	**SVC**	71.97	**0.61**	60.47	71.97	65.5	71.97	71.97	61.91
	RF	71.27	0.66	44.03	71.27	46.81	71.27	71.27	45.22
	GBC	**75.63**	**0.61**	66.84	75.63	60.21	75.63	75.63	62.43
	MLP	72.25	0.63	60.93	72.25	55.64	72.25	72.25	57.53
Ext+D+BFS	**SVC**	**80.56**	**0.5**	74.07	80.56	65.64	80.56	80.56	68.74
	RF	74.79	0.62	49.7	74.79	47.96	74.79	74.79	47.37
	GBC	80.14	0.52	71.83	80.14	66.28	80.14	80.14	68.54
	MLP	80.14	0.53	72.42	80.14	68.19	80.14	80.14	69.98

Table B.6: Quantitative comparison of the classification algorithms on the training `SA_3_class` dataset

Prepro cessing	Method	Accuracy	ROC AUC	Precision	Recall	Specificity	NPV	F1 score
O	Naive Bayes	66.93	0.6139	50.58	44.67	78.1	73.78	47.45
	KNN	69.58	0.5661	67.11	17.53	95.69	69.81	27.79
	SVC	36.62	0.4985	33.33	89.69	10	65.91	48.6
	RF	76.35	**0.7385**	64.12	66.32	81.38	82.81	65.2
	GBC	**76.81**	0.6949	73.8	47.42	91.55	77.63	57.74
	MLP	71.41	0.6056	67.5	27.84	93.28	72.04	39.42
S	Naive Bayes	66.93	0.6139	50.58	44.67	78.1	73.78	47.45
	KNN	72.22	0.6313	65.41	35.74	90.52	73.74	46.22
	SVC	66.59	0.50	0	0	100	66.59	0
	RF	74.74	0.7401	60.23	71.82	76.21	84.35	65.52
	GBC	77.15	0.7103	71.5	52.58	89.48	79	60.59
	MLP	**78.76**	**0.7472**	70.54	62.54	86.9	82.22	66.3
D	Naive Bayes	68.77	0.5566	62.67	16.15	95.17	69.35	25.68
	KNN	76.46	0.70	70.67	50.52	89.48	78.28	58.92
	SVC	78.38	**0.7737**	63.2	77.32	77.41	87.18	69.55
	RF	73.36	0.7409	57.66	76.29	71.9	85.8	65.68
	GBC	80.19	0.7309	71.49	57.73	88.45	80.66	63.88
	MLP	**80.53**	0.7369	71.67	59.11	88.28	81.14	64.78
S+RFE	**SVC**	76.58	**0.76**	62.68	73.88	77.93	85.61	67.82
	RF	74.86	0.66	74	38.14	93.28	75.03	50.34
	GBC	77.15	0.72	70.35	54.64	88.45	79.53	61.51
	MLP	**77.73**	0.72	71.75	54.98	89.14	79.78	62.26
S+BFS	SVC	37.08	0.51	34	93.81	8.62	73.53	49.91
	RF	75.89	**0.75**	61.81	72.85	77.41	85.04	66.88
	GBC	**77.5**	0.72	70.93	55.33	88.62	79.81	62.16
	MLP	76	0.69	69.9	49.48	89.31	77.89	57.95
Ext+S+RFE	**SVC**	75.66	**0.73**	63.21	64.95	81.03	82.17	64.07
	RF	72.33	0.69	58.22	60.82	78.1	79.89	59.5
	GBC	**77.38**	0.71	71.56	53.61	89.31	79.33	61.3
	MLP	75.43	0.69	68.97	48.11	89.14	77.4	56.68
Ext+S+BFS	SVC	80.14	0.74	78.37	56.01	92.24	80.69	65.33
	RF	79.45	**0.78**	67.83	73.2	82.59	86	70.41
	GBC	**82.66**	**0.78**	80.43	63.57	92.24	83.46	71.02
	MLP	79.45	0.75	73.53	60.14	89.14	81.67	66.16
D+RFE	**SVC**	76.46	**0.77**	61.81	77.32	76.03	86.98	68.7
	RF	72.68	0.73	56.88	75.26	71.38	85.19	64.79
	GBC	78.3	0.73	71.98	57.39	88.79	80.59	63.86
	MLP	**78.53**	0.74	71.49	59.45	88.1	81.24	64.92
D+BFS	**SVC**	**79.68**	0.75	74.36	59.79	89.66	81.63	66.29
	RF	66.7	**0.8**	100	0.34	100	66.67	0.68
	GBC	77.5	0.72	70.21	56.7	87.93	80.19	62.74
	MLP	69.8	0.57	65.91	19.93	94.83	70.24	30.61
Ext+D+RFE	**SVC**	**81.06**	**0.8**	69.69	76.63	83.28	87.66	73
	RF	76.92	0.76	63.55	72.51	79.14	85.16	67.74
	GBC	80.71	0.77	74.31	64.6	88.79	83.33	69.12
	MLP	80.02	0.76	72.08	65.64	87.24	83.5	68.71
Ext+D+BFS	**SVC**	**84.27**	**0.83**	75.67	78.01	87.41	88.79	76.82
	RF	79.68	0.79	67.59	75.26	81.9	86.84	71.22
	GBC	84.16	0.81	78.44	72.51	90	86.71	75.36
	MLP	83.58	0.8	80.33	67.35	91.72	84.85	73.27

Table B.7: Quantitative comparison of different classification algorithms on the training `SA_2_class` dataset

Prepro cessing	Method	Accuracy	ROC AUC	Precision	Recall	Specificity	NPV	F1 score
O	Naive Bayes	73.31	0.7974	96.44	67.45	92.04	46.95	79.38
	KNN	78.65	0.7214	87.02	84.58	59.7	54.79	85.78
	SVC	76.16	0.50	76.16	100	0	86.46	
	RF	78.88	**0.8289**	96.22	75.23	90.55	53.37	84.44
	GBC	**84.27**	0.7518	87.78	90.65	59.7	66.67	89.2
	MLP	80.66	0.6971	84.96	90.65	48.76	62.03	87.72
S	Naive Bayes	73.43	0.7914	95.63	68.22	90.05	47.01	79.64
	KNN	82.44	0.7087	85.29	92.99	48.76	68.53	88.97
	SVC	84.46	0.8177	92.23	86.92	76.62	64.71	89.49
	RF	78.65	**0.8308**	96.57	74.61	91.54	53.03	84.18
	GBC	84.93	0.7866	89.68	90.65	66.67	69.07	90.16
	MLP	**86.17**	0.783	89.35	91.43	65.17	70.43	90.38
D	Naive Bayes	77.7	0.7852	92.51	76.95	80.1	52.1	84.01
	KNN	80.78	0.7047	85.4	90.19	50.75	61.82	87.73
	SVC	75.8	0.8069	95.82	71.34	90.05	49.59	81.79
	RF	78.05	**0.8286**	96.73	73.68	92.04	52.26	83.64
	GBC	84.46	0.7767	89.13	90.65	64.68	68.42	89.88
	MLP	**84.58**	0.7894	90	89.72	68.16	67.49	89.86
S+RFE	**SVC**	83.87	**0.85**	95.83	82.4	88.56	61.17	88.61
	RF	84.46	0.78	89.25	90.5	65.17	68.23	89.87
	GBC	85.77	0.81	90.78	90.5	70.65	69.95	90.64
	MLP	**86.6**	0.82	91.65	90.65	73.63	71.15	91.15
S+BFS	**SVC**	80.43	**0.84**	95.95	77.57	89.55	55.56	85.79
	RF	**82.68**	0.75	88.04	89.41	61.19	64.4	88.72
	GBC	82.44	0.75	88	89.1	61.19	63.73	88.54
	MLP	**82.68**	0.76	88.27	89.1	62.19	64.1	88.68
Ext+S+RFE	**SVC**	76.51	**0.82**	96.84	71.5	92.54	50.41	82.26
	RF	**82.33**	0.74	87.41	89.72	58.71	64.13	88.55
	GBC	**82.33**	0.75	87.98	88.94	61.19	63.4	88.46
	MLP	79.72	0.69	84.68	89.56	48.26	59.15	87.06
Ext+S+BFS	**SVC**	81.61	**0.86**	97.65	77.73	94.03	56.93	86.56
	RF	84.58	0.78	89.26	90.65	65.17	68.59	89.95
	GBC	**85.29**	0.79	89.85	90.97	67.16	69.95	90.4
	MLP	84.58	0.78	89.63	90.19	66.67	68.02	89.91
D+RFE	SVC	83.99	0.76	88.24	91.12	61.19	68.33	89.66
	RF	80.55	**0.85**	97.61	76.32	94.03	55.43	85.66
	GBC	**85.41**	0.79	89.98	90.97	67.66	70.1	90.47
	MLP	84.93	0.78	89.55	90.81	66.17	69.27	90.18
D+BFS	**SVC**	80.19	**0.84**	96.66	76.64	91.54	55.09	85.49
	RF	83.51	0.75	87.37	91.59	57.71	68.24	89.43
	GBC	**84.82**	0.8	90.41	89.56	69.65	67.63	89.98
	MLP	83.87	0.77	88.57	90.5	62.69	67.38	89.52
Ext+D+RFE	**SVC**	81.14	**0.84**	96.18	78.35	90.05	56.56	86.35
	RF	83.87	0.76	88.1	91.12	60.7	68.16	89.59
	GBC	85.41	0.79	90.11	90.81	68.16	69.9	90.46
	MLP	**86.48**	0.81	91.12	91.12	71.64	71.64	91.12
Ext+D+BFS	SVC	86	0.82	91.85	89.56	74.63	69.12	90.69
	RF	80.31	**0.85**	97.98	75.7	95.02	55.04	85.41
	GBC	**86.12**	0.81	91.34	90.34	72.64	70.19	90.84
	MLP	86	0.82	91.59	89.88	73.63	69.48	90.72

Table B.8: Quantitative comparison of the classification algorithms on the training `ITC_2_class` dataset

Prepro cessing	Method	Accuracy	Log loss	Precision (macro)	Precision (micro)	Recall (macro)	Recall (micro)	F1 (macro)	F1 (micro)
S	SVC	60.61	0.68	51.23	60.61	59.15	60.61	60.61	53.86
	RF	60.61	0.84	51.18	60.61	66.29	60.61	60.61	54.94
	GBC	**69.7**	**0.74**	67.35	69.7	58.93	69.7	69.7	61.85
	MLP	64.65	0.83	21.55	64.65	33.33	64.65	64.65	26.18
S+RFE	SVC	28.28	0.85	9.43	28.28	33.33	28.28	28.28	14.7
	RF	69.7	0.7	65.4	69.7	55.58	69.7	69.7	52.04
	GBC	**74.75**	**0.62**	64.83	74.75	63.54	74.75	74.75	62.67
	MLP	67.68	0.71	58.96	67.68	42.49	67.68	67.68	42.44
S+BFS	SVC	28.28	0.87	9.43	28.28	33.33	28.28	28.28	14.7
	RF	64.65	0.77	49.8	64.65	48.07	64.65	64.65	45.97
	GBC	67.68	0.74	54.84	67.68	52.98	67.68	67.68	52.4
	MLP	**73.74**	**0.62**	66.33	73.74	60.34	73.74	73.74	59.69
S+Ext+RFE	SVC	64.65	0.72	21.55	64.65	33.33	64.65	64.65	26.18
	RF	74.75	0.61	79.44	74.75	54.39	74.75	74.75	59.2
	GBC	**76.76**	**0.6**	70.3	75.76	66.29	75.76	75.76	62.17
	MLP	72.73	0.66	61.8	72.73	65.4	72.73	72.73	60.32
S+Ext+BFS	**SVC**	65.66	**0.69**	54.12	65.66	62.43	65.66	65.66	55.74
	RF	69.7	0.76	58.44	69.7	50.45	69.7	69.7	52.59
	GBC	**74.75**	0.71	65.56	74.75	58.63	74.75	74.75	60.42
	MLP	73.74	0.73	69.55	73.74	63.91	73.74	73.74	61.49
D	GBC	76.77	0.66	67.76	76.77	63.91	76.77	76.77	63.17
	RF	48.48	0.94	44.06	48.48	53.57	48.48	48.48	45.19
	GBC	**79.8**	**0.57**	72.74	79.8	70.39	79.8	79.8	69.17
	MLP	64.65	0.82	52.23	64.65	59.9	64.65	64.65	53.97
D+RFE	SVC	57.58	0.75	47.57	57.58	57.59	57.58	57.58	48.62
	RF	46.46	0.96	41.95	46.46	51.86	46.46	46.46	42.1
	GBC	67.68	**0.72**	56.48	67.68	57.89	67.68	67.68	56.18
	MLP	**73.74**	0.74	66.12	73.74	61.01	73.74	73.74	61.5
D+BFS	SVC	49.49	0.76	44.71	49.49	61.9	49.49	49.49	45.59
	RF	48.48	0.98	44.64	48.48	57.81	48.48	48.48	44.42
	GBC	**71.72**	**0.68**	62.29	71.72	66.22	71.72	71.72	62.61
	MLP	63.64	0.79	52.43	63.64	55.13	63.64	63.64	53.15
D+Ext+RFE	**SVC**	61.62	**0.71**	49.94	61.62	51.19	61.62	61.62	50.49
	RF	42.42	0.95	42.68	42.42	46.21	42.42	42.42	41.81
	GBC	66.67	0.76	56.78	66.67	53.12	66.67	66.67	54.24
	MLP	**73.71**	0.75	65.03	70.71	59.45	70.71	70.71	59.72
D+Ext+BFS	SVC	74.75	0.65	71.11	74.75	65.77	74.75	74.75	64.19
	RF	50.51	0.87	42.48	50.51	59.75	50.51	50.51	43.66
	GBC	**80.79**	**0.63**	73.16	78.79	75.45	78.79	78.79	72.46
	MLP	72.73	0.74	68.65	72.73	68.97	72.73	72.73	66.43

Table B.9: Quantitative comparison of the classification algorithms on the test `SA_3_class` dataset

Prepro cessing	Method	Accuracy	ROC AUC	Precision	Recall	Specificity	NPV	F1 score
S	**SVC**	**69.57**	0.5	0	0	100	69.57	0
	RF	53.26	0.52	32.56	50	54.69	71.43	39.44
	GBC	61.96	**0.59**	40	50	67.19	75.44	44.44
	MLP	**69.57**	0.5	0	0	100	69.57	0
S+RFE	**SVC**	**71.57**	**0.55**	100	9.38	100	70.71	17.14
	RF	67.39	0.49	25	3.57	95.31	69.32	6.25
	GBC	43.14	0.51	31.94	71.88	30	70	44.23
	MLP	**71.57**	**0.55**	100	9.38	100	70.71	17.14
S+BFS	SVC	30.43	0.5	30.43	100	0	46.67	
	RF	52.94	0.51	32.61	46.88	55.71	69.64	38.46
	GBC	66.67	0.57	45.45	31.25	82.86	72.5	37.04
	MLP	**71.57**	0.55	100	9.38	100	70.71	17.14
Ext+S+RFE	SVC	67.39	0.52	40	14.29	90.62	70.73	21.05
	RF	60.87	**0.6**	40	57.14	62.5	76.92	47.06
	GBC	**71.74**	**0.6**	57.14	28.57	90.62	74.36	38.1
	MLP	67.65	0.52	42.86	9.38	94.29	69.47	15.38
Ext+S+BFS	SVC	71.57	0.55	100	9.38	100	70.71	17.14
	RF	31.37	0.49	31	96.88	1.43	50	46.97
	GBC	**77.17**	**0.65**	81.82	32.14	96.88	76.54	46.15
	MLP	71.57	0.55	100	9.38	100	70.71	17.14
D	SVC	66.3	0.52	36.36	14.29	89.06	70.37	20.51
	RF	52.17	0.53	32.61	53.57	51.56	71.74	40.54
	GBC	**78.43**	**0.65**	81.25	40.62	95.71	77.91	54.17
	MLP	45.65	0.52	31.67	67.86	35.94	71.88	43.18
D+RFE	SVC	39.13	0.5	30.56	78.57	21.88	70	44
	RF	40.22	0.45	27.12	57.14	32.81	63.64	36.78
	GBC	60.87	**0.55**	36.67	39.29	70.31	72.58	37.93
	MLP	**69.57**	0.53	50	10.71	95.31	70.93	17.65
D+BFS	SVC	45.65	0.45	26.09	42.86	46.88	65.22	32.43
	RF	**69.57**	**0.5**	0	0	100	69.57	0
	GBC	47.06	**0.51**	32.26	62.5	40	70	42.55
	MLP	**69.57**	**0.5**	0	0	100	69.57	0
Ext+D+RFE	SVC	73.53	0.6	72.73	25	95.71	73.63	37.21
	RF	73.91	0.62	64.29	32.14	92.19	75.64	42.86
	GBC	76.47	0.62	100	25	100	74.47	40
	MLP	**77.66**	**0.69**	80	90.91	46.43	68.42	85.11
Ext+D+BFS	**SVC**	79.45	**0.81**	64.53	84.32	77.03	90.84	73.11
	RF	76.79	0.78	61.56	79.79	75.3	88.26	69.5
	GBC	**81.41**	0.77	76.03	64.11	89.98	83.49	69.57
	MLP	79.45	0.74	74.01	58.54	89.81	81.38	65.37

Table B.10: Quantitative comparison of the classification algorithms on the test `SA_2_class` dataset

Prepro cessing	Method	Accuracy	ROC AUC	Precision	Recall	Specificity	NPV	F1 score
S	**SVC**	**77.66**	**0.83**	97.87	69.7	96.43	57.45	81.42
	RF	74.47	0.78	92	69.7	85.71	54.55	79.31
	GBC	**77.66**	0.74	84.62	83.33	64.29	62.07	83.97
	MLP	**77.66**	0.68	79.22	92.42	42.86	70.59	85.31
S+RFE	SVC	58.51	0.61	80	54.55	67.86	38.78	64.86
	RF	79.79	0.75	85.07	86.36	64.29	66.67	85.71
	GBC	**81.91**	**0.75**	83.56	92.42	57.14	76.19	87.77
	MLP	**81.91**	**0.76**	84.51	90.91	60.71	73.91	87.59
S+BFS	**SVC**	76.6	**0.8**	94	71.21	89.29	56.82	81.03
	RF	71.28	0.67	80.95	77.27	57.14	51.61	79.07
	GBC	**81.91**	0.78	86.57	87.88	67.86	70.37	87.22
	MLP	79.79	0.69	79.75	95.45	42.86	80	86.9
Ext+S+RFE	**SVC**	74.47	**0.8**	95.65	66.67	92.86	54.17	78.57
	RF	73.4	0.66	78.87	84.85	46.43	56.52	81.75
	GBC	74.47	0.68	80.88	83.33	53.57	57.69	82.09
	MLP	**75.53**	0.64	77.22	92.42	35.71	66.67	84.14
Ext+S+BFS	**SVC**	**81.91**	**0.85**	96.23	77.27	92.86	63.41	85.71
	RF	77.66	0.73	83.58	84.85	60.71	62.96	84.21
	GBC	78.72	0.74	83.82	86.36	60.71	65.38	85.07
	MLP	78.72	0.71	81.94	89.39	53.57	68.18	85.51
D	SVC	75.53	0.67	79.45	87.88	46.43	61.9	83.45
	RF	74.47	0.74	86.21	75.76	71.43	55.56	80.65
	GBC	75.53	0.65	77.92	90.91	39.29	64.71	83.92
	MLP	**80.85**	**0.75**	84.29	89.39	60.71	70.83	86.76
D+RFE	**SVC**	79.79	**0.76**	86.15	84.85	67.86	65.52	85.5
	RF	74.47	**0.77**	90.38	71.21	82.14	54.76	79.66
	GBC	**80.85**	0.75	84.29	89.39	60.71	70.83	86.76
	MLP	77.66	0.69	80	90.91	46.43	68.42	85.11
D+BFS	**SVC**	73.4	**0.74**	87.27	72.73	75	53.85	79.34
	RF	71.28	0.63	77.46	83.33	42.86	52.17	80.29
	GBC	**75.53**	0.67	79.45	87.88	46.43	61.9	83.45
	MLP	74.47	0.73	85	77.27	67.86	55.88	80.95
Ext+D+RFE	**SVC**	76.6	**0.79**	92.31	72.73	85.71	57.14	81.36
	RF	74.47	0.65	78.38	87.88	42.86	60	82.86
	GBC	**78.72**	0.74	83.82	86.36	60.71	65.38	85.07
	MLP	77.66	**0.79**	90.91	75.76	82.14	58.97	82.64
Ext+D+BFS	SVC	75.53	0.61	75.29	96.97	25	77.78	84.77
	RF	73.4	0.77	91.84	68.18	85.71	53.33	78.26
	GBC	**84.04**	**0.78**	85.92	92.42	64.29	78.26	89.05
	MLP	76.6	0.65	77.5	93.94	35.71	71.43	84.93

Table B.11: Quantitative comparison of the classification algorithms on the test `ITC_2_class` dataset

APPENDIX C

The Optimal Parameters For Classification And Regression Models

This appendix contains the best parameters obtained for each estimator employed for the Performance Prediction and the Algorithm Selection problems. All parameters have been found for each solver/dataset separately based on the specified parameter range using 10-fold cross-validation using the training data. As the performance metrics for defining the best parameters for the estimator, `RMSE` has been chosen for regression problems, `ROC_AUC` for the binary classification problems, and `Log_Loss` for the multi-class problems.

For our experiments, we used different built-in functions in Scikit-learn, such as Greed Search (GS) for SVR/SVC and KNN methods that corresponds to the exhaustive enumeration. In the other case, when GS would be too time-consuming, Randomized Search (RS) with 90 iterations for RF, GBR/GBS and MLP has been used. For regularization methods we employed the method-specific techniques represented in Scikit-Learn, namely LARs for Lasso and Leave-One-Out for Ridge and ENet. All parameters and the parameters range tested can be found in the following tables. Each table corresponds to one method tested where regression and classification versions of the methods are considered as two different techniques.

C.1 The Optimal Parameters For The Regression Methods

Table C.1: The optimal parameters obtained on the training data using 10-fold cross-validation for Lasso and Ridge regression

Parameter	**Parameter range**	
alpha α	from 0 to 1	
Dataset	**Optimal parameters**	
SACP_ITC2007 (O)	Lasso	$\alpha = 0.005$
SACP_ITC2007 (O)	Ridge	$\alpha = 0.08$
SACP_ITC2007 (S)	Lasso	$\alpha = 0.005$
SACP_ITC2007 (S)	Ridge	$\alpha = 16.3$
GRASP_ITC2007 (O)	Lasso	$\alpha = 0.005$
GRASP_ITC2007 (O)	Ridge	$\alpha = 0.14$
GRASP_ITC2007 (S)	Lasso	$\alpha = 0.005$
GRASP_ITC2007 (S)	Ridge	$\alpha = 12.38$
SACP_SA (O)	Lasso	$\alpha = 0.005$
SACP_SA (O)	Ridge	$\alpha = 0.1$
SACP_SA (S)	Lasso	$\alpha = 0.005$
SACP_SA (S)	Ridge	$\alpha = 16.37$
GRASP_SA (O)	Lasso	$\alpha = 0.005$
GRASP_SA (O)	Ridge	$\alpha = 0.1$
GRASP_SA (S)	Lasso	$\alpha = 0.0066$
GRASP_SA (S)	Ridge	$\alpha = 12.38$
SA_SA (O)	Lasso	$\alpha = 0.02$
SA_SA (O)	Ridge	$\alpha = 616$
SA_SA (S)	Lasso	$\alpha = 0.005$
SA_SA (S)	Ridge	$\alpha = 28$

Table C.2: The optimal parameters obtained on the training data using 10-fold cross-validation for Elastic Net regression

Parameter	**Parameter range**

alpha α
l1_ratio
	from 0 to 1
	from 0.05 to 0.95 with step 0.05

Dataset	**Optimal parameters**
SACP_ITC2007 (O)	α=0.005, l1_ratio=0.05
SACP_ITC2007 (S)	α=0.01, l1_ratio=0.2
SACP_ITC2007_F(S)	α=0.005, l1_ratio=0.3
SACP_ITC2007_F_RFE(S)	α=0.005, l1_ratio=0.05
SACP_ITC2007_BDS(S)	α=0.061, l1_ratio=0.05
SACP_ITC2007_E_RFE(S)	α=0.005, l1_ratio=0.05
SACP_ITC2007_E_RFE2(S)	α=0.087, l1_ratio=0.05
SACP_ITC2007_E_BDS(S)	α=0.012, l1_ratio=0.4
GRASP_ITC2007 (O)	α=0.005, l1_ratio=0.05
GRASP_ITC2007 (S)	α=0.0066, l1_ratio=0.85
GRASP_ITC2007_F(S)	α=0.005, l1_ratio=0.05
GRASP_ITC2007_F_RFE(S)	α=0.05, l1_ratio=0.05
GRASP_ITC2007_BDS(S)	α=0.046, l1_ratio=0.05
GRASP_ITC2007_E_RFE(S)	α=0.005, l1_ratio=0.35
GRASP_ITC2007_E_RFE2(S)	α=0.0087, l1_ratio=0.05
GRASP_ITC2007_E_BDS(S)	α=0.015, l1_ratio=0.05
SACP_SA (O)	α=0.005, l1_ratio=0.05
SACP_SA (S)	α=0.005, l1_ratio=0.45
SACP_SA_F(S)	α=0.005, l1_ratio=0.3
SACP_SA_F_RFE(S)	α=0.005, l1_ratio=0.9
SACP_SA_BDS(S)	α=0.005, l1_ratio=0.5
SACP_SA_E_RFE(S)	α=0.005, l1_ratio=0.05
SACP_SA_E_RFE2(S)	α=0.005, l1_ratio=0.05
SACP_SA_E_BDS(S)	α=0.005, l1_ratio=0.05

Continued on next page

Dataset	**Optimal parameters**
GRASP_SA (O)	α=0.005, l1_ratio=0.25
GRASP_SA (S)	α=0.005, l1_ratio=0.3
GRASP_SA_F(S)	α=0.005, l1_ratio=0.05
GRASP_SA_F_RFE(S)	α=0.066, l1_ratio=0.6
GRASP_SA_BDS(S)	α=0.0152, l1_ratio=0.1
GRASP_SA_E_RFE(S)	α=0.0066, l1_ratio=0.05
GRASP_SA_E_RFE2(S)	α=0.0066, l1_ratio=0.05
GRASP_SA_E_BDS(S)	α=0.015, l1_ratio=0.95
SA_SA (O)	α=0.046, l1_ratio=0.4
SA_SA (S)	α=0.027, l1_ratio=0.1
SA_SA_F(S)	α=0.005, l1_ratio=0.05
SA_SA_F_RFE(S)	α=0.005, l1_ratio=0.05
SA_SA_BDS(S)	α=0.035, l1_ratio=0.05
SA_SA_E_RFE(S)	α=0.142, l1_ratio=0.3
SA_SA_E_RFE2(S)	α=0.108, l1_ratio=0.6
SA_SA_E_BDS(S)	α=0.0116, l1_ratio=0.05

Table C.3: The optimal parameters obtained on the training data using 10-fold cross-validation for the KNN regressor

Parameter	**Parameter range**
weights n_neighbors leaf_size metric p	uniform, distance 3, 5, 10, 15, 20, 25, 50, 75 5, 10, 15, 20, 25 euclidean, manhattan, chebyshev, minkowski 1, 2, 3
Dataset	**Optimal parameters**

SACP_ITC2007 (O)
SACP_ITC2007 (S)
	weights=distance, n_neighbors=20, leaf_size=5, metric=manhattan, p=1
	weights=distance, n_neighbors=10, leaf_size=5, metric=manhattan, p=1

GRASP_ITC2007 (O)
GRASP_ITC2007 (S)
	weights=distance, n_neighbors=20, leaf_size=5, metric=manhattan, p=1
	weights=distance, n_neighbors=15, leaf_size=5, metric=manhattan, p=1

SACP_SA (O)
SACP_SA (S)
	weights=distance, n_neighbors=20, leaf_size=5, metric=manhattan, p=1
	weights=distance, n_neighbors=10, leaf_size=5, metric=manhattan, p=1

GRASP_SA (O)
GRASP_SA (S)
	weights=distance, n_neighbors=20, leaf_size=5, metric=manhattan, p=1
	weights=distance, n_neighbors=15, leaf_size=5, metric=manhattan, p=1

SA_SA (O)
SA_SA (S)
	weights=distance, n_neighbors=20, leaf_size=5, metric=manhattan, p=1
	weights=distance, n_neighbors=10, leaf_size=5, metric=manhattan, p=1

Table C.4: The optimal parameters obtained on the training data using 10-fold cross-validation for the SVR

Parameter	**Parameter range**
gamma_range	from 2^{-15} to 2^{3}
C_range	from 2^{-15} to 2^{3}
epsilon	(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1.0)
Dataset	**Optimal parameters**
SACP_ITC2007 (O)	C=2.0, epsilon=1.0, gamma=3.0
SACP_ITC2007 (S)	C=128, epsilon=0.01, gamma=0.0005
SACP_ITC2007_F	C=2048.0, epsilon=0.1, gamma=0.00049
SACP_ITC2007_RFE	C=512, epsilon=0.01, gamma=0.0078
SACP_ITC2007_BFS	C=2.0, epsilon=0.3, gamma=0.0078
SACP_ITC2007_E_F_RFE	C=128, epsilon=0.001, gamma=0.00049
SACP_ITC2007_E_F_RFE2	C=32, epsilon=0.1, gamma=0.0078
SACP_ITC2007_E_F_BFS	C=2048, epsilon=0.1, gamma=0.0078
GRASP_ITC2007 (O)	C=2.0, epsilon=1.0, gamma=3.0
GRASP_ITC2007 (S)	C=128, epsilon=0.01, gamma=0.0005
GRASP_ITC2007_F	C=512.0, epsilon=0.1, gamma=0.00049
GRASP_ITC2007_RFE	C=32, epsilon=0.01, gamma=0.0078
GRASP_ITC2007_BFS	C=2.0, epsilon=0.3, gamma=0.0078
GRASP_ITC2007_E_F_RFE	C=128, epsilon=0.1, gamma=0.00049
GRASP_ITC2007_E_F_RFE2	C=32, epsilon=0.001, gamma=0.0078
GRASP_ITC2007_E_F_BFS	C=2, epsilon=0.3, gamma=0.0078
SACP_SA (O)	C=0.03, epsilon=0.7, gamma=3.0
SACP_SA (S)	C=128, epsilon=0.1, gamma=0.0005
SACP_SA_F	C=32, epsilon=0.1, gamma=0.0078
SACP_SA_RFE	C=8, epsilon=0.01, gamma=0.125
SACP_SA_BFS	C=8, epsilon=0.5, gamma=0.0078
SACP_SA_E_F_RFE	C=8, epsilon=0.01, gamma=0.0078
SACP_SA_E_F_RFE2	C=8, epsilon=0.01, gamma=0.0078
SACP_SA_E_F_BFS	C=512, epsilon=0.1, gamma=0.0078

Continued on next page

Dataset	**Optimal parameters**
GRASP_SA (O)	C=0.03, epsilon=0.7, gamma=3.0
GRASP_SA (S)	C=128, epsilon=0.1, gamma=0.0005
GRASP_SA_F	C=2048, epsilon=0.1, gamma=0.0005
GRASP_SA_RFE	C=512, epsilon=0.1, gamma=0.0078
GRASP_SA_BFS	C=8, epsilon=0.7, gamma=0.0078
GRASP_SA_E_F_RFE	C=512, epsilon=0.7, gamma=0.0078
GRASP_SA_E_F_RFE2	C=32, epsilon=0.01, gamma=0.0078
GRASP_SA_E_F_BFS	C=512, epsilon=0.3, gamma=0.0078
SA_SA (O)	C=8, epsilon=1.0, gamma=3.0
SA_SA (S)	C=128, epsilon=0.1, gamma=0.0005
SA_SA_F	C=32, epsilon=0.1, gamma=0.0078
SA_SA_RFE	C=8, epsilon=0.1, gamma=0.125
SA_SA_BFS	C=2, epsilon=0.5, gamma=0.125
SA_SA_E_F_RFE	C=32, epsilon=0.01, gamma=0.0078
SA_SA_E_F_RFE2	C=32, epsilon=0.1, gamma=0.0078
SA_SA_E_F_BFS	C=512, epsilon=0.01, gamma=0.0078

Table C.5: The optimal parameters obtained on the training data using 10-fold cross-validation for the RF Regressor

Parameter	Parameter range
max_features	from 0.05 to 1.0 with step 0.05
min_samples_leaf	from 3 to 50 with step 2
criterion	mse, mae
max_depth	from 3 to 50 with step 2
n_estimators	from 3 to 150 with step 2
min_samples_split	from 1 to 200 with step 5
Dataset	**Optimal parameters**
SACP_ITC2007 (O)	max_features=0.75, min_samples_leaf=40, criterion=mse, max_depth=45, n_estimators=115, min_samples_split=77
SACP_ITC2007 (S)	max_features=0.75, min_samples_leaf=40, criterion=mse, max_depth=45, n_estimators=115, min_samples_split=77
GRASP_ITC2007 (O)	max_features=0.7, min_samples_leaf=31, criterion=mse, max_depth=46, n_estimators=55, min_samples_split=75
GRASP_ITC2007 (S)	max_features=0.7, min_samples_leaf=31, criterion=mse, max_depth=46, n_estimators=155, min_samples_split=75
SACP_SA (O)	max_features=0.9, min_samples_leaf=28, criterion=mse, max_depth=69, n_estimators=150, min_samples_split=82
SACP_SA (S)	max_features=0.9, min_samples_leaf=28, criterion=mse, max_depth=69, n_estimators=150, min_samples_split=82
GRASP_SA (O)	max_features=0.65, min_samples_leaf=28, criterion=mse, max_depth=50, n_estimators=100, min_samples_split=102
GRASP_SA (S)	max_features=0.7, min_samples_leaf=28, criterion=mse, max_depth=47, n_estimators=120, min_samples_split=111

Continued on next page

Dataset	**Optimal parameters**
SA_SA (O)	max_features=0.8, min_samples_leaf=27, criterion=mse, max_depth=64, n_estimators=120, min_samples_split=112
SA_SA (S)	max_features=0.7, min_samples_leaf=27, criterion=mse, max_depth=60, n_estimators=89, min_samples_split=98

Table C.6: The optimal parameters obtained on the training data using 10-fold cross-validation for the GBR

Parameter	**Parameter range**
max_features	from 0.05 to 1.0 with step 0.05
min_samples_leaf	from 3 to 50 with step 2
criterion	mse, mase
max_depth	from 3 to 50 with step 2
n_estimators	from 3 to 150 with step 2
min_samples_split	from 1 to 200 with step 5
subsample	from 0.05 to 1.0 with step 0.05
learning_rate	0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.7
Dataset	**Optimal parameters**
SACP_ITC2007 (S)	max_features=0.95, min_samples_leaf=38, criterion=mse, max_depth=70, n_estimators=120, min_samples_split=99, subsample=0.55, learning_rate=0.2
SACP_ITC2007 (F)	max_features=0.8, min_samples_leaf=26, criterion=mse, max_depth=63, n_estimators=118, min_samples_split=55, subsample=0.7, learning_rate=0.1
SACP_ITC2007 (RFE)	max_features=1.0, min_samples_leaf=29, criterion=mse, max_depth=65, n_estimators=137, min_samples_split=96, subsample=0.75, learning_rate=0.1
SACP_ITC2007 (BFS)	max_features=0.85, min_samples_leaf=40, criterion=mse, max_depth=66, n_estimators=117, min_samples_split=210, subsample=0.9, learning_rate=0.05
SACP_ITC2007_Ext (RFE)	max_features=0.85, min_samples_leaf=29, criterion=mse, max_depth=52, n_estimators=118, min_samples_split=143, subsample=0.85, learning_rate=0.1

Continued on next page

Dataset	**Optimal parameters**
SACP_ITC2007_Ext (BFS)	max_features=0.4, min_samples_leaf=58, criterion=mse, max_depth=52, n_estimators=140, min_samples_split=143, subsample=0.85, learning_rate=0.2
GRASP_ITC2007 (S)	max_features=0.4, min_samples_leaf=62, criterion=mse, max_depth=54, n_estimators=86, min_samples_split=247, subsample=0.9, learning_rate=0.2
GRASP_ITC2007 (F)	max_features=0.25, min_samples_leaf=30, criterion=mse, max_depth=56, n_estimators=140, min_samples_split=151, subsample=0.8, learning_rate=0.2
GRASP_ITC2007 (RFE)	max_features=0.6, min_samples_leaf=47, criterion=mse, max_depth=43, n_estimators=140, min_samples_split=117, subsample=0.85, learning_rate=0.2
GRASP_ITC2007 (BFS)	max_features=0.85, min_samples_leaf=33, criterion=mse, max_depth=43, n_estimators=140, min_samples_split=191, subsample=0.55, learning_rate=0.1
GRASP_ITC2007_Ext (RFE)	max_features=0.75, min_samples_leaf=33, criterion=mse, max_depth=35, n_estimators=140, min_samples_split=142, subsample=0.85, learning_rate=0.2
GRASP_ITC2007 (BFS)	max_features=0.5, min_samples_leaf=29, criterion=mse, max_depth=52, n_estimators=140, min_samples_split=84, subsample=0.6, learning_rate=0.2
SACP_SA (S)	max_features=0.85, min_samples_leaf=43, criterion=mse, max_depth=46, n_estimators=144, min_samples_split=144, subsample=0.55, learning_rate=0.2
SACP_SA_F (S)	max_features=0.8, min_samples_leaf=47, criterion=mse, max_depth=42, n_estimators=139, min_samples_split=84, subsample=0.95, learning_rate=0.1

Continued on next page

Dataset	Optimal parameters
SACP_SA_RFE (S)	max_features=0.75, min_samples_leaf=47, criterion=mse, max_depth=53, n_estimators=139, min_samples_split=224, subsample=1.0, learning_rate=0.2
SACP_SA_BFS (S)	max_features=0.45, min_samples_leaf=44, criterion=mse, max_depth=63, n_estimators=102, min_samples_split=182, subsample=0.6, learning_rate=0.1
SACP_SA_Ext_RFE (S)	max_features=0.85, min_samples_leaf=30, criterion=mse, max_depth=58, n_estimators=143, min_samples_split=85, subsample=0.5, learning_rate=0.1
SACP_SA_Ext_BFS (S)	max_features=0.6, min_samples_leaf=32, criterion=mse, max_depth=70, n_estimators=118, min_samples_split=125, subsample=0.8, learning_rate=0.2
GRASP_SA (S)	max_features=0.8, min_samples_leaf=61, criterion=mse, max_depth=52, n_estimators=145, min_samples_split=220, subsample=0.95, learning_rate=0.1
GRASP_SA_F (S)	max_features=0.5, min_samples_leaf=64, criterion=mse, max_depth=59, n_estimators=125, min_samples_split=169, subsample=1.0, learning_rate=0.1
GRASP_SA_RFE (S)	max_features=0.55, min_samples_leaf=41, criterion=mse, max_depth=42, n_estimators=102, min_samples_split=156, subsample=0.8, learning_rate=0.1
GRASP_SA_BFS (S)	max_features=0.7, min_samples_leaf=35, criterion=mse, max_depth=73, n_estimators=90, min_samples_split=159, subsample=0.75, learning_rate=0.1
GRASP_SA_Ext_RFE (S)	max_features=0.4, min_samples_leaf=28, criterion=mse, max_depth=52, n_estimators=125, min_samples_split=171, subsample=0.9, learning_rate=0.1

Continued on next page

Dataset	**Optimal parameters**
GRASP_SA_Ext_BFS (S)	max_features=1.0, min_samples_leaf=55, criterion=mse, max_depth=30, n_estimators=143, min_samples_split=172, subsample=1.0, learning_rate=0.1
SA_SA (S)	max_features=0.8, min_samples_leaf=26, criterion=mse, max_depth=61, n_estimators=93, min_samples_split=140, subsample=0.6, learning_rate=0.1
SA_SA_F (S)	max_features=0.8, min_samples_leaf=33, criterion=mse, max_depth=69, n_estimators=143, min_samples_split=92, subsample=0.95, learning_rate=0.1
SA_SA_RFE (S)	max_features=0.9, min_samples_leaf=29, criterion=mse, max_depth=58, n_estimators=90, min_samples_split=141, subsample=1.0, learning_rate=0.2
SA_SA_BFS (S)	max_features=0.45, min_samples_leaf=31, criterion=mse, max_depth=64, n_estimators=130, min_samples_split=63, subsample=0.9, learning_rate=0.02
SA_SA_Ext_RFE (S)	max_features=1.0, min_samples_leaf=30, criterion=mse, max_depth=35, n_estimators=76, min_samples_split=113, subsample=1.0, learning_rate=0.1
SA_SA_Ext_RFE (S)	max_features=0.5, min_samples_leaf=52, criterion=mse, max_depth=36, n_estimators=130, min_samples_split=137, subsample=1.0, learning_rate=0.2

C.2 The Optimal Parameters For The Classification Methods

Table C.7: The optimal parameters obtained on the training data using 10-fold cross-validation for the KNN classifier

Parameter	**Parameter range**
weights	uniform, distance
n_neighbors	3, 5, 10, 15, 20, 25, 50, 75
leaf_size	5, 10, 15, 20, 25
metric	euclidean, manhattan, chebyshev, minkowski
p	1, 2, 3
Dataset	**Optimal parameters**
SA_3_class (O)	weights=distance, n_neighbors=50, leaf_size=5, metric=manhattan, p=1
SA_3_class (S)	weights=distance, n_neighbors=75, leaf_size=5, metric=minkowski, p=3
SA_3_class (D)	weights=uniform, n_neighbors=75, leaf_size=5, metric=euclidean, p=1
SA_2_class (O)	weights=distance, n_neighbors=50, leaf_size=5, metric=manhattan, p=1
SA_2_class (S)	weights=distance, n_neighbors=15, leaf_size=5, metric=euclidean, p=1
SA_2_class (D)	weights=uniform, n_neighbors=25, leaf_size=10, metric=minkowski, p=3
ITC2007_2_class (O)	weights=uniform, n_neighbors=50, leaf_size=5, metric=minkowski, p=3
ITC2007_2_class (S)	weights=distance, n_neighbors=25, leaf_size=5, metric=minkowski, p=3
ITC2007_2_class (D)	weights=distance, n_neighbors=20, leaf_size=5, metric=manhattan, p=1

Table C.8: The optimal parameters obtained on the training data using 10-fold cross-validation for the SVC

Parameter	**Parameter range**
gamma_range	from 2^{-15} to 2^{3}
C_range	from 2^{-15} to 2^{3}
class_weight	None, balanced
Dataset	**Optimal parameters**
SA_3_class (O)	C=128, gamma=0.0078125, class_weight=balanced
SA_3_class (S)	C=2, gamma=0.0078125, class_weight=balanced
SA_3_class (D)	C=8, gamma=0.0078125, class_weight=balanced
SA_3_class_FFS(S)	C=512, gamma=0.0078125, class_weight=balanced
SA_3_class_FFS(D)	C=8, gamma=0.0078125, class_weight=balanced
SA_3_class_RFE(S)	C=0.5, gamma=3.05-e05, class_weight=balanced
SA_3_class_RFE(D)	C=2.0, gamma=0.0078125, class_weight=balanced
SA_3_class_BFS(S)	C=0.03125, gamma=3.05-e05, class_weight=balanced
SA_3_class_BFS(D)	C=0.5, gamma=0.125, class_weight=balanced
SA_3_class_E_RFE(S)	C=2.0, gamma=0.125, class_weight=balanced
SA_3_class_E_RFE(D)	C=128.0, gamma=0.0005, class_weight=balanced
SA_3_class_E_BFS(S)	C=512.0, gamma=0.0078125, class_weight=None
SA_3_class_E_BFS(D)	C=2.0, gamma=0.0078125, class_weight=None
SA_2_class (O)	C=0.03125, gamma=3.05e-05, class_weight=(1:1000)
SA_2_class (S)	C=0.5, gamma=3.05e-05, class_weight=None

Continued on next page

Dataset	Optimal parameters
SA_2_class (D)	C=32, gamma=0.0078125, class_weight=balanced
SA_2_class_FFS (S)	C=0.5, gamma=3.05e-05, class_weight=balanced
SA_2_class_FFS (D)	C=128.0, gamma=0.0078125, class_weight=balanced
SA_2_class_RFE (S)	C=2, gamma=0.0078125, class_weight=balanced
SA_2_class_RFE (D)	C=128.0, gamma=0.0078125, class_weight=balanced
SA_2_class_BFS (S)	C=0.03125, gamma=3.05e-05, class_weight=balanced
SA_2_class_BFS (D)	C=8192, gamma=0.00049, class_weight=None
SA_2_class_E_RFE (S)	C=32, gamma=0.0078125, class_weight=balanced
SA_2_class_E_RFE (D)	C=8, gamma=0.0078125, class_weight=balanced
SA_2_class_E_BFS (S)	C=2, gamma=0.0078125, class_weight=None
SA_2_class_E_BFS (D)	C=8, gamma=0.0078125, class_weight=balanced
ITC_2_class (O)	C=0.5, gamma=3.05e-05, class_weight=balanced
ITC_2_class (S)	C=2.0, gamma=0.0078125, class_weight=balanced
ITC_2_class (D)	C=8.0, gamma=0.00049, class_weight=balanced
ITC_2_class_F (D)	C=32.0, gamma=0.00049, class_weight=None
ITC_2_class_RFE (S)	C=8.0, gamma=0.0078125, class_weight=balanced
ITC_2_class_RFE (D)	C=512.0, gamma=3.05e-05, class_weight=None
ITC_2_class_BFS (S)	C=8.0, gamma=0.0078, class_weight=balanced
ITC_2_class_BFS (D)	C=32.0, gamma=0.0078, class_weight=balanced
ITC_2_class_E_RFE (S)	C=32.0, gamma=0.00049, class_weight=balanced

Continued on next page

Dataset	Optimal parameters
ITC_2_class_E_RFE (D)	C=128.0, gamma=3.05e-05, class_weight=balanced
ITC_2_class_E_BFS (S)	C=512.0, gamma=3.05e-05, class_weight=balanced
ITC_2_class_E_BFS (D)	C=8.0, gamma=0.0078, class_weight=None

Table C.9: The optimal parameters obtained on the training data using 10-fold cross-validation for the RF classifier

Parameter	**Parameter range**
max_features	from 0.05 to 1.0 with step 0.05
min_samples_leaf	from 3 to 50 with step 2
criterion	gini, entropy
max_depth	from 3 to 50 with step 2
n_estimators	from 3 to 150 with step 2
min_samples_split	from 1 to 200 with step 5
class_weight	None, balanced
Dataset	**Optimal parameters**
SA_3_class (O)	max_features=0.7, min_samples_leaf=35, criterion=entropy, max_depth=33, n_estimators=130, min_samples_split=82, class_weight=balanced
SA_3_class (S)	max_features=0.7, min_samples_leaf=32, criterion=entropy, max_depth=33, n_estimators=150, min_samples_split=71, class_weight=balanced
SA_3_class (D)	max_features=0.45, min_samples_leaf=29, criterion=gini, max_depth=29, n_estimators=100, min_samples_split=62, class_weight=balanced
SA_3_class_F (S)	max_features=0.8, min_samples_leaf=32, criterion=entropy, max_depth=74, n_estimators=98, min_samples_split=91, class_weight=None
SA_3_class_F (D)	max_features=0.25, min_samples_leaf=27, criterion=entropy, max_depth=69, n_estimators=129, min_samples_split=67, class_weight=None
SA_3_class_RFE (S)	max_features=0.35, min_samples_leaf=37, criterion=gini, max_depth=72, n_estimators=119, min_samples_split=101, class_weight=None

Continued on next page

Dataset	Optimal parameters
SA_3_class_RFE (D)	max_features=0.65, min_samples_leaf=31, criterion=gini, max_depth=64, n_estimators=102, min_samples_split=117, class_weight=None
SA_3_class_BFS (S)	max_features=0.85, min_samples_leaf=31, criterion=entropy, max_depth=51, n_estimators=114, min_samples_split=95, class_weight=None
SA_3_class_BFS (D)	max_features=0.6, min_samples_leaf=40, criterion=entropy, max_depth=28, n_estimators=99, min_samples_split=51, class_weight=None
SA_3_class_E_RFE (S)	max_features=0.85, min_samples_leaf=36, criterion=entropy, max_depth=28, n_estimators=59, min_samples_split=62, class_weight=None
SA_3_class_E_RFE (D)	max_features=0.9, min_samples_leaf=28, criterion=gini, max_depth=42, n_estimators=66, min_samples_split=51, class_weight=None
SA_3_class_E_BFS (S)	max_features=0.55, min_samples_leaf=40, criterion=entropy, max_depth=55, n_estimators=131, min_samples_split=101, class_weight=None
SA_3_class_E_BFS (D)	max_features=0.25, min_samples_leaf=26, criterion=entropy, max_depth=43, n_estimators=55, min_samples_split=81, class_weight=None
SA_2_class (O)	max_features=0.45, min_samples_leaf=27, criterion=entropy, max_depth=46, n_estimators=133, min_samples_split=57, class_weight=balanced
SA_2_class (S)	max_features=0.2, min_samples_leaf=52, criterion=gini, max_depth=43, n_estimators=60, min_samples_split=104, class_weight=balanced

Continued on next page

Dataset	**Optimal parameters**
SA_2_class (D)	max_features=0.25, min_samples_leaf=27, criterion=entropy, max_depth=66, n_estimators=127, min_samples_split=125, class_weight=(0:10000)
SA_2_class_F (S)	max_features=0.15, min_samples_leaf=40, criterion=gini, max_depth=49, n_estimators=114, min_samples_split=91, class_weight=None
SA_2_class_F (D)	max_features=0.15, min_samples_leaf=28, criterion=entropy, max_depth=72, n_estimators=134, min_samples_split=115, class_weight=balanced
SA_2_class_RFE (S)	max_features=0.25, min_samples_leaf=28, criterion=entropy, max_depth=50, n_estimators=119, min_samples_split=111, class_weight=None
SA_2_class_RFE (D)	max_features=0.15, min_samples_leaf=35, criterion=gini, max_depth=45, n_estimators=90, min_samples_split=74, class_weight="balanced"
SA_2_class_BFS (S)	max_features=0.05, min_samples_leaf=40, criterion=gini, max_depth=42, n_estimators=68, min_samples_split=51, class_weight=balanced
SA_2_class_BFS (D)	max_features=0.15, min_samples_leaf=30, criterion=gini, max_depth=66, n_estimators=118, min_samples_split=192, class_weight=None
SA_2_class_E_RFE (S)	max_features=0.9, min_samples_leaf=26, criterion=entropy, max_depth=69, n_estimators=136, min_samples_split=62, class_weight=balanced
SA_2_class_E_RFE (D)	max_features=0.95, min_samples_leaf=30, criterion=entropy, max_depth=71, n_estimators=131, min_samples_split=81, class_weight=balanced
SA_2_class_E_BFS (S)	max_features=0.65, min_samples_leaf=38, criterion=entropy, max_depth=51, n_estimators=61, min_samples_split=86, class_weight=balanced

Continued on next page

Dataset	**Optimal parameters**
SA_2_class_E_BFS (D)	max_features=0.1, min_samples_leaf=32, criterion=entropy, max_depth=63, n_estimators=74, min_samples_split=78, class_weight=balanced
ITC2007_2_class (O)	max_features=0.85, min_samples_leaf=33, criterion=gini, max_depth=30, n_estimators=140, min_samples_split=62, class_weight=balanced
ITC2007_2_class (S)	max_features=0.9, min_samples_leaf=27, criterion=entropy, max_depth=34, n_estimators=100, min_samples_split=93, class_weight=balanced
ITC2007_2_class (D)	max_features=0.4, min_samples_leaf=28, criterion=gini, max_depth=33, n_estimators=100, min_samples_split=63, class_weight=balanced
ITC2007_2_class_F (S)	max_features=0.7, min_samples_leaf=30, criterion=entropy, max_depth=45, n_estimators=148, min_sample_split=80, class_weight=None
ITC2007_2_class_F (D)	max_features=1.0, min_samples_leaf=56, criterion=entropy, max_depth=72, n_estimators=90, min_samples_split=52, class_weight=None
ITC2007_2_class_RFE (S)	max_features=0.75, min_samples_leaf=37, criterion=entropy, max_depth=71, n_estimators=104, min_samples_split=62, class_weight=None
ITC2007_2_class_RFE (D)	max_features=0.5, min_samples_leaf=28, criterion=entropy, max_depth=69, n_estimators=140, min_samples_split=62, class_weight=balanced
ITC2007_2_class_BFS (S)	max_features=1.0, min_samples_leaf=68, criterion=entropy, max_depth=67, n_estimators=134, min_samples_split=106, class_weight=None

Continued on next page

Dataset	Optimal parameters
ITC2007_2_class_BFS (D)	max_features=0.55, min_samples_leaf=30, criterion=entropy, max_depth=44, n_estimators=106, min_samples_split=84, class_weight=None
ITC2007_2_class_E_RFE (S)	max_features=0.25, min_samples_leaf=38, criterion=entropy, max_depth=52, n_estimators=146, min_samples_split=117, class_weight=None
ITC2007_2_class_E_RFE (D)	max_features=0.8, min_samples_leaf=38, criterion=entropy, max_depth=40, n_estimators=102, min_samples_split=140, class_weight=None
ITC2007_2_class_E_BFS (S)	max_features=0.8, min_samples_leaf=26, criterion=gini,max_depth=48, n_estimators=147, min_samples_split=52, class_weight=None
ITC2007_2_class_E_BFS (D)	max_features=0.1, min_samples_leaf=35, criterion=gini, max_depth=59, n_estimators=122, min_samples_split=60, class_weight=balanced

Table C.10: The optimal parameters obtained on the training data using 10-fold cross-validation for the GBC

Parameter	**Parameter range**
max_features	from 0.05 to 1.0 with step 0.05
min_samples_leaf	from 3 to 50 with step 2
criterion	mse, mase
max_depth	from 3 to 50 with step 2
n_estimators	from 3 to 150 with step 2
min_samples_split	from 1 to 200 with step 5
subsample	from 0.05 to 1.0 with step 0.05
learning_rate	0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.7
Dataset	**Optimal parameters**
SA_3_class (O)	max_features=0.8, min_samples_leaf=33, criterion=mse, max_depth=37 n_estimators=83, min_samples_split=200, subsample=0.75, learning_rate=0.05
SA_3_class (S)	max_features=0.4, min_samples_leaf=52, criterion=friedman_mse, max_depth=71 n_estimators=81, min_samples_split=97, subsample=0.75, learning_rate=0.05
SA_3_class (D)	max_features=0.8, min_samples_leaf=37, criterion=mse, max_depth=55 n_estimators=62, min_samples_split=74, subsample=0.8, learning_rate=0.05
SA_3_class FFS (S)	max_features=0.7, min_samples_leaf=70, criterion=mse, max_depth=51 n_estimators=203, min_samples_split=178, subsample=1.0, learning_rate=0.02
SA_3_class FFS (D)	max_features=0.05, min_samples_leaf=46, criterion=mse, max_depth=69 n_estimators=180, min_samples_split=107, subsample=0.8, learning_rate=0.05
SA_3_class RFE (S)	max_features=0.75, min_samples_leaf=39, criterion=mse, max_depth=49 n_estimators=132, min_samples_split=63, subsample=0.8, learning_rate=0.02

Continued on next page

Dataset	Optimal parameters
SA_3_class RFE (D)	max_features=1.0, min_samples_leaf=33, criterion=mse, max_depth=31 n_estimators=189, min_samples_split=241, subsample=0.9, learning_rate=0.05
SA_3_class BFS (S)	max_features=0.25, min_samples_leaf=60, criterion=mse, max_depth=66 n_estimators=83, min_samples_split=123, subsample=0.7, learning_rate=0.05
SA_3_class BFS (D)	max_features=0.4, min_samples_leaf=39, criterion=mse, max_depth=55 n_estimators=176, min_samples_split=215, subsample=0.9, learning_rate=0.05
SA_3_class_E_RFE (S)	max_features=0.05, min_samples_leaf=29, criterion=mse, max_depth=53 n_estimators=187, min_samples_split=146, subsample=1.0, learning_rate=0.02
SA_3_class_E_RFE (D)	max_features=0.8, min_samples_leaf=27, criterion=mse, max_depth=69 n_estimators=213, min_samples_split=93, subsample=0.95, learning_rate=0.02
SA_3_class_E_BFS (S)	max_features=0.95, min_samples_leaf=35, criterion=mse, max_depth=40 n_estimators=75, min_samples_split=163, subsample=0.85, learning_rate=0.05
SA_3_class_E_BFS (D)	max_features=0.65, min_samples_leaf=64, criterion=mse, max_depth=55 n_estimators=205, min_samples_split=136, subsample=0.85, learning_rate=0.05
SA_2_class (O)	max_features=0.35, min_samples_leaf=71, criterion=mse, max_depth=69 n_estimators=59, min_samples_split=92, subsample=0.85, learning_rate=0.05
SA_2_class (S)	max_features=0.55, min_samples_leaf=54, criterion=mse, max_depth=42 n_estimators=73, min_samples_split=57, subsample=1.0, learning_rate=0.05
SA_2_class (D)	max_features=0.1, min_samples_leaf=39, criterion=mae, max_depth=30 n_estimators=72, min_samples_split=96, subsample=1.0, learning_rate=0.1

Continued on next page

Dataset	**Optimal parameters**
SA_2_class_F (S)	max_features=0.15, min_samples_leaf=49, criterion=mse, max_depth=28 n_estimators=127, min_samples_split=74, subsample=0.85, learning_rate=0.05
SA_2_class_F (D)	max_features=0.8, min_samples_leaf=48, criterion=mse, max_depth=55 n_estimators=166, min_samples_split=105, subsample=0.95, learning_rate=0.05
SA_2_class_RFE (S)	max_features=0.85, min_samples_leaf=35, criterion=mse, max_depth=37 n_estimators=209, min_samples_split=111, subsample=0.85, learning_rate=0.02
SA_2_class_RFE (D)	max_features=0.05, min_samples_leaf=28, criterion=mae, max_depth=68 n_estimators=85, min_samples_split=98, subsample=0.9, learning_rate=0.2
SA_2_class_BFS (S)	max_features=0.2, min_samples_leaf=34, criterion=mse, max_depth=50 n_estimators=119, min_samples_split=225, subsample=1.0, learning_rate=0.05
SA_2_class_BFS (D)	max_features=0.2, min_samples_leaf=51, criterion=mse, max_depth=60 n_estimators=73, min_samples_split=228, subsample=0.95, learning_rate=0.1
SA_2_class_E_RFE (S)	max_features=0.1, min_samples_leaf=51, criterion=mse, max_depth=41 n_estimators=216, min_samples_split=208, subsample=0.8, learning_rate=0.05
SA_2_class_E_RFE (D)	max_features=1.0, min_samples_leaf=30, criterion=mse, max_depth=49 n_estimators=195, min_samples_split=113, subsample=0.3, learning_rate=0.05
SA_2_class_E_BFS (S)	max_features=0.9, min_samples_leaf=49, criterion=mse, max_depth=53 n_estimators=182, min_samples_split=83, subsample=0.9, learning_rate=0.02
SA_2_class_E_BFS (D)	max_features=0.75, min_samples_leaf=41, criterion=mse, max_depth=28 n_estimators=176, min_samples_split=249, subsample=0.85, learning_rate=0.2
ITC2007_2_class (O)	max_features=0.3, min_samples_leaf=72, criterion=mse, max_depth=68

Continued on next page

Dataset	Optimal parameters
ITC2007_2_class (S)	n_estimators=130, min_samples_split=202, subsample=0.85, learning_rate=0.2 max_features=0.55, min_samples_leaf=51, criterion=mse, max_depth=43
ITC2007_2_class (D)	n_estimators=139, min_samples_split=147, subsample=0.95, learning_rate=0.05 max_features=0.1, min_samples_leaf=39, criterion=mae, max_depth=30
ITC2007_2_class_F (S)	n_estimators=72, min_samples_split=96, subsample=1.0, learning_rate=0.1 max_features=0.2, min_samples_leaf=32, criterion=mse, max_depth=39
ITC2007_2_class_F (D)	n_estimators=200, min_samples_split=114, subsample=0.6, learning_rate=0.02 max_features=0.35, min_samples_leaf=61, criterion=mse, max_depth=28
ITC2007_2_class_RFE (S)	n_estimators=191, min_samples_split=94, subsample=1.0, learning_rate=0.02 max_features=0.5, min_samples_leaf=34, criterion=mse, max_depth=71
ITC2007_2_class_RFE (D)	n_estimators=173, min_samples_split=127, subsample=0.95, learning_rate=0.05 max_features=0.5, min_samples_leaf=40, criterion=mse, max_depth=55
ITC2007_2_class_BFS (S)	n_estimators=177, min_samples_split=172, subsample=1.0, learning_rate=0.05 max_features=0.95, min_samples_leaf=42, criterion=mse, max_depth=66
ITC2007_2_class_BFS (D)	n_estimators=196, min_samples_split=143, subsample=0.75, learning_rate=0.05 max_features=0.65, min_samples_leaf=48, criterion=mse, max_depth=73
ITC2007_2_class_E_RFE (D)	n_estimators=139, min_samples_split=73, subsample=0.8, learning_rate=0.05 max_features=0.8, min_samples_leaf=66, criterion=mse, max_depth=40
ITC2007_2_class_E_RFE (S)	n_estimators=141, min_samples_split=78, subsample=0.55, learning_rate=0.05 max_features=0.55, min_samples_leaf=74, criterion=mse, max_depth=54
	n_estimators=136, min_samples_split=194, subsample=0.65, learning_rate=0.1

Continued on next page

Dataset	**Optimal parameters**
ITC2007_2_class_E_BFS (S)	max_features=0.7, min_samples_leaf=35, criterion=mse, max_depth=48 n_estimators=170, min_samples_split=64, subsample=0.7, learning_rate=0.05
ITC2007_2_class_E_BFS (D)	max_features=0.65, min_samples_leaf=57, criterion=mse, max_depth=59 n_estimators=147, min_samples_split=213, subsample=0.9, learning_rate=0.05

Table C.11: The optimal parameters obtained on the training data using 10-fold cross-validation for the MLP

Parameter	**Parameter range**
activation	identity, logistic, tanh, relu
solver	lbfgs, sgd, adam
learning_rate	constant, invscaling, adaptive
max_iter	1000
hidden_layer_sizes	(300, 300, 300)
Dataset	**Optimal parameters**
SA_3_class (O)	activation=logistic, solver=adam, learning_rate=invscaling
SA_3_class (S)	activation=logistic, solver=sgd, learning_rate=constant
SA_3_class (D)	activation=logistic, solver=adam, learning_rate=constant
SA_3_class (S + RFE)	activation=tanh, solver=sgd, learning_rate=adaptive
SA_3_class (D + RFE)	activation=relu, solver=adam, learning_rate=invscaling
SA_3_class (S + BFS)	activation=relu, solver=sgd, learning_rate=adaptive
SA_3_class (D + BFS)	activation=relu, solver=sgd, learning_rate=constant
SA_3_class_Ext (S + RFE)	activation=tanh, solver=sgd, learning_rate=constant
SA_3_class_Ext (D + RFE)	activation=tanh, solver=sgd, learning_rate=constant
SA_3_class_Ext (S + BFS)	activation=logistic, solver=adam, learning_rate=invscaling
SA_3_class_Ext (D + BFS)	activation=tanh, solver=sgd, learning_rate=adaptive
SA_2_class (O)	activation=logistic, solver=adam, learning_rate=adaptive
SA_2_class (S)	activation=tanh, solver=sgd, learning_rate=constant

Continued on next page

Dataset	**Optimal parameters**
SA_2_class (D)	activation=relu, solver=sgd, learning_rate=constant
SA_2_class (S + RFE)	activation=relu, solver=sgd, learning_rate=adaptive
SA_2_class (D + RFE)	activation=relu, solver=sgd, learning_rate=adaptive
SA_2_class (S + BFS)	activation=relu, solver=sgd, learning_rate=adaptive
SA_2_class (D + BFS)	activation=logistic, solver=adam, learning_rate=invscaling
SA_2_class_Ext (S + RFE)	activation=relu, solver=sgd, learning_rate=constant
SA_2_class_Ext (D + RFE)	activation=tanh, solver=sgd, learning_rate=constant
SA_2_class_Ext (S + BFS)	activation=relu, solver=sgd, learning_rate=constant
SA_2_class_Ext (D + BFS)	activation=logistic, solver=adam, learning_rate=adaptive
ITC_2_class (O)	activation=logistic, solver=adam, learning_rate=constant
ITC_2_class (S)	activation=relu, solver=sgd, learning_rate=constant
ITC_2_class (D)	activation=logistic, solver=adam, learning_rate=invscaling
ITC_2_class (S + RFE)	activation=tanh, solver=lbfgs, learning_rate=adaptive
ITC_2_class (D + RFE)	activation=indentity, solver=adam, learning_rate=adaptive
ITC_2_class (S + BFS)	activation=identity, solver=sgd, learning_rate=adaptive
ITC_2_class (D + BFS)	activation=logistic, solver=adam, learning_rate=constant
ITC_2_class_Ext (S + RFE)	activation=relu, solver=adam, learning_rate=invscaling
ITC_2_class_Ext (D + RFE)	activation=tanh, solver=sgd, learning_rate=adaptive
ITC_2_class_Ext (S + BFS)	activation=logistic, solver=adam, learning_rate=invscaling
ITC_2_class_Ext (D + BFS)	activation=tanh, solver=adam, learning_rate=adaptive

List of Figures

List of Tables

Bibliography

[140] Itc 2007 benchmarking. `http://www.cs.qub.ac.uk/itc2007/index_files/benchmarking.htm`. Online; accessed 05.07.2016.

[AAB+07] S. Abdullah, S. Ahmadi, E.K. Burke, M Dror, and B McCollum. A tabu based large neighbourhood search methodology for the capacitated examination timetabling problem. *Journal of Operational Research*, 58:1494–1502, 2007.

[AABD07] S. Abdullah, S. Ahmadi, E.K. Burke, and M Dror. Investigating ahuja-orlins large neighbourhood search for examination timetabling. *OR Spectrum*, 29(2):351–372, 2007.

[ABC+03] S. Ahmadi, P. Barone, P. Cheng, P. Cowling, and B. McCollum. Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. In *Proceedings of Multidisciplinary International Scheduling: Theory and Applications (MISTA 2003)*, pages 155–173, 2003.

[ABKD14] M.A. Al-Betar, A.T. Khader, and I.A. Dous. Memetic techniques for examination timetabling. *Annals of Operations Research*, 218:23–50, 2014.

[ACRA10] P. Almajano, J. Cerquides, and J.A. Rodriguez-Aguilar. Empirical hardness for mixed auctions. In *Current Topics in Artificial Intelligence*, volume 5988 of *Lecture notes in Computer Science*, pages 161–170. Springer Berlin Heidelberg, 2010.

[AK98] E. Amaldi and V. Kann. On the approximation of minimizing non zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[ANI08] M. Atsuta, K. Nonobe, and T. Ibaraki. Itc 2007 track 1: An approach using general csp solver. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, 2008.

[Ans73] F.J. Anscombe. Graphs in statistical analysis. *American Statistician*, 27:17–21, 1973.

[Bar96] V.A. Bardadym. Computer-aided school and university timetabling: The new wave. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling: First International Conference Edinburgh, U.K., August 29–September 1, 1995 Selected Papers*, pages 241–250. Springer Berlin Heidelberg, 1996.

[Bat94] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[BB12] J Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[BBNP04] E.K. Burke, Y. Bykov, J.P. Newall, and S. Petrovic. A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, 36(6):509–528, 2004.

[BEM+06] E.K. Burke, A.J. Eckersley, B. McCollum, S. Petrovic, and R. Qu. Hybrid variable neighbourhood approaches to university exam timetabling. Technical report, School of Computer Science, University of Nottingham, 2006.

[BGH+13] E.K. Burke, M. Gendreau, M. Hyde, J. Kendall, G Ochoa, E. Ozcan, and E. Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64:1695–1724, 2013.

[BKdW04] E. K. Burke, J. H. Kingston, and D. de Werra. Applications to timetabling. In Jonathan L. Gross and Jay Yellen, editors, *The Handbook of Graph Theory*, pages 241–250. Chapman Hall/CRC Press, 2004.

[BN96] J. P. Boufflet and S. Negre. Three methods used to solve an examination timetable problem. In E. Burke and P. Ross, editors, *First International Conference Edinburgh, U.K., August 29–September 1, 1995 Selected Papers*, volume 1153 of *Lecture notes in Computer Science*, pages 327–344. Springer, 1996.

[BN99] E.K. Burke and J.P. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

[BN04] E.K. Burke and J.P. Newall. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operational Research*, 129:107–134, 2004.

[BNW96] E. K. Burke, J. P. Newall, and R. F. Weare. A memetic algorithm for university exam timetabling. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling: First International Conference. Selected Papers*, pages 241–250. Springer Berlin Heidelberg, 1996.

[BNW99] E.K. Burke, J.P. Newall, and R.F. Weare. A simple heuristically guided search for the timetable problem. In *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems*, pages 574–579, 1999.

[BPQ06] E. K. Burke, S. Petrovic, and R. Qu. Case-based heuristic selection for timetabling problems. *Journal of Scheduling*, 9(2):115–132, 2006.

[BQS14] E.K. Burke, R. Qu, and A. Soghier. Adaptive selection of heuristics for improving exam timetables. *Annals of Operations Research*, 218(1):129–145, 2014.

[Bre79] D. Brelaz. New methods to color the vertices of a graph. *Communication of the ACM*, 22(4):251–256, 1979.

[BSU14] M. Battistutta, A. Schaerf, and T. Urli. Feature-based tuning of single-stage simulated annealing for examination timetabling. In *Practice and Theory of Automated Timetabling (PATAT 2014)*, pages 53–61, 2014.

[Car78] M.W. Carter. Examination scheduling: Documentation. Technical report, Data Processing Department, University of Waterloo, 1978.

[Car86] M.W. Carter. A survey of practical applications of examination timetabling algorithms. *Journal Operations Research*, 34(2):193–202, 1986.

[CK96] T. B. Cooper and J. H. Kingston. The complexity of timetable construction problems. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling: First International Conference. Selected Papers*, Lecture notes in Computer Science, pages 281–295. Springer Berlin Heidelberg, 1996.

[CLL96] M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 47(3):373–383, 1996.

[CR10] D. W. Corne and A.P. Reynolds. Optimisation and generalisation: Footprints in instance space. In *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture notes in Computer Science*, pages 22–31. Springer, 2010.

[CS14] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(4):16–28, 2014.

[CV95] C. Cortes and V Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[CWS05] P. Cote, T. Wong, and R. Sabouri. Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem.

In E.K. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 5th International Conference*, volume 3616 of *Lecture notes in Computer Science*, pages 151–168. Springer, 2005.

[CY09] L.Y. Chuang and C.H. Yang. Tabu search and binary particle swarm optimization for feature selection using microarray data. *Journal of computational biology*, 16(12):1689–1703, 2009.

[DCE+13] J. Demvsar, T. Curk, A. Erjavec, C. Gorup, M. Milutinovic, T. Hocevar, M. Polajnar, M. Polajnar, Toplak M., A. Staric, M Stajdohar, L. Umek, L. Zagar, J. Zbontar, M. Zitnik, and B. Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[dG02] L. di Gaspero. Recolour, shake and kick: A recipe for the examination timetabling problem. In E.K. Burke and P. de Causmaecker, editors, *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling*, pages 404–407, 2002.

[dGS01] L. di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference*, volume 2079 of *Lecture notes in Computer Science*, pages 104–117. Springer Berlin Heidelberg, 2001.

[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning (ICML'95)*, pages 194–202, 1995.

[dS08] G. de Smet. Itc 2007 - examination track. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, 2008.

[DT05] K.A. Dowsland and J. Thompson. Ant colony optimization for the examination scheduling problem. *Journal of Operational Research Society*, 56:426–438, 2005.

[Due90] G. Dueck. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Computational Physics*, 90:161–175, 1990.

[DWS98] J. Duncan Watts and Steven Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, 1998.

[ER04] T. Elomaa and J. Rousu. Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. *Data Mining and Knowledge Discovery*, 8(2):97–126, 2004.

[FI93] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Uncertainty in AI*, pages 1022–1027, 1993.

[GAH08] S. Gogos, P. Alefragis, and E. Housos. A multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 194(1):203–221, 2008.

[GAH12] C. Gogos, P. Alefragis, and E. Housos. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 194(1):203–221, 2012.

[Gay51] A.K. Gayen. The frequency distribution of the product moment correlation coefficient in random samples of any size draw from non-normal universes. *Biometrika*, 38:219–247, 1951.

[GE03] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[GLS$^+$13] S. Garcia, J. Luengo, J.A. Saez, V. Lopez, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013.

[Gro] ASAP Group. Timetabling competition 2007: Examination timetabling competition. `http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index.html`. Online; accessed 05.07.2016.

[Guo03] H. Guo. *Algorithm Selection for Sorting and Probabilistic Inference: A Machine Learning-Based Approach.* PhD thesis, Kansas State University, 2003.

[GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[Hal99] Mark A. Hall. Correlation-based feature selection for machine learning. Technical report, 1999.

[HCL10] Ch Hsu, Ch. Chang, and Ch. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2010.

[Her91] A. Herz. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54:37–47, 1991.

[HFH$^+$09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[HHLB10] F. Hutter, H.H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming (CPAIOR 2010)*, volume 6140 of *Lecture notes in Computer Science*, pages 186–202. Springer, 2010.

[HHLB11] F. Hutter, H.H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the Fifth International Conference on Learning and Intelligent Optimization (LION'11)*, volume 6683 of *Lecture notes in Computer Science*, pages 507–523. Springer Berlin Heidelberg, 2011.

[HLBH12] F. Hutter, K. Leyton-Brown, and H.H. Hoos. Parallel algorithm configuration. In *Learning and Intelligent Optimization (LION 6)*, pages 55–70, 2012.

[HLBHH06] F. Hutter, K. Leyton-Brown, H.H. Hoos, and Y. Hamadi. Performance prediction and automated tuning of randomized and parametric algorithms. In *Principles and Practice of Constraint Programming - CP 2006*, volume 4204 of *Lecture notes in Computer Science*, pages 213–228. Springer, 2006.

[HLS14] H.H. Hoos, M. Lindauer, and T. Schaub. Claspfolio 2: Advances in algorithm selection for answer set programming. *Theory and Practice of Logic Programming*, 14:569–585, 2014.

[HP04] L. Hong and S.E. Page. Groups of diverse problem solvers can outperform groups of highability problem solvers. *Proceedings of the National Academy of Sciences of the United States of America*, 101(46):16385–16389, 2004.

[HXHLB14] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods and evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[ICfDA] University College Cork Insight Centre for Data Analytics. Icon challenge on algorithm selection. `http://challenge.icon-fet.eu/`. Online; accessed 05.07.2016.

[Jan00] N. Jankowski. Data regularization. In L. Rutkowski and R. Tadeusiewicz, editors, *Neural Networks and Soft Computing*, pages 209–214, 2000.

[JSW98] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, The IBM Research Symposia Series, pages 805–103. Springer, 1972.

[KdCHS11] J. Kanda, A.C.P.L.F. de Carvalho, E.R. Hruschka, and C. Soares. Selection of algorithms to solve traveling salesman problems using meta-learning. *Int. J. Hybrid Intell. Syst.*, 8:117–128, 2011.

[KGV83] S. Kirkpatrick, S.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[KJ97] R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[KK06] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

[KL03] S.S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15:1667–1689, 2003.

[Kon95] I. Kononenko. On biases in estimating multi-valued attributes. In *IJCAI*, volume 95, pages 1034–1040, 1995.

[Kot14] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI MAGAZINE*, 35(3):48–60, 2014.

[KS04] P. Kostuch and K. Socha. Hardness prediction for the university course timetabling problem. In *Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture notes in Computer Science*, pages 135–144. Springer Berlin Heidelberg, 2004.

[KS05] N. Krasnogor and J.E. Smith. A tutorial for competent memetic algorithms: model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[LBHHX14] K. Leyton-Brown, H.H. Hoos, F. Hutter, and L. Xu. Understanding the empirical hardness of np-complete problems. *Communications of the ACM*, 57(5):98–107, 2014.

[LBNA$^+$03] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. Boosting as a metaphor for algorithm design. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03*, volume 2833 of *Lecture notes in Computer Science*, pages 899–903. Springer, 2003.

[LBNS06] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In P. Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, volume 2470 of *Lecture notes in Computer Science*, pages 556–572. Springer Berlin Heidelberg, 2006.

[LG97] M. Laguna and F. Glover. *Tabu search.* Kluwer Academic Publishers Norwell, 1997.

[LHHS15] M. Lindauer, H.H. Hoos, F. Hutter, and T. Schaub. Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence*, 53:745–778, 2015.

[Lin02] S.L.M. Lin. A broker algorithm for timetabling problem. In *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling*, pages 372–386, 2002.

[LL98] L. Lobjois and M. Lemaitre. Branch and bound algorithm selection by performance prediction. In *Proceedings of the national conference on artificial intelligence*, 1998.

[LTM$^+$12] C. Lazar, J. Taminau, S. Meganck, D. Steenhoff, A. Coletta, C. Molter, V. de Schaetzen, R. Duque, H. Bersini, and A. Nowé. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM Trans Comput Biol Bioinform*, 9(4):1106–1119, 2012.

[MB10] Nicolai Meinshausen and Peter Buhlmann. Stability selection. *Journal of the Royal Statistical Society*, 72(4):417–473, 2010.

[MBHP03] L.T.G. Merlot, N. Boland, B.D. Hughes, and Stuckey P.J. A hybrid algorithm for the examination timetabling problem. In E.K. Burke and P. de Causmaecker, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference*, pages 207–231, 2003.

[MdC11] T. Messelis and P. de Causmaecker. An algorithm selection approach for nurse rostering. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, pages 160–166, 2011.

[ME14] K.M. Malan and A. Engelbrecht. Fitness landscape analysis for metaheuristic performance prediction. In A. Engelbrecht and H. Richter, editors, *Recent Advances in the Theory and Application of Fitness Landscapes*, volume 6 of *Emergence, Complexity and Computation*, pages 103–132. Springer Berlin Heidelberg, 2014.

[Mes14] T. Messelis. *Algorithm performance prediction for practical combinatorial optimisation problems.* PhD thesis, KU Leuven, 2014.

[MF00] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, 2000.

[MM08] Barry McCollum and Paul McMullan. The second international timetabling competition: Examination timetabling track. technical report. Technical report, The Second International Timetabling Competition, 2008.

[MMP^{+}09] B. McCollum, P. McMullan, A.J. Parkes, E.K. Burke, and S. Abdullah. An extended great deluge approach to the examination timetabling problem. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009)*, pages 424–434, 2009.

[MS13] N. Musliu and M. Schwengerer. Algorithm selection for the graph coloring problem. In *Learning and Intelligent Optimization Conference LION 7*, pages 389–403, 2013.

[Mue09] T. Mueller. Itc2007 solver description: A hybrid approach. *Annals of Operations Research*, 172:429, 2009.

[MZ06] R. Meiri and J. Zahavi. Using simulated annealing to optimize the feature selection problem in marketing applications. *European Journal of Operational Research*, 171(3):842–858, 2006.

[NA04] Z. Naji Azimi. Comparison of metaheuristic algorithms for examination timetabling problem. *Applied Mathematics and Computation*, 16(1-2):337–354, 2004.

[NA05] Z. Naji Azimi. Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, 163(2):705–733, 2005.

[Net] Metaheuristics Network. International timetable competition 2002-2003. `http://www.idsia.ch/Files/ttcomp2002/`. Online; accessed 02.07.2016.

[New06a] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2006.

[New06b] M.E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, 2006.

[NLBD^{+}04] E. Nudelman, K. Leyton-Brown, A. Devkar, Y. Shoham, and H.H. Hoos. Understanding random sat: Beyond the clauses-to-variables ratio. In *International Conference on Principles and Practice of Constraint Programming (CP)*, volume 3258 of *Lecture notes in Computer Science*, pages 438–452. Springer Berlin Heidelberg, 2004.

[OHH$^+$08] E. O'Mahony, E. Hebrard, A. Holland, C. Nugent, and B. O'Sullivan. Using case-based reasoning in an algorithm portfolio for constraint solving. In *Irish Conference on Artificial Intelligence and Cognitive Science*, 2008.

[Omo89] S.M. Omohundro. Five balltree construction algorithms. Technical report, International Computer Science Institute, 1989.

[OUOK07] O. O. Ulker, E. Ozcan, and E.E. Korkmaz. Linear linkage encoding in grouping problems: applications on graph coloring and timetabling. In E.K. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 6th International Conference*, volume 3867 of *Lecture notes in Computer Science*, pages 347–363. Springer, 2007.

[PB07a] N. Pillay and W. Banzhaf. A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In *13th Portuguese Conference on Aritficial Intelligence (EPIA 2007)*, volume 4874 of *Lecture notes in Computer Science*, pages 223–234. Springer, 2007.

[PB07b] N. Pillay and W. Banzhaf. A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In *13th Portuguese Conference on Aritficial Intelligence (EPIA 2007)*, volume 4874 of *Lecture notes in Computer Science*, pages 223–234. Springer, 2007.

[PB08] N. Pillay and W. Banzhaf. A developmental approach to the uncapacitated examination timetabling problem. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature*, pages 276–285, 2008.

[PB10] N. Pillay and W. Banzhaf. An informed genetic algorithm for the examination timetabling problem. *Applied Soft Computing*, 10(2):457–467, 2010.

[PBGC00] B. Pfahringer, H. Bensusan, and C.G. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, pages 743–750, 2000.

[Pil04] N. Pillay. An analysis of representations for hyper-heuristics for the uncapacitated examination timetabling problem in a genetic programming system. In *Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, ACM International Conference Proceeding Seri, pages 188–192, 2004.

[Pil10] N. Pillay. Evolving hyper-heuristics for a highly constrained examination timetabling problem. In *In Proceedings of the 8th International Conference* *on the Practice and Theory of Automated Timetabling (PATAT'10)*, pages 336–346, 2010.

[Pil12] N. Pillay. Evolving hyper-heuristics for the uncapacitated examination timetabling problem. *Journal of the Operational Research Society*, 63(1):47–58, 2012.

[PM14] J. Pihera and N. Musliu. Application of machine learning to algorithm selection for tsp. In *ICTAI 2014*, pages 47–54, 2014.

[PT09] L. Pulina and A. Tacchella. A self-adaptive multi-engine solver for quantified boolean formulas. *Constraints*, 14(1):80–116, 2009.

[PW66] J.E. Peck and M.R. Williams. Algorithm 286 – examination scheduling. *Communication of the ACM*, 9:433–434, 1966.

[QB07] R. Qu and E.K. Burke. Adaptive decomposition and construction for examination timetabling problems. In *Multidisciplinary International Scheduling: Theory and Applications 2007 (MISTA'07)*, pages 418–425, 2007.

[QBM$^+$09] R. Qu, E. K. Burke, B. Mccollum, L. T. Merlot, and S. Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *J. of Scheduling*, 12(1):55–89, February 2009.

[Ree99] C. R. Reeves. Landscapes, operators and heuristic search. *Annals of Operations Research*, 86:473–490, 1999.

[RH07] M. Roberts and A. Howe. Learned models of performance for many planners. In *ICAPS 2007, Workshop AI Planning and Learning*, 2007.

[RHC98] P. Ross, E. Hart, and D. Corne. Some observations about ga-based exam timetabling. In E.K. Burke and M.W. Carter, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 2nd International Conference*, volume 1408 of *Lecture notes in Computer Science*, pages 115–129. Springer, 1998.

[Ric76] J.R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[RMBH04] P. Ross, J.G. Marin-Blazquez, and E Hart. Hyper-heuristics applied to class and exam timetabling problems. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, pages 1691–1698, 2004.

[RNM06] C. Rich and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceeding ICML '06 Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

[RNMY08] C. Rich, A. Niculescu-Mizil, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceeding ICML '08 Proceedings of the 25th international conference on Machine learning*, pages 96–103, 2008.

[RRF$^+$11] D.N. Reshef, Y.A. Reshef, H.K. Finucane, S.R. Grossman, G. McVean, P.J. Turnbaugh, E.S. Lander, M. Mitzenmacher, and P.C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[Sch99] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

[sci] Scikit-learn. machine learning in python. `http://http://scikit-learn.org/`. Online; accessed 05.07.2016.

[Sha98] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Fourth International Conference on Principles and Practice of Constraint Programming*, volume 1520 of *Lecture notes in Computer Science*, pages 417–431. Springer, 1998.

[She02] K. Sheibani. An evolutionary approach for the examination timetabling problems. In E.K. Burke and P. de Causmaecker, editors, *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling*, pages 387–396, 2002.

[SK04] S.C. Shah and A. Kusiak. Data mining and genetic algorithm based gene/snp selection. *Artificial Intelligence in Medicine*, 31(3):183–196, 2004.

[SM08] K.A. Smith-Miles. Algorithm selection for combinatorial search problems: A survey. *ACM Computing Surveys (CSUR)*, 41(1):6:1–6:25, 2008.

[SMI11] K. A. Smith-Miles and R.M.D. Islam. Meta-learning of instance selection for data summarization. In N. Jankowski, W. Duch, and K. Grabczewski, editors, *Meta-Learning in Computational Intelligence*, volume 358 of *Studies in Computational Intelligence*, pages 77–95. Springer Berlin Heidelberg, 2011.

[SML11] K. A. Smith-Miles and L. Lopes. Generalising algorithm performance in instance space: A timetabling case study. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture notes in Computer Science*, pages 524–538. Springer Berlin Heidelberg, 2011.

[SML12] K. A. Smith-Miles and L. Lopes. Measuring instance difficulty for combinatorial optimization problems. *Computers and Operations Research*, 39(5):875–889, 2012.

[SMT12] K. A. Smith-Miles and T.T. Tan. Measuring algorithm footprints in instance space. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 3446–3463, 2012.

[SMvHL10] K. A. Smith-Miles, J. van Hemert, and X.Y. Lim. Understanding tsp difficulty by learning from evolved instances. In *Learning and Intelligent Optimization*, volume 6073 of *Lecture notes in Computer Science*, pages 266–280. Springer, 2010.

[SMWLI13] K. A. Smith-Miles, B. Wreford, L. Lopes, and N. Insani. Predicting metaheuristic performance on graph coloring problems using data mining. In *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 417–432. Springer Berlin Heidelberg, 2013.

[SS92] P. F Stadler and W. Schnabl. The landscape of the traveling salesman problem. *Physics Letters A*, 161(4):337–344, 1992.

[SW48] C.E. Shannon and W. Weaver. The mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[TD98] J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Computers and Operations Research*, 25:637–648, 1998.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

[TMW99] E. Tsang, P. Mills, and R. Williams. A computer aided constraint programming system. In *The 1st International Conference on The Practical Application of Constraint Technologies and Logic Programming (PACLP)*, pages 81–93, 1999.

[TPC08] J. Tavares, F.B. Pereira, and E. Costa. Multidimensional knapsack problem: A fitness landscape analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(3):604–616, 2008.

[vLA87] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Company, Kluwer Academic Publishers Group, 1987.

[VWW06] V. Vassilevska, R. Williams, and S.L.M. Woo. Confronting hardness using a hybrid approach. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1–10, 2006.

[WCG02] T. Wong, P. Cote, and P Gely. Final exam timetabling: a practical approach. In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2002)*, volume 2, pages 726–731, 2002.

[Wel47] B.L Welch. The generalization of "student's" problem when several different population variances are involved. *Biometrika*, 1-2:28–35, 1947.

[WFH11] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 2011.

[WKRQ$^+$07] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, 2007.

[WM97] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[WM00] D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[WP67] D.J.A. Welsh and M.B. Powell. The upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 11:41–47, 1967.

[WX01] G. M. White and B.S. Xie. Examination timetables and tabu search with longer-term memory. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference*, volume 2079 of *Lecture notes in Computer Science*, pages 85–103. Springer Berlin Heidelberg, 2001.

[WXZ04] G. M. White, B.S. Xie, and S. Zonjic. Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, 153(16):80–91, 2004.

[XHHLB08] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research*, 32(1):565–606, 2008.

[XHHLB12] L. Xu, H.H. Hutter, H.H. Hoos, and K. Leyton-Brown. Satzilla 2012: Improved algorithm selection based on cost-sensitive classification models. In *Proceedings of SAT Challenge 2012 : Solver and Benchmark Descriptions*, pages 57–58, 2012.

[XHLB12] L. Xu, H.H. Hutter, and K. Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. In *International Conference on Theory and Applications of Satisfability Testing (SAT'12)*, pages 228–241, 2012.

[Zel74] M. Zeleny. A concept of compromise solutions and method of displaced ideal. *Computers and Operations Researh*, 1(4):479–496, 1974.