Journal of Scheduling (2021) 24:103–121
https://doi.org/10.1007/s10951-020-00641-y

Measuring the complexity of university timetabling instances

Felipe de la Rosa-Rivera[1] · Jose I. Nunez-Varela[1] · Cesar A. Puente-Montejano[1] · Sandra E. Nava-Muñoz[1]

Published online: 19 February 2020

Abstract
University timetabling is a real-world problem frequently encountered in higher education institutions. It has been studied by many researchers who have proposed a wide variety of solutions. Measuring the variation of the performance of solution approaches across instance spaces is a critical factor for algorithm selection and algorithm configuration, but because of the diverse conditions that define the problem within different educational contexts, measurement has not been formally addressed within the university timetabling context. In this paper, we propose a set of metrics to predict the performance of combinatorial optimization algorithms that generate initial solutions for university timetabling instances. These metrics, derived from the fields of enumerative combinatorics and graph coloring, include size-related instance properties, counting functions, feature ratios and constraint indexes evaluated through a feature selection methodology that, based on regression algorithms, characterizes the empirical hardness of a subspace of synthetically generated instances. The results obtained with this methodology show the current need not only to develop solution strategies for particular cases of the problem, but also to produce a formal description of the conditions that make instance spaces hard to solve, in order to improve and integrate the available solution approaches.

Keywords University timetabling · Empirical hardness · Feature selection · Instance generator

1 Introduction

University timetabling is a combinatorial optimization problem that school institutions must solve frequently. At its core, it requires the assignment of resources and time slots to a defined set of academic events (i.e., classes), according to a series of hard (mandatory) and soft (optional) constraints. However, given the diversity of the conditions that defines the timetabling problem, some researchers have chosen to divide it into three related problems (McCollum et al. 2010):

- *Examination timetabling (Ex-TT)* Schedules examination sessions of a given duration into a number of periods while satisfying a number of hard constraints.
- *Post-enrollment course timetabling (PE-CTT)* Generates timetables in such a way that all students can attend all the classes in which they have previously enrolled.
- *Curriculum-based course timetabling (CB-CTT)* Schedules predefined sets of classes in which groups of students are to be enrolled.

Because of their educational models, many universities must solve the CB-CTT problem to produce feasible timetables that maximize both professor/student satisfaction and usage of resources. But as this problem has proved to be NP-complete in practice (Cooper and Kingston 1995), finding an optimal solution is difficult to accomplish.

Recent surveys (Pillay 2014; MirHassani and Habibi 2013; Bouajaja and Dridi 2016; Sahargahi and Drakhshi 2016) show an increase in the number and scope of heuristics and algorithms proposed to solve the CB-CTT problem. However, because of a lack of characterization of the *solution*

✉ Felipe de la Rosa-Rivera
fdelarosa@alumnos.uaslp.edu.mx

Jose I. Nunez-Varela
jose.nunez@uaslp.mx

Cesar A. Puente-Montejano
cesar.puente@uaslp.mx

Sandra E. Nava-Muñoz
senavam@uaslp.mx

[1] Facultad de Ingeniería, Universidad Autónoma de San Luis Potosí, Av. Dr. Manuel Nava No. 8, Edificio I. Zona Universitaria, C.P. 78290 San Luis Potosí, Mexico

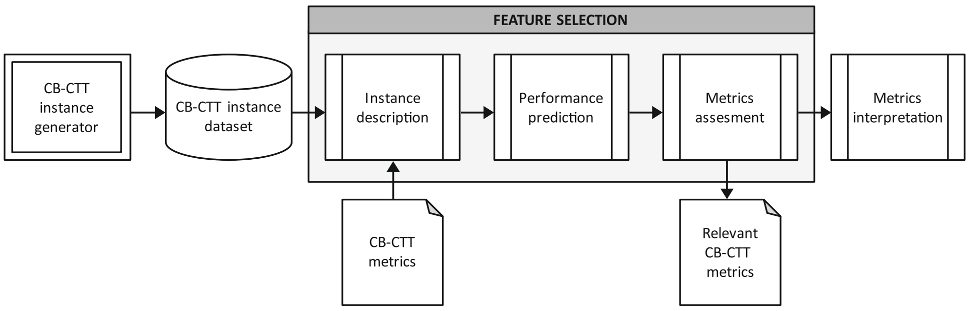

Fig. 1 Feature selection methodology used to identify the metrics that better describe the empirical hardness of CB-CTT instances

subspaces in which they perform well, choosing the solving approach that best suits the particular conditions of an instance is a complex task, often addressed with experimental approaches rather than with formal analyses.

This paper presents a formal analysis of the conditions that define the complexity of generating initial solutions for CB-CTT instances based on the concept of an *empirical hardness model*. As defined by Leyton-Brown et al. (2002), an empirical hardness model is a model able to predict the performance of an algorithm on distributions or particular instances of a problem, based on a set of relevant features that describes the main properties of the problem. Therefore, it can be used both to develop new solving approaches and to improve the theoretical understanding of a problem domain.

Based on a feature selection methodology, this paper first proposes: (1) the design of a parameterized CB-CTT instance generator that can be applied to produce instances of different complexity, and (2) a collection of metrics to characterize them. Then, as a result of the interpretation of empirical hardness models built from different machine learning methods, it explains the set of relevant features that best describe the hardness (difficulty of solution) of the instances.

The remainder of this paper is organized according to the feature selection methodology presented in Fig. 1. Section 2 defines the basic concepts of the CB-CTT problem, its mathematical formulation and representation. Section 3 describes the design and setting of the *CB-CTT instance generator* for formulating instances with different characteristics. Section 4 describes the diversity of the generated *CB-CTT instance dataset*. Section 5 summarizes the set of *CB-CTT metrics* that are proposed to characterize the hardness of the instances. Section 6 explains the feature selection processes (*instance description*, *performance prediction*, and *metrics assessment*), that are applied in selecting the *relevant CB-CTT metrics* (described in Sect. 7). Section 8 presents the main insights drawn from the *metrics interpretation*. Finally, Sect. 9 summarizes the main conclusions and provides some directions for further research.

2 CB-CTT fundamentals

2.1 Mathematical formulation

As defined by Di Gaspero et al. (2007) for the Second International Timetabling Competition (ITC), the Curriculum-Based Course Timetabling (CB-CTT) problem consists of the weekly scheduling of lectures for several university courses given a number of teachers, rooms, and time periods. Any conflicts between courses are set according to the curricula published by the university.

Based on this definition, the basic input data that are required to formulate a CB-CTT problem instance are:

– A set of teaching days D (where a day $d \in D$); and a set of teaching modules per day M_d, (where a module $m_d \in M_d$).
– A set of time slots TS composed of a day and a module $< d, m_d >$.
– A set of teachers T, where each teacher $t \in T$ has a limited weekly workload $w_t \in \mathbb{N}$.
– A set of rooms R, where each room $r \in R$ has a limited weekly availability $a_r \in \mathbb{N}$.
– A set of courses C, where each course $c \in C$ has a total weekly duration $d_c \in \mathbb{N}$, which in turn defines the number of lectures to be scheduled. A course requires at least three resources: a teacher, a room, and a group of students.
– A set of curricula Q, where a curriculum $q \in Q$ is a group of courses that shares a group of students. Therefore, this group of courses cannot be scheduled at the same times.

Thus, in general terms, the solution to a CB-CTT instance consists of assigning the required time slots and resources to all lectures, according to a set of constraints. Hard constraints are mandatory; they must be fulfilled in order to produce a *feasible* (valid) solution. In contrast, soft constraints are optional; their fulfillment increases the quality of the feasible solution.

Fig. 2 Structure of the extended course timetabling format *ECTT*

```
Courses: <CourseID> <Teacher> <# Lectures> <MinWorkingDays> <# Students> <Double Lectures>
Rooms: <RoomID> <Capacity> <Site>
Curricula: <Curriculum Id> <# Courses> <CourseID> ... <CourseID>
Unavailability_Constraints: <CourseID> <Day> <Day_period>
Room_Constraints: <CourseID> <RoomID>
```

2.2 Timetabling data formats

In the literature, at least two data formats have been used to represent CB-CTT problems: the extended course timetabling format (*ECTT*) and the XML High School Timetabling format (*XHSTT*).

2.2.1 Extended course timetabling format

Bonutti et al. (2012) proposed the ECTT format to extend the representation capacity of the *course timetabling data format* (*CTT*), developed to handle the instances of the ITC 2007.

The ECTT format represents the data of timetabling instances according to the following five sections (shown in Fig. 2):

- *Courses* Events to be scheduled, each labeled with an ID and defined by the following fields: teacher, number of lectures, minimum number of days in which lectures must be given, maximum number of students, and specification of double lectures.
- *Rooms* Set of available classrooms, each labeled with an ID and described in terms of its capacity and location.
- *Curricula* Events shared by a group of students, each labeled with an ID and described by the number and the IDs of the courses which belong to the curriculum.
- *Unavailability constraints* Set of time slots in which courses cannot be allocated.
- *Room constraints* Classrooms in which courses must be allocated.

2.2.2 XML high school timetabling format

The XHSTT data format was selected by the Euro Working Group on Automated Timetabling (Euro WATT) as the standard to be used in ITC 2011. The structure of this format is defined by an XML schema composed of four entities (Post et al. 2014):

- *Times* Set of possible time slots in which events can be scheduled. These time slots are often grouped into *time groups* (e.g., days, weeks).
- *Resources* Set of available resources that can be assigned to the events. Each resource belongs to a specific resource type (e.g., teacher, rooms); resources can be grouped into *resource groups* for administration purposes.
- *Events* Set of classes to be scheduled. Each event has a duration, which represents the amount of time slots that must be scheduled, and a demand of a set of resources. Events can also be grouped into *event groups*.
- *Constraints* Set of constraints that must be fulfilled during the scheduling of events. Each constraint has a cost that indicates the penalty value of a single violation, and a cost function, which defines how the penalty values are added to the objective function to be minimized. Table 1 summarizes the 16 types of constraints available.

2.2.3 Comparison of CB-CTT data formats

The two data formats described above differ both in their structure and in their representation capability. On the one hand, the ECTT format operates based on a text file which encodes instances according to a fixed data order. In addition to the basic resource allocation and task scheduling constraints, it allows the representation of CB-CTT problems which are limited by two types of constraints: unavailable times of courses and allocation of rooms. On the other hand, the XHSTT format structures data according to an XML schema, which uses four entities to define scheduling problems based on a set of sixteen types of constraints that can be applied to represent a diverse set of real-life conditions.

For our research, the XHSTT format was selected because of its broader capacity to represent real-life conditions that cannot be encoded using the ECTT format, such as allocating rooms to courses and defining working shifts for teachers. These real-life conditions are particularly relevant to our study because their effect in the solution space of CB-CTT instances has not been addressed in literature.

3 CB-CTT instance generator

The first issue that needs to be addressed in order to build empirical hardness models is counting with enough data to represent the diversity of conditions found in real situations. Within the field of university timetabling, at present, there is only an available standard benchmarking library of instances modeled according to the XHSTT format, namely the XHSTT-2014 dataset hosted by the Centre of Telematics and Information Technology (CTIT) of the University of Twente (DMPP Group UoT 2014). The dataset consists

Table 1 Description of the 16 types of constraints available in the XHSTT format

	Name	Acronym	Description
1	Assign Resource Constraint	ARC	Requires that resources demanded by the classes be assigned
2	Assign Time Constraint	ATC	Requires the assignment of times to classes
3	Split Events Constraint	SEC	Limits the number of lectures that can be derived from a given class, and their duration
4	Distribute Split Events Constraint	DSEC	Limits the number of lectures of a particular duration that can be derived from a given class
5	Prefer Resources Constraint	PRC	Specifies that some resources are preferred to be assigned to some classes
6	Prefer Times Constraint	PTC	Specifies that some times are preferred to be assigned to some classes
7	Avoid Split Assignments Constraint	ASAC	Specifies that the resources assigned to all non-consecutive lectures derived from a given class must not vary
8	Spread Events Constraint	SPEC	Specifies that non-consecutive lectures should be spread out in time
9	Link Events Constraint	LEC	Specifies that a certain set of classes must be assigned the same times
10	Order Events Constraint	OEC	Specifies that the times of two classes must be assigned in such a way that the first class ends before the second begins
11	Avoid Clashes Constraint	ACC	Specifies that certain resources must have no clashes; that is, a student should not be expected to attend two or more classes simultaneously
12	Avoid Unavailable Times Constraint	AUTC	Specifies that some resources are unavailable to attend any class at certain times
13	Limit Idle Times Constraint	LITC	Limits the number of times that resources stand idle within a *time group*
14	Cluster Busy Times Constraint	CBTC	Limits the number of *time groups* a resource is busy
15	Limit Busy Times Constraint	LBTC	Limits the number of times within a *time group* that a resource is busy
16	Limit Workload Constraint	LWC	Limits the total workload assigned to a resource

of a collection of 25 real-world instances that are encoded according to the XML schema proposed by Post et al. (2014). However, because of its size, it is not representative enough to analyze the characterization of CB-CTT instances.

To overcome the unavailability of CB-CTT instances, we have created a customizable CB-CTT instance generator, able to produce a large number of instances with varying characteristics. The design and the required settings of this generator are described in this section.

3.1 Design of a CB-CTT instance generator

To generate a diverse CB-CTT dataset, a customizable two-stage instance generator was developed to model both the *curricular plan* (Stage 1) and the *particular conditions* that are required to formulate a CB-CTT instance (Stage 2).

Curricular plans are general descriptions of academic programs, and in a broad sense define: (1) the set of obligatory and optional courses, (2) the sequence of these courses, and (3) the times and resources that they require. *Particular conditions* are institutional regulations that must be followed for scheduling lectures and assigning resources within a determined educational context, for example, limiting to five the number of lectures a student can attend in a single day.

As explained in Sect. 2.2.2, the XHSTT data format represents timetabling instances based on four entities, namely *times*, *resources*, *events* and *constraints*. *Events* and their requirements are defined in Stage 1, as an answer to the planning question: "What is to be scheduled?", whereas *times*, *resources*, and *constraints* are defined in Stage 2, as an answer to the planning question: "How is to be scheduled?".

3.1.1 Stage 1: curricular plan

The curricular plan is described at two levels. At a macro-level, the number of terms that specify the periods (e.g., semesters, trimesters, quarters) required to attain a bachelor's degree, and the number of courses in each term define the *curricular grid*. At a micro-level, the time slots and the resources required by each course define the *courses requirements*.

After setting the *curricular grid* and the *courses requirements*, the number of classes to be scheduled is defined by two parameters: *active terms* and *number of groups*. The first parameter refers to a subset of terms, from the curricular plan, planned to be available for student enrollment. The second parameter refers to the number of groups planned to be available for each course of these *active terms*.

Figure 3 presents a graphical example of a five-term curricular plan whose first, third, and fifth terms (in white color) have been defined as active terms. Each square represents a course, whose requirements are defined by three randomly generated features: *weekly duration*, *type of professor required*, and *type of room required*.

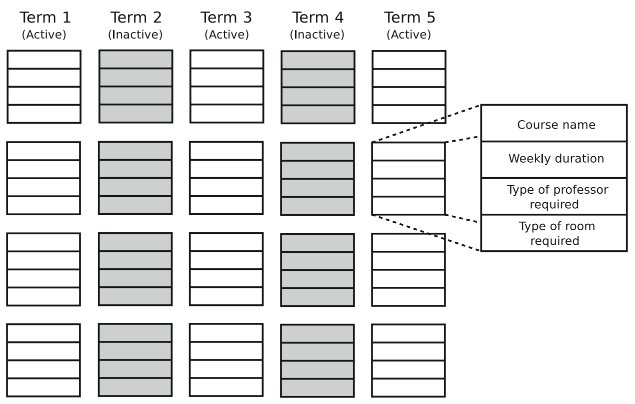

Fig. 3 Graphical representation of a five-term curricular plan, where terms 1, 3 and 5 are available for student enrollment, and each square represents a course defined by its name, weekly duration, type of professor and type of room required

3.1.2 Stage 2: particular conditions

Times and resources

Times are defined by a list of non-overlapping time intervals, called *time slots*, that are grouped into sets of *days* and *working shifts*. For example, a time grid consisting of four morning time slots and four evening time slots, from Monday to Friday, would be modeled with the following parameters: <*time slots per day*><8>; <*days*><5>; <*shifts*><2>.

Resources are elements that have to be present during an entire *subevent* (i.e., a lecture). Our instance generator considers three types of resources:

- *Curricula* The concept of curriculum, which refers to a set of classes that are shared by a group of students, is modeled by assigning the same resource to all the *events* planned for a group. For example, if two groups are considered for the courses in the *active terms* shown in Fig. 3, the instance generator will define six curricula: Term 1 Group A, Term 1 Group B, Term 3 Group A, Term 3 Group B, Term 5 Group A, and Term 5 Group B.
- *Teachers* Professors are grouped based on three attributes: *preferred working shift* (e.g., morning, evening), *knowledge area* (e.g., math, science, management, etc.), and *type of contract* (full-time or part-time teacher).
- *Rooms* The physical spaces in which classes take place are categorized according to their equipment as classrooms or laboratories.

Constraints

After defining *events*, *times*, and *resources*, the set of available constraints that are applied to model real conditions are set up by three general parameters:

- *Status* Specifies if the constraint will be "turned on" or "turned off," that is, whether or not it will be included in the instance.
- *Penalty type* Specifies if the constraint will be considered as hard or soft in the cost function.
- *Weight* Specifies the penalty value which will be added to the cost function if the constraint is not fulfilled.

Based on the values defined for these parameters, a total of 27 conditions can be modeled by the instance generator, such as assign rooms to classes, avoid clashes of resources, limit the weekly workload of a teacher, and allocate a class in a specific type of laboratory. The full set of these conditions is described in Appendix 1.

3.1.3 Pre-assignment

The instance generator allows the pre-assignment of *times* and *resources* to the *events* to be scheduled, thus reducing the amount of assignments required to generate the initial solution of an instance.

Time pre-assignment is performed by defining a proportion of events to be randomly assigned to a set of consecutive time slots. On the other hand, *Resource pre-assignment* is performed by defining a proportion of events to which either a teacher or a room is to be randomly assigned, according to the requirements of each course.

3.2 Setting-up the instance generator

Setting-up the parameters of the instance generator is in itself a time-consuming task. It involves selecting from a domain of possible values the ones that guarantee logical relationships between the elements of an instance. A wrong selection of these parameters can lead to the generation of defective instances with no feasible solution spaces. Therefore, to prevent the occurrence of these type of instances, the generator was enabled to handle two types of errors: (1) insufficiency of resources and (2) over-constraining.

3.2.1 Insufficiency of resources

To ensure that it is possible to allocate the required resources to all events, the generator first calculates the total demand for each type of teacher and room resources. Then, according to the maximum weekly workload defined for each type, it generates the minimum number of teachers and rooms that are necessary to balance supply and demand of resources.

3.2.2 Over-constraining

To avoid setting conflicts between constraints, the generator incorporates the set of particular conditions (described in

Appendix 1) to instances in a hierarchical way. Conditions that define the basic assignment tasks are defined first and included as hard constraints. These are: (1) *Assign teachers*, (2) *Assign rooms* and (3) *Assign times* to all events; (4) *Avoid clashes* of resources; (7) *Prefer teachers* and (8) *Prefer rooms* to be allocated according the courses requirements; ensure (10) *Teachers stability*, (11) *Rooms stability* and (12) *Courses stability*; avoid assigning teachers and curricula out of their respective (16) *Working shifts* and (17) *Study shifts*. Next, the rest of the available conditions are further included (randomly as hard or soft constraints), one by one, only if their inclusion does not cause conflict with the ones previously defined.

4 CB-CTT instance dataset

The CB-CTT instance generator, described in the previous section, was designed to create a diverse set of instances, in both size and constraint density. As explained, it works in two stages: The first stage defines the elements of an instance (times, resources, and events), while the second stage defines the logical relationships between those elements (constraints).

As a means of defining the experimental space without incurring logical errors, in this study the parameters of the CB-CTT instance generator were set according to the following five-step process, based on the guidelines provided by Barr et al. (1995) for performing computational experiments.

4.1 Definition of relevant parameters

The instance generator provides a set of 152 parameters that can be tuned to define a CB-CTT instance. Twenty-six parameters are related to *times*, *resources* and *events*, and 126 are related to *constraints*. In order to generate a diverse CB-CTT dataset, all of the 152 parameters were considered as relevant and required to be set in the instance generation process.

4.2 Setting boundaries for the parameter values

The values of the parameters were limited to represent conditions that are commonly found in real university timetabling problems, thus specifying the domain of the random values to be generated. For example, the domain of values for the parameter <*Terms of the curricular plan*>, which defines the number of terms of a curricular plan, was set to {8, 9, 10, 11}. Therefore, we generated curricular plans that range from a duration of 8 to 11 terms (i.e., semesters, trimesters, or quarters).

4.3 Defining instance scenarios

A structure for analyzing the effects that different combinations of parameter values can have over the hardness of an instance can be constructed by defining instance scenarios. Therefore, the domain of the parameters was partitioned into groups of values that were expected to exhibit different empirical hardness. For example, the diversity of the classrooms required by the courses was varied by defining three values for the parameter *Types of classrooms* (<*Types of classrooms*>: <1>, <2>, or <3>), since it is known that the allocation of more diverse sets of resources requires the application of a higher number of constraints.

4.4 Generating instances

Once all domains and scenarios for the parameters were defined, pseudo-random generators were applied to produce all required values from a uniform probability distribution. We generated 10,000 instances, validated and categorized as described next.

4.5 Verifying consistency and diversity of the instances

As described in Sect. 2.2.2, the XHSTT data format was proposed in 2011 as an international standard to represent timetabling instances from different countries. However, at this time, not many solvers have been proposed to support this format. In the current state of the art, two types of XHSTT instance solvers can be distinguished: (1) solvers proposed to *generate* initial solutions and (2) solvers proposed to *improve* initial solutions. This paper focuses on the first type; therefore, the *KHE timetabling engine* (Kingston 2016)—a solver that has proved to be effective for generating initial solutions in current research works (Kheiri et al. 2012; Brito et al. 2012; Fonseca and Santos 2014; Kristiansen et al. 2015; Fonseca et al. 2017)—was employed to accomplish two goals: (1) validate the structural consistency of the generated instances and (2) measure their empirical hardness.

If the KHE timetable engine is able to generate an initial solution for an instance, then this validates the consistency of such an instance. As part of the solution, the KHE timetable engine returns a report describing both the total cost of violations of hard constraints (*infeasibility value*) and the total cost of violations of soft constraints (*objective value*). To measure the empirical hardness of the instances, these values were combined into a single cost, called *penalty value*, that assigns a cost value of 1000 to each violation of a hard constraint and a cost value of 1 to each violation of a soft constraint.

The penalty value was used to group the generated instances in order to evaluate the diversity of the dataset. Based on an empirical statistical analysis of histogram plots, instances were grouped into five categories according to their penalty values:

- *Very easy* Instances with a penalty value between [0–1000).
- *Easy* Instances with a penalty value between [1000–10,000).
- *Medium* Instances with a penalty value between [10,000–50,000).
- *Hard* Instances with a penalty value between [50,000–100,000).
- *Very hard* Instances with a penalty equal to or greater than 100,000.

From the 10,000 generated instances, and to avoid dealing with problems related to unbalanced datasets, 1200 instances were selected so that each category would be equally represented. Thus, a total of 6000 instances were included in our final dataset.

5 CB-CTT metrics

In the previous section, the *penalty value* produced by the KHE timetabling engine was used to determine the diversity of the 6000 CB-CTT instances. In this section, we present the set of features that were formulated to describe their empirical hardness.

Formulating features to characterize instances on a problem domain is a laborious task which requires a combination of general and domain-specific knowledge. However, once a descriptive set of features has been formulated, it can be used by the research community as a starting point to propose new solving methods and drive research forward (Kotthoff 2014).

Features within the combinatorial optimization field can be exploited in different ways, for example, to choose the best algorithm to solve a given instance (Kanda et al. 2016; Messelis and De Causmaecker 2014), to guide the selection of the best operator to be applied at each iteration of a hyper-heuristic approach (Soria-Alcaraz et al. 2014; Soghier and Qu 2013), or to improve the searching strategies of meta-heuristics (Teoh et al. 2015; Pillay 2014). However, their effectiveness in performing these tasks cannot be *a priori* guaranteed, since it depends on their descriptive capability over the problem space.

Currently, one set of features that has been proposed to characterize instances within the university timetabling domain is the set of 32 features formulated by Smith-Miles and Lopes (2011). The set considers four types of features: 3 size-related features, 2 landmarking features, 21 graph-coloring features, and 6 timetabling features. However, as it is mainly based on the properties of conflict graphs $G(V, E)$ (where V is a set of vertices corresponding to *events* that need to be timetabled and E is a set of edges connecting any two vertices when the events cannot occur at the same time), the set is not suitable to describe the full set of constraints considered by our CB-CTT instance generator.

To overcome this limitation this work proposes four types of features to characterize timetabling instances within the CB-CTT domain. As each type measures different properties of the problem (e.g., solution space, resource conflicts), we refer to them as *metrics*.

The proposed CB-CTT metrics consist of: (1) 74 *simple metrics* that provide a basic notion of the size of the elements that compose an instance; (2) 44 *solution space metrics* that, based on counting functions, measure the size of possible combinations for task scheduling and resource allocation; (3) 17 *feature ratios* that calculate numerical relationships between *simple* and *solution space* metrics; and iv) 14 *constraint density indexes* defined by Kingston (2016). Therefore, the set consists of a total of 149 CB-CTT metrics or features that are described next.

5.1 Simple metrics

We define *simple metrics* as observable data that provide a basic notion of the size of an instance without the need for performing further calculations, for example, total number of classes, total number of teachers, total number of rooms, total number of constraints, etc. For explanation purposes, the set of simple metrics is presented in two tables. Table 2 presents four metrics used to describe each of the 14 types of XHSTT constraints considered by our instance generator[1] (a total of 56 metrics), while Table 3 presents 18 metrics related to the general structure of a CB-CTT instance.

5.2 Solution space metrics

At its basic formulation, the CB-CTT problem is a combinatorial problem which requires performing two types of assignments: (1) assigning lectures to time slots (task scheduling) and (2) assigning resources to lectures (resource allocation). Although these assignment tasks are performed simultaneously and must not be considered independent stages of the solution process, they work in different solution spaces: the first one related to *time slots* and the second one to *resources*. In this paper, we refer to them with two shortened terms: *scheduling* (for task scheduling) and *allocation* (for resource allocation).

[1] From the available 16 types of XHSTT constraints described in Table 1, Distribute Split Events Constraint and Order Events Constraint are not used by the instance generator.

Table 2 Simple constraint-related metrics used to describe each one of the 14 types of constraints used by the CB-CTT generator

	Metric	Description
1	Constraints	Number of constraints of a certain type that are applied to an instance
2	Points of application	Number of possible violations that need to be evaluated to verify the fulfillment of a type constraint
3	Soft penalty value	Soft penalty value that is going to be added to the cost function if a certain type of constraint fails to fulfill all points of application
4	Hard penalty value	Hard penalty value that is going to be added to the cost function if a certain type of constraint fails to fulfill all points of application

Table 3 Simple structure-related metrics used to describe the size of the basic elements of an instance

	Metric	Description
1	Time slots	Available number of time slots that are considered to solve an instance
2	Days	Number of days in which time slots are grouped
3	Shifts	Number of shifts in which time slots are grouped
4	Knowledge areas	Number of categories that defines the expertise areas of the teachers (e.g., math, engineering, management) required by the classes
5	Types of rooms	Number of room types required by the classes
6	Teachers	Available number of teachers that can be allocated
7	Full-time teachers	Available number of full-time teachers that can be allocated
8	Part-time teachers	Available number of part-time teachers that can be allocated
9	Rooms	Available number of rooms in which classes can be allocated
10	Curricula	Number of curricula in which classes are grouped
11	Curriculum times	Maximum number of weekly time slots that a curriculum can be allocated
12	Events	Total number of classes to be scheduled
13	Total events duration	Total duration of the classes to be scheduled
14	Preassigned times	Total number of time pre-assignments
15	Total resource requests	Total number of resources requests required by the classes
16	Preassigned resources	Total number of resource pre-assignments
17	Total points of application	Total points of application to be fulfilled to solve an instance
18	Total penalty value	Sum of the penalty values of all points of application

According to Ochiai et al. (2016), the optimization process performed by any heuristic is a matter of searching tasks within feasible solution spaces. Therefore, describing the features of such spaces could lead to the development of better solution methods. To measure the dimensions of both solution spaces, 44 metrics (16 scheduling space metrics and 28 allocation space metrics), based on the concept of counting functions, were formulated.

5.2.1 Scheduling space metrics

Within the CB-CTT context, *scheduling* consists of splitting the weekly duration of a class into a set of lectures to be appointed according to a set of time-related constraints. The number of ways in which this process can be performed defines the *scheduling space* of an instance. For example, consider a 3-h literature class to be scheduled on a weekly basis. There are three possible configurations for scheduling its lectures: a single 3-h lecture (*3*), a combination of 2-h and 1-h lectures (*2-1*), and a set of three 1-h lectures (*1–1–1*). If eight available time slots per day, from Monday to Friday, are available to schedule this class, then the number of possible ways (without repetition) to perform the scheduling of each configuration is (*3*): 30; (*2–1*): 1270; (*1–1–1*): 4290. Hence, if no time constraints are further applied, the *non-constrained scheduling space* of this literature class is calculated by the sum of *non-constrained counting functions* that define the possible scheduling combinations for these configurations that, for this example, would be 5590 possible ways.

Every time a time-related constraint is applied to an instance, its scheduling space is reduced by an amount that depends on the particular conditions modeled by the constraint. In this work, the reduction of each constraint over the *non-constrained scheduling space* is quantified in three ways:

– *Space* Counts the remaining size of the scheduling space after applying *only* a certain type of constraint.

- *Removed* Counts the number of combinations that are removed from the scheduling space after applying *only* a certain type of constraint.
- *Percentage* Calculates the percentage in which the scheduling space is reduced after applying *only* a certain type of constraint.

In summary, the set of scheduling space metrics consist of: 1 *non-constrained scheduling space* metric that aggregates the initial *scheduling spaces* of the classes of an instance without limiting the number of ways in which the scheduling task can be performed, 12 scheduling reduction metrics that measure the shrinkage of the scheduling space due to each one of the applied XHSTT constraints (described in Table 1) related to the scheduling space (i.e., SEC, PTC, SPEC and LEC), and 3 scheduling constrained metrics that measure the overall net effect of the constraints on reducing the initial scheduling space (i.e., *constrained scheduling space, constrained scheduling removed* and *constrained scheduling percentage*). Out of the 44 solution space metrics, 16 are scheduling space metrics.

5.2.2 Allocation space metrics

Within the CB-CTT context, allocation is the assignment of the available resources to the lectures which require them. The number of ways (defined by a set of resource-related constraints) in which this assignment can be performed is what defines the *allocation space*.

For example, consider a class to be scheduled four times a week that requires a literature teacher. If there are three possible teachers to be assigned to such lectures and the defined time grid consists of twenty time slots, then the number of possible allocations for this class, which defines the *non-constrained allocation space*, is 720.

Similarly to the *scheduling space*, the *unconstrained allocation space* is reduced if any of the remaining eight resource-related constraints (i.e., PRC, ASAC, ACC, AUTC, LITC, CBTC, LBTC, LWC) described in Table 1, are applied to an instance. For example, if the condition *weekly workload of full-time teachers* is set to limit to 15 the number of hours that teachers can work per week, the allocation space would be reduced from 720 to 540.

In summary, the set of allocation space metrics consists of: 1 *non-constrained allocation space* that aggregates all the possible allocations of resources to classes, 24 allocation reduction metrics, and 3 allocation constrained metrics (i.e., *constrained allocation space*, *constrained allocation removed* and *constrained allocation percentage*). Out of the 44 solution space metrics, 28 are allocation space metrics.

5.3 Feature ratios

Ratios are simple mathematical (statistical) calculations that are used to express relationships and to perform meaningful comparisons between numerical data. Hence, in order to describe potential significant relationships between the sets of *simple* and *solution space* metrics, described above, a set of what we call *feature ratios* was formulated. Seventeen of such feature ratios were defined, and these are described in Table 4.

5.4 Constraint density

In addition to the three sets of metrics that were formulated for this study in the previous subsections, a set of 14 constraint density indexes defined by Kingston (2016), as part of the *KHE timetabling engine*, was included into our collection of CB-CTT metrics.

As defined by Kingston, the *density* of each type of XHSTT constraint is calculated by dividing the number of elements to which a constraint is applied by the number of elements to which this constraint could be applied. For example, if the constraint *Assign time constraint* (ATC) is set to require that 80 of a total of 100 lectures defined in an instance be scheduled, then the density of this constraint is calculated as: total lectures to which ATC is applied divided by the possible lectures to which ATC could be applied. In this case, the ATC density = 80/100.

The density of each constraint is calculated based on different elements, according to the condition it models. These elements could be times, resources or events of an instance.

6 Feature selection

The collection of CB-CTT metrics described in the previous section presents different properties that could be descriptive of the empirical hardness of CB-CTT instances. However, in order to verify how useful those metrics are in characterizing the CB-CTT problem space, this section presents a statistical validation based on the feature selection process shown in Fig. 1.

The main goal of any feature selection process is to reduce data dimensionality without losing much of the total information. In theory, more features should produce better learning models. But as stated by Yu and Liu (2004), in practice, excessive features lead to higher learning times and over-fitted models. Therefore, finding the optimal subset of features to enhance learning efficiency and prediction accuracy is a critical task.

To obtain the optimal subset of features for characterizing the empirical hardness of CB-CTT instances, using the generated dataset defined in Sect. 4, three sequential processes

Table 4 Description of the 17 feature ratios included in the set of CB-CTT metrics

	Metric	Description
1	Total room conflicts	Sum of the ratios of *required time slots per type of room* to *available time slots per type of room*
2	Total teacher conflicts	Sum of the ratios of *required time slots per type of teacher* to *available time slots per type of teacher*
3	Total curriculum conflicts	Sum of the ratios of *required time slots per curriculum* to *available time slots per curriculum*
4	Total resource conflicts	Sum of total room conflicts, total teacher conflicts and total curriculum conflicts
5	Average room conflicts	Average ratio of *required time slots per type of room* to *available time slots per type of room*
6	Average teacher conflicts	Average ratio of *required time slots per type of teacher* to *available time slots per type of teacher*
7	Average curriculum conflicts	Average ratio of *required time slots per curriculum* to *available time slots per curriculum*
8	Average resource conflicts	*Total resource conflicts* divided by the total number of resources
9	Weighted room conflicts	Weighted average ratio of *required time slots per room type* to *available time slots per type of room*
10	Weighted teacher conflicts	Weighted average ratio of *required time slots per teacher type* to *available time slots per type of teacher*
11	Percentage of assigned resources	*Preassigned resources* divided by the *Total resource requests*
12	Average event duration	*Total events duration* divided by the number of *Events*
13	Scheduling space shrinkage	*Constrained scheduling space* divided by the *non-constrained scheduling space* (*ATC space*)
14	Allocation space shrinkage	*Constrained allocation space* divided by the *non-constrained allocation space* (*ARC space*)
15	Average constraint penalty	*Total penalty value* of the instance divided by the *total points of application* defined by its constraints
16	Average allocation space per resource	*Constrained allocation space* divided by the number of *resources*
17	Average scheduling space per event	*Constrained scheduling space* divided by the number of *events*

Table 5 Attribute-value format to describe the generated CB-CTT instance dataset

Instances	Metrics			Performance
	M_1	$\dots$	M_{125}	
I_1	$m_{1,1}$	$\dots$	$m_{1,125}$	p_1
I_2	$m_{2,1}$	$\dots$	$m_{2,125}$	p_2
I_3	$m_{3,1}$	$\dots$	$m_{3,125}$	p_3
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
I_{6000}	$m_{6000,1}$	$\dots$	$m_{6000,125}$	p_{6000}

were applied: *instance description*, *performance prediction*, and *metrics assessment* (see Fig. 1).

6.1 Instance description

As the first step of the feature selection process, the 6000 CB-CTT generated instances in our dataset (where each instance is coded as an XML tree) were described in terms of the collection of CB-CTT metrics presented in Sect. 5. All values were scaled to have a mean value zero and a unit variance, and as a preprocessing step two types of features were removed: (1) uni-valued features (that have the same value in all instances) and (2) perfectly correlated features (that provide the same information). After this elimination, the number of metrics was reduced from 149 to 125.

The generated instances were described using the attribute-value format shown in Table 5. In this table, each row represents a CB-CTT instance I_i, where $1 \leq i \leq 6000$, and each column M_j represents a metric (or feature), where $1 \leq j \leq 125$, which has value m_{ij} for each instance. The value *Performance* indicates the empirical hardness of an instance, measured as the *penalty value* of the initial solution generated with the KHE timetabling engine.

6.2 Performance prediction

The analysis of the data summarized in the attribute-value format (see Table 5) was performed based on a regression approach that employs supervised learning methods to assess the relevance of the collection of CB-CTT metrics to predict the penalty value of the instances.

For the regression task, eight regression algorithms were tested: decision tree, AdaBoost, K-nearest neighbor, random forest, neural network, XGBoost, gradient boosting tree and support vector machine. To avoid the risk of drawing wrong conclusions from over-fitted models, their performance was measured in terms of the explained variance score (i.e., the proportion to which the models account for the variation of the learning data), of a tenfold cross-validation process.

Figure 4 shows the explained variance score (from lower to higher) of the eight learning algorithms. As observed, they exhibit a significant disparity in their mean explained vari-

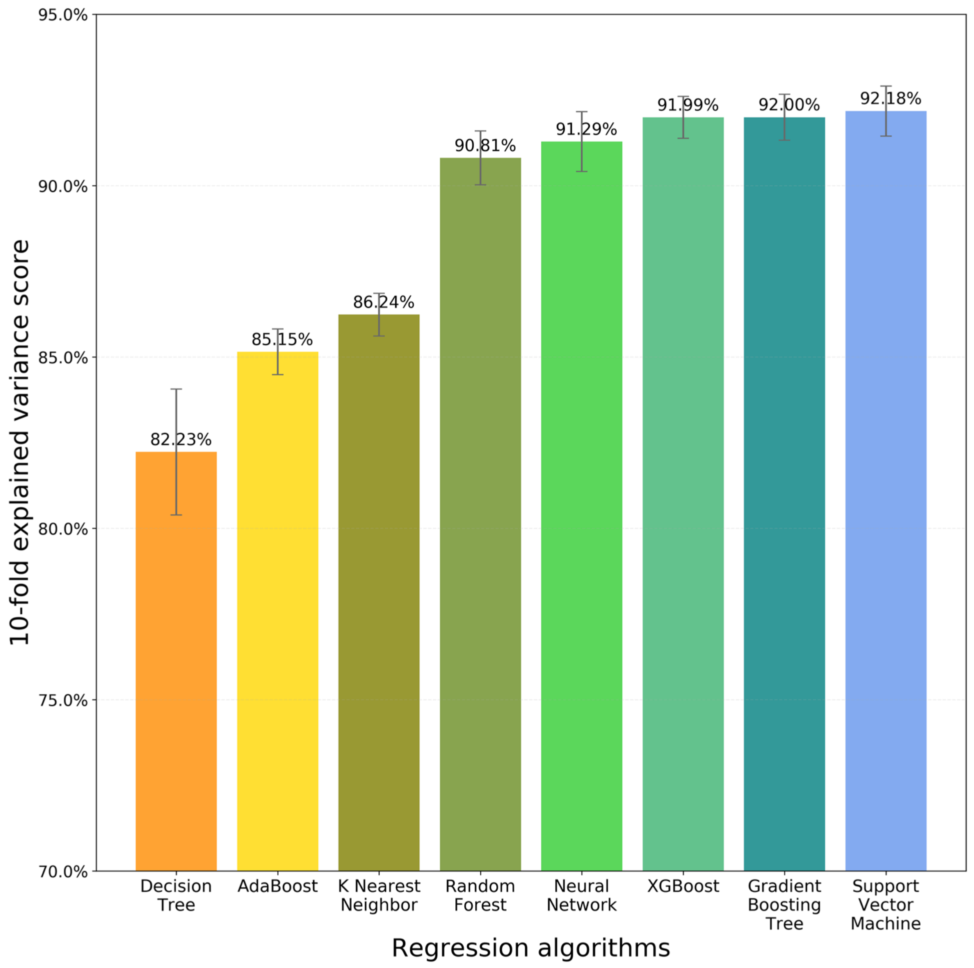

Fig. 4 Explained variance score of machine learning regressors applied to predict the penalty value of the generated XHSTT dataset using a tenfold cross-validation

ance scores that ranges from 82.23% (obtained with decision tree), to 92.18% (obtained with support vector machine). The best five of these algorithms, although different in the mathematical methods on which they are based, achieved a similar performance. This indicates that the mean explained variance score for the regression models that can be built from this CB-CTT metrics is around 91%.

6.3 Metrics assessment

As shown in Fig. 1, the final goal of the feature selection process is to find a subset of *Relevant CB-CTT metrics* able to characterize the generated instances to an equal (or almost equal) degree as the full set of *CB-CTT metrics*. Therefore, as the last step of the process, the performance of each metric on predicting the penalty value of the generated instances was assessed.

As pointed out in recent research works (Štrumbelj and Kononenko 2014; Lundberg and Lee 2017), even though learning algorithms have been adopted to build models in a wide range of applications, most of the time they are implemented as black boxes. They do not provide an understanding of the problems being modeled. Model interpretation is a problem in itself, whose difficulty depends on the method used in its construction. Some methods are easily explainable since their results can be explicitly tracked to their components, while some others, such as neural networks, require the construction of auxiliary explanation techniques (Lundberg and Lee 2017).

In this study, the components of two of the classification models that exhibited the highest cross-validated explained variance score were analyzed: (1) support vector machine (SVM) and (2) gradient boosting tree (GB tree). Both of these could be interpreted to assess the relevance of the collection of CB-CTT metrics. The interpretation of these models is described next.

6.3.1 Support vector machines (SVMs)

The metrics, or features, that could be considered as relevant from SVM-based models were obtained by applying a *Recursive Feature Elimination* method (RFE) (Guyon et al. 2002). This method follows the *Sequential Backward Generation* approach (SBG) to iteratively remove, based on a

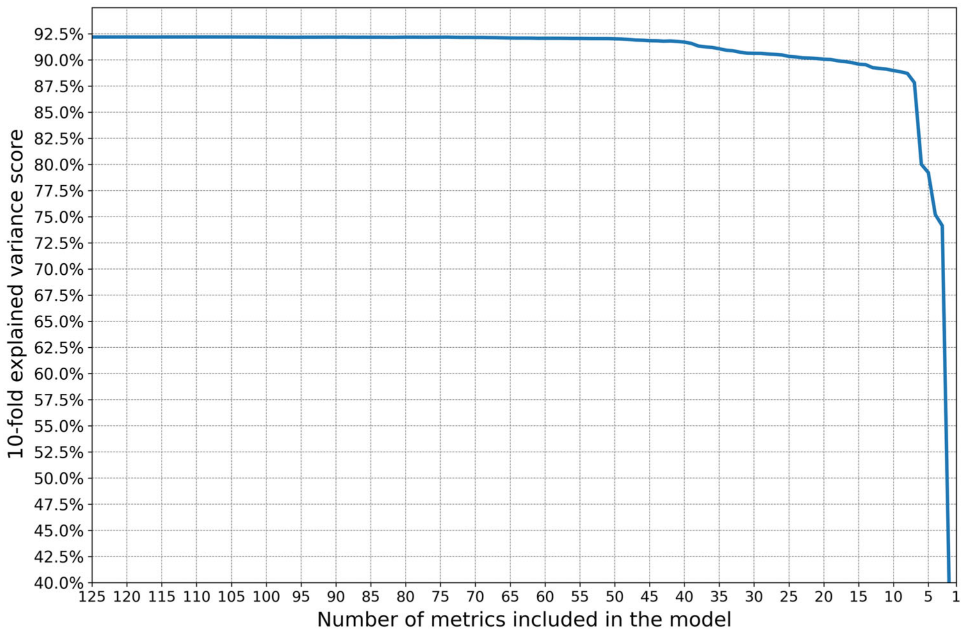

Fig. 5 Tenfold cross-validation explained variance score of SVMs built from a different number of features (CB-CTT metrics)

cross-validated evaluation, the least important of a set of features.

Figure 5 shows the cross-validated explained variance score of the SVM-based models built at each iteration of the RFE method. As observed, the explained variance score decreased significantly when only a few metrics were included in the models. Therefore, to statistically determine the minimum number of required metrics, a time series analysis was performed based on a moving average of 20 metric windows, with a convergence threshold of 0.5% for the root mean square deviation. According to these calculations, it was determined that at least 31 metrics were required to build SVM-based models as accurate as those that use the full set of metrics. Therefore, this number of metrics was chosen to perform the coefficient feature analysis described next.

At its most basic, an SVM defines a set of hyperplanes in an n-feature dimensional space to maximize the margin distance between vectors which belong to different classes. However, to predict continuous-valued outputs in regression tasks, this margin distance is often optimized with Vapnik's ε-insensitive loss functions.

As stated by Guyon et al. (2002), a straightforward method for determining the importance of a model's related features is by comparing the relative size of the n-coefficients in the hyperplanes. By performing this comparison at each one of randomly generated 10 training–testing folds, we selected the set of metrics that best predicted the penalty values of the instances.

Table 6 presents the set of 31 relevant CB-CTT metrics, organized according to the type of metric as defined in Sect. 5. Each metric is numbered in ascending order, from most to least relevant, according to the coefficients analysis of the support vector machines.

6.3.2 Gradient boosting (GB) trees

As with the SVMs, the minimum number of metrics required to build GB trees with an accuracy similar to that of a model using all metrics, was determined by the implementation of an RFE method that follows an SBG approach. By applying the same convergence threshold to the cross-validated accuracy of the GB trees (plotted in Fig. 6), we defined as 16 the minimum number of required metrics.

As stated by Hastie et al. (2005), the motivation for *boosting* is combining the outputs of many "weak" classifiers (whose error rate is only slightly better than a random guess) to produce a powerful "committee." That is, a sequence of weak classifiers $G_m(x)$, $m = 1, 2, \ldots, M$, which, combined through a weighted voting function, produce the final prediction.

When applied to regression tasks, boosting approaches combine predictors sequentially to minimize the loss function defined to quantify the prediction errors. At each iteration, error residuals are analyzed and minimized by adding a new predictor to the "committee" until error residuals become constant (i.e., without patterns).

In the applied GB tree implementation, predictors consist of individual decision trees that split according to the mean squared error defined by Friedman (2001) and are fitted to the data based on purity scores to minimize the sum of the prediction squared errors.

Table 6 Set of 31 relevant CB-CTT metrics selected from the analysis of SVMs, organized according to the type of metric

Simple	Solution space	Feature ratios
(1) Timeslots	(11) SPEC space	(17) Total room conflicts
(2) Shifts	(12) Constrained scheduling space	(18) Total curriculum conflicts
(3) Teachers	(13) ASAC space	(19) Total resource conflicts
(4) Rooms	(14) ACC space	(20) Average curriculum conflicts
(5) Total resources requests	(15) AUTC space	(25) Average allocation space per resource
(6) Total events duration	(16) Constrained allocation space	
(7) Preassigned resources	(23) PRC removed	
(8) ARC hard penalty value	(24) ASAC removed	
(9) ASAC hard penalty value	(27) AUTC removed	
(10) AUTC hard penalty value	(31) SPEC percentage	
(21) Full-time teachers		
(22) Curricula times		
(26) Types of rooms		
(28) LWC hard penalty value		
(29) ACC hard penalty value		
(30) LWC points of application		

The number indicates how relevant the metric is (1 been the most relevant)

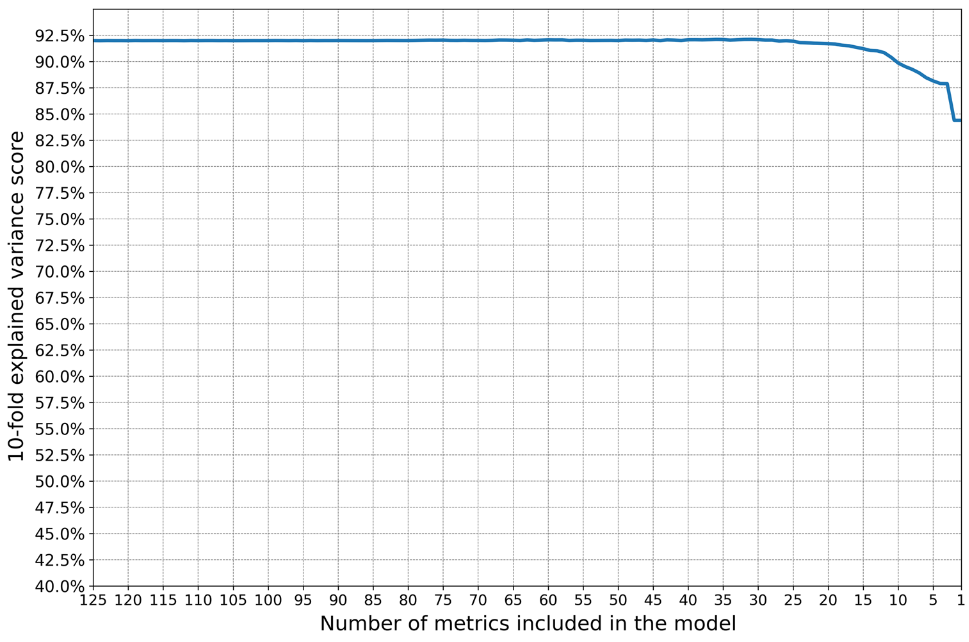

Fig. 6 Tenfold cross-validation explained variance score of GB trees constructed at each step of the RFE method

To evaluate the relevance of the metrics in the GB trees, a hierarchical approach was adopted. First, the relevance of the metrics at the individual trees was calculated by determining the weighted reduction in node purity from the split at each node. Next, the relevance values of the metrics in all trees were averaged.

Table 7 presents the set of 16 relevant metrics, organized according to the type of metric as defined in Sect. 5. Each metric is numbered in ascending order, from most to least relevant, according the node purity analysis of the GB trees.

7 Relevant CB-CTT metrics

As a result of the feature selection process presented in the previous section, three sets of *relevant metrics* were defined:

- **SVM metrics** Set of 31 relevant metrics selected from SVMs.
- **GBtree metrics** Set of 16 relevant metrics selected from GB trees.

Table 7 Set of 16 relevant metrics selected from the analysis of GB trees, organized according to the type of metric

Simple	Solution space	Feature ratios	Constraint density
(1) Total events duration	(10) PRC removed	(7) Total curriculum conflicts	(6) AUTC density
(2) Preassigned resources	(13) AUTC space	(8) Total resource conflicts	
(3) AUTC hard penalty value	(14) LEC space	(9) Average event duration	
(4) LITC hard penalty value		(11) Percentage of assigned resources	
(5) LWC hard penalty value		(12) Average curriculum conflicts	
(15) LEC points of applications		(16) Weighted room conflicts	

The number indicates how relevant the metric is (1 being the most relevant)

Table 8 Set of 9 common metrics, according to their mean observed relevance in the previous two sets

Simple	Solution space	Feature ratios
(1) Total events duration	(6) AUTC space	(4) Total curriculum conflicts
(2) Preassigned resources	(9) PRC removed	(5) Total resource conflicts
(3) AUTC hard penalty value		(7) Average curriculum conflicts
(8) LWC hard penalty value		

– **Common metrics** Set of 9 metrics that are the intersection of *SVM metrics* and *GBtree metrics*. This set of metrics is presented in Table 8 and was numbered, from most to least relevant, according to their mean observed relevance in the previous two sets. As observed, this set does not include any of the *constraint density* indexes proposed by Kingston (described in Sect. 5.4), as none of them proved to be consistently relevant in the previous two sets of metrics.

The information loss due to the dimensionality reduction of each set was assessed through a comparison with the cross-validated explained variance score of the full set of metrics. As shown in Fig. 7, the sets of *Common metrics* and *GB tree metrics* exhibited higher prediction performances in GB tree models than in SVM models. However, unlike *SVM metrics*, they also showed higher performance deviations between both regression approaches. This suggests more variable description capabilities when used to build prediction models based on a different set of learning algorithms.

As an additional analysis of the three sets of relevant features, we evaluated their robustness to characterize the empirical hardness of the instances by a less evident attribute, also related to the complexity of the instances: the *penalty value per lecture* (i.e., *penalty value* divided by *total events duration*).

Figure 8 shows the prediction performance of the regression models built from the same set of relevant metrics, except for the metric total event duration, that were used to calculate the predicted value. As observed, the cross-validated explained variance score was reduced for all sets of metrics in different degrees. The set of *Common metrics* demonstrated a high-performance variability between the two applied learning algorithms, while the performance of the set of *SVM metrics*, the lowest. Still, because it exhibited the highest performance in both regression approaches, the set of *GBtree metrics*, proved to be more robust to the defined changes.

Regarding the robustness of the applied learning algorithms, it is clear that the loss of prediction performance was, on average, less for gradient boosting trees than for support vector machines.

8 Metrics interpretation

Despite the importance of matching instance properties of academic timetabling problems with the performance of available solving methods (Rossi-Doria et al. 2002), few research works in the literature have proposed features to describe both the search space and the relative hardness of the instances (Kostuch and Socha 2004; Smith-Miles et al. 2009; Rodriguez-Maya et al. 2016). Therefore, to fill this gap, three sets of *relevant CB-CTT metrics*, defined to predict the empirical hardness of CB-CTT instances, were compared in the previous section.

Due to their combinatorial nature, similar problem instances, in terms of size and structure, often exhibit different empirical hardness (Kostuch and Socha 2004). Hence, knowing the basic parameters of an instance does not guarantee a good description of its solution space, since it mainly depends on the particular constraints and objectives that are added to the problem beyond basic feasibility constraints (Smith-Miles and Tan 2012).

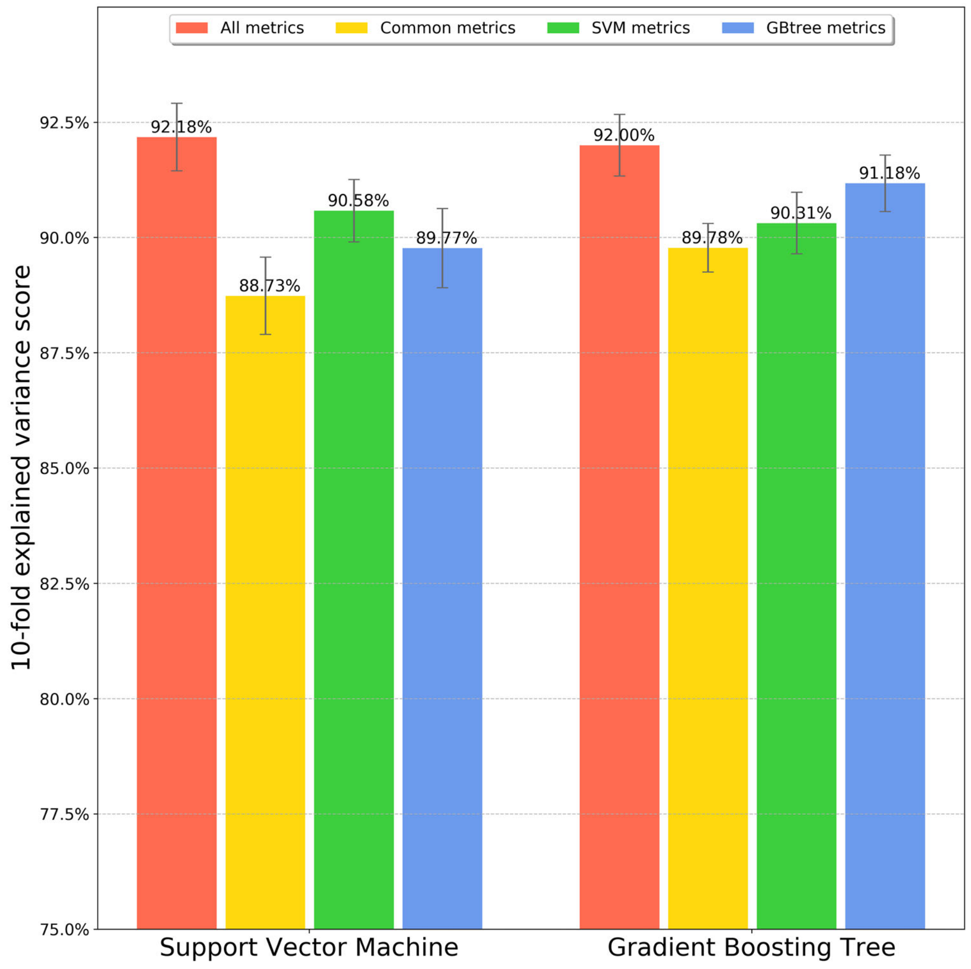

Fig. 7 Tenfold cross-validation explained variance score of GB trees and SVMs using the three defined set of metrics to predict the penalty value

As a contribution to the current state of the art, this section discusses the relationships between the proposed collection of *relevant CB-CTT metrics* and the empirical hardness of the generated CB-CTT instances according to three main aspects: (1) size, (2) scheduling (task scheduling), and (3) allocation (resource allocation).

8.1 Size-related hardness

As shown in Table 6, most of the features in the set of *SVM metrics* correspond to simple metrics, proposed to characterize the structure of CB-CTT instances as regards to its basic dimensions (e.g., number of time slots, number of teachers, number of rooms). However, only two of them are also considered relevant in the set of *GBtree metrics*: (1) *Total events duration* and (2) *Preassigned resources*.

On the one hand, *Total events duration* measures the number of time slots required to schedule the classes of an instance. On the other hand, *Preassigned resources* measures the number of resources pre-assignments that reduces the required amount of resource allocations. These metrics are ranked as the most important features of the set of *Common metrics* and are likely to be good indicators of the assignment tasks required in both solution spaces (described in Sects. 5.2.1 and 5.2.2),

8.2 Allocation hardness

As pointed out in recent surveys (Pillay 2014; MirHassani and Habibi 2013; Bouajaja and Dridi 2016; Sahargahi and Drakhshi 2016), most of the constraints that are commonly defined to formulate timetabling problems are related to allocating resources to lectures. Therefore, the fact that the majority of the metrics selected as relevant by the feature selection process are mainly related to the *allocation space* of the problem is not surprising.

The size of the *allocation space* of an instance has a crucial role in defining the hardness of an instance. This is evident because of the relevance of the metric *Constrained allocation space* (in the set of *SVM metrics*) and of other *solution space metrics* that measure the reduction in the initial number of feasible <class, resource> combinations,

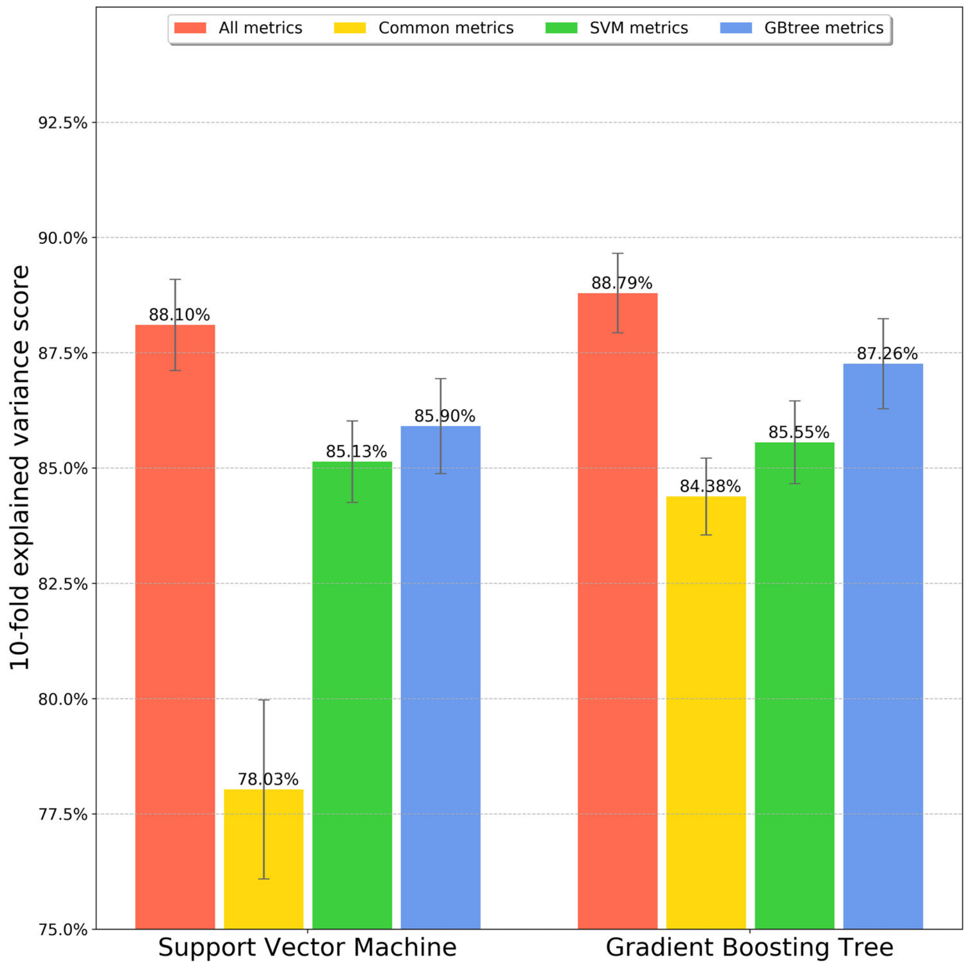

Fig. 8 Tenfold cross-validation explained variance score of GB trees and SVMs using the three defined set of metrics to predict the penalty value per lecture

due to the set of resource-related constraints applied to the instances.

Besides the basic *Avoid Clashes Constraint* (ACC) and *Avoid Split Assignment Constraint* (ASAC), three types of allocation-related constraints are recurrent on the sets of relevant metrics and proved to be descriptive of the allocation space:

- *Avoid unavailable time constraints* (AUTC) Are applied to model two conditions: *working shifts* and *study shifts*. These constraints limit the time slots in which teachers and curricula can attend classes. For example, when two shifts (morning and evening) are defined for an instance, the allocation space of teachers and curricula is reduced by half.
- *Limit workload constraints* (LWC) Are applied to model three conditions: *daily workload of full-time teachers*, *daily workload of part-time teachers*, and *daily workload of curricula*. These constraints define the maximum number of time slots per day in which both teachers and curricula can attend classes.
- *Prefer resources constraints* (PRC) Are applied to model two conditions: *prefer teacher* and *prefer room*. These constraints specify the types of teachers and rooms required for each class (event), thus reducing the set of possible resources to be allocated.

In addition to these metrics, three ratios (defined in Sect. 5.3) associated with the concept of *slackness* are included in the set of *Common metrics*: *Total curriculum conflicts*, *Total resource conflicts*, and *Average curriculum conflicts*. In each, the *slackness* (which measures the easiness of allocating a resource to the lecture that requires it) is calculated by dividing the demand of a resource by its available supply. For example, if 20-h literature lectures are to be scheduled weekly, and there is only one literature teacher with a defined weekly workload of 30 h a week, the slackness for this allocation task is calculated as 20/30. Therefore, the higher the slackness, the greater the hardness of allocating a resource.

8.3 Scheduling hardness

Similarly to the allocation space, the size of the *scheduling space*, which aggregates the number of possible assignments of lectures to time slots, proved to be a crucial feature in defining the hardness of the generated CB-CTT instances. This is evident because of the relevance of the *Constrained scheduling space* (in the set of *SVM metrics*) and of time-related metrics that measure the reduction of the initial *scheduling solution space* because of the application of two constraints:

– *Spread events constraints* (SPEC) Are applied to model two conditions: *single lecture* and *daily lecture*. On the one hand, *single lecture* requires that only one lecture of each class be scheduled per day. On the other hand, *daily lecture* requires that the lectures of a set of randomly selected classes be scheduled on a daily basis. These constraints are the main factors for the reduction in the number of valid combinations of the scheduling space, up to a percentage between 66.4% and 89.7% in half of the generated instances.
– *Link events constraint* (LEC) Is applied to model the *link events* condition, which requires that all lectures of a set of classes be scheduled simultaneously. Unlike SPEC constraints, this constraint does not reduce the scheduling space of the instances by a high proportion. However, it causes an increase in the complexity of the *layer tree* structure, which is used to handle time assignment by the KHE solver (Kingston 2016).

9 Conclusions and further research

The CB-CTT problem is a widely known combinatorial problem for which many solution approaches have been proposed. However, as observed in recent surveys, most of the current research works are focused on developing new solution approaches for existing instance benchmarks rather than on developing a formal description of the conditions that make CB-CTT instances hard to solve across the problem space.

In this paper, we proposed the implementation of a feature selection-based methodology to identify and interpret a set of features that better characterizes the hardness of CB-CTT instances within the problem space defined by a customizable instance generator. To do so, we described the generated instances using four sets of *CB-CTT metrics* that were used to predict the *penalty value* of the instances through regression models constructed from different machine learning methods.

Based on the interpretation of the most accurate empirical hardness models, three sets of relevant features were defined. As shown in Fig. 7, they demonstrated to be almost as precise as the full set of metrics in predicting the *penalty value* of the instances. Furthermore, when used to predict a different measure of empirical hardness (*penalty value per event*), two of them (*SVM metrics* and *GBtree metrics*) did not exhibit a significant performance loss (see Fig. 8).

In contrast to previous works (Smith-Miles and Lopes 2011), in which *simple* and *graph-coloring* features have been applied to characterize the instance space of CB-CTT elementary problems, our research proposes a combination of counting functions, ratios and indexes to describe the relationships between reality-like conditions that define the solution spaces of CB-CTT instances and their empirical hardnesses.

As a result of the applied feature selection methodology, two major contributions were obtained from our research work: (1) the design of a CB-CTT generator that uses the standard XHSTT format to model a wider range of real-like conditions than those addressed in previous research works, and (2) a collection of relevant metrics that could be used to characterize the empirical hardness of CB-CTT instances.

The interpretation of the relevant metrics led us to conclude that the hardness of CB-CTT instances is highly related to the percentage of reduction of the scheduling and allocation spaces (measured based on counting functions) and the slackness of resources (measured based on feature ratios). These findings may be further exploited to: (1) design new solving algorithms, (2) improve current solution methods, (3) perform sensitivity analysis on the conditions that define an instance, and (4) select the best algorithm for a particular instance.

Despite its contribution to the analysis of conditions that define the complexity of *generating* initial solutions for CB-CTT instances (formulated according to the XHSTT data format), in our methodology, the empirical hardness was defined based on the performance of one solver (KHE timetabling engine). Therefore, it remains for future research the implementation of different solving methods, currently scarce in the literature, to generalize these findings for the CB-CTT algorithm space. Moreover, to extend the application of these features within the university timetabling problem domain, further work is needed to determine whether these metrics could also be good predictors of the performance of solvers proposed to *improve* initial solutions.

Acknowledgements We gratefully acknowledge the support of CONACYT-Mexico (Reg. 618204/461410) and the helpful suggestions of the reviewers.

Conditions that can be modeled by the CB-CTT instance generator

	Condition	Description
1	Assign teachers	Classes must have an assigned teacher
2	Assign rooms	Classes must have an assigned room
3	Assign times	Classes must be scheduled for the number of time slots required by their durations
4	Avoid clashes	Resources (i.e., curricula, teachers, and rooms) must not be assigned to different lectures at the same time
5	Split theory event	Specifies the valid set of lecture configurations in which classes of theory courses (which do not require a laboratory) can be scheduled
6	Split practice event	Specifies the valid set of configurations in which classes of practice courses (which require a laboratory) can be scheduled
7	Prefer teachers	Defines the subset of classes to which teachers can be allocated
8	Prefer rooms	Defines the subset of classes to which rooms and laboratories can be allocated
9	Prefer times	Limits a randomly selected percentage of classes to be scheduled only on a user-defined set of days
10	Teachers stability	Requires that all lectures derived from a class must be allocated to the same teacher
11	Rooms stability	Requires that all lectures derived from a class must be allocated to the same room
12	Courses stability	Requires that all lectures derived from a class which requires both a classroom (for theory instruction) and a laboratory (for practice activities) are allocated the same teacher
13	Single lecture	Requires that for each class only one lecture be scheduled per day
14	Daily lecture	Requires that a randomly selected percentage of classes be scheduled in a daily basis, within a user-defined set of days.
15	Link events	Requires that all lectures derived from a set of classes be scheduled simultaneously. This set of classes is defined by randomly selecting a class from each one of the active terms in the curricular plan
16	Working shifts	Requires that teachers be allocated only in the set of time slots defined by their work shifts
17	Study shifts	Requires that curricula be allocated only in the set of time slots defined by their study shifts
18	Idle times of part-time teachers	Specifies the range of daily idle time slots that part-time teachers must have
19	Idle times of curricula	Specifies the range of daily idle time slots that curricula must have
20	Busy days of full-time teachers	Specifies the number of working days that full-time teachers must give classes
21	Busy days of part-time teachers	Specifies the number of working days that part-time teachers must give classes
22	Busy days of curricula	Specifies the number of days that curricula must be assigned classes
23	Daily workload of full-time teachers	Specifies the number of daily time slots that full-time teachers must give classes
24	Daily workload of part-time teachers	Specifies the number of daily time slots that part-time teachers must give classes
25	Daily workload of curricula	Specifies the number of daily time slots that curricula must attend classes
26	Weekly workload of full-time teachers	Limits the number of weekly time slots that full-time teachers can be allocated
27	Weekly workload of part-time teachers	Limits the number of weekly time slots that part-time teachers can be allocated

References

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G., & Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, *1*, 9–32.

Bonutti, A., De Cesco, F., Di Gaspero, L., & Schaerf, A. (2012). Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, *194*, 59–70.

Bouajaja, S., & Dridi, N. (2016). A survey on human resource allocation problem and its applications. *Operational Research*, *17*, 1–31.

Brito, S. S., Fonseca, G. H., Toffolo, T. A., Santos, H. G., & Souza, M. J. (2012). A SA-VNS approach for the high school timetabling problem. *Electronic Notes in Discrete Mathematics*, *39*, 169–176.

Cooper, T. B., & Kingston, J. H. (1995). The complexity of timetable construction problems. *International conference on the practice and theory of automated timetabling* (pp. 281–295). Berlin: Springer.

Di Gaspero, L., McCollum, B., & Schaerf, A. (2007). The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Technical report. Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1. 0, Queen's University, Belfast, United Kingdom.

DMPP Group UoT (2014). *Overview XHSTT-2014 (Instances and best solutions)*. https://www.utwente.nl/en/eemcs/dmmp/hstt/archives/XHSTT-2014/overview.html. Retrieved on May 5, 2017.

Fonseca, G. H., & Santos, H. G. (2014). Variable neighborhood search based algorithms for high school timetabling. *Computers & Operations Research*, *52*, 203–208.

Fonseca, G. H., Santos, H. G., Carrano, E. G., & Stidsen, T. J. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research*, *262*, 28–39.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*, 1189–1232.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*, 389–422.

Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). The elements of statistical learning: Data mining, inference and prediction. *The Mathematical Intelligencer*, *27*, 83–85.

Kanda, J., de Carvalho, A., Hruschka, E., Soares, C., & Brazdil, P. (2016). Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features. *Neurocomputing*, *205*, 393–406.

Kheiri, A., Ozcan, E., & Parkes, A. J. (2012). Hysst: Hyper-heuristic search strategies and timetabling. In: *Proceedings of the ninth international conference on the practice and theory of automated timetabling (PATAT 2012)* (pp. 497–499). Citeseer.

Kingston, J. H. (2016). *A software library for high school timetabling*. http://www.it.usyd.edu.au/~jeff/khe/. Retrieved on November 2016.

Kostuch, P., & Socha, K (2004). Hardness prediction for the university course timetabling problem. In: *European conference on evolutionary computation in combinatorial optimization* (pp. 135–144). Springer.

Kotthoff, L. (2014). Algorithm selection for combinatorial search problems: A survey. *Ai Magazine*, *35*, 48–60.

Kristiansen, S., Sørensen, M., & Stidsen, T. R. (2015). Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, *18*, 377–392.

Leyton-Brown, K., Nudelman, E., & Shoham, Y. (2002). Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In: *International conference on principles and practice of constraint programming* (pp. 556–572). Springer.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In: *Advances in neural information processing systems* (pp. 4768–4777).

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., et al. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, *22*, 120–130.

Messelis, T., & De Causmaecker, P. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, *233*, 511–528.

MirHassani, S., & Habibi, F. (2013). Solution approaches to the course timetabling problem. *Artificial Intelligence Review*, *39*, 1–17.

Ochiai, H., Kanazawa, T., Tamura, K., & Yasuda, K. (2016). Combinatorial optimization method based on hierarchical structure in solution space. *Electronics and Communications in Japan*, *99*, 25–37.

Pillay, N. (2014). A survey of school timetabling research. *Annals of Operations Research*, *218*, 261–293.

Post, G., Kingston, J. H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., et al. (2014). Xhstt: An xml archive for high school timetabling problems in different countries. *Annals of Operations Research*, *218*, 295–301.

Rodriguez-Maya, N., Flores, J. J., & Graff, M. (2016). Predicting the RCGA performance for the university course timetabling problem. In: *International symposium on intelligent computing systems* (pp. 31–45). Springer.

Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L. M., Knowles, et al. (2002). A comparison of the performance of different metaheuristics on the timetabling problem. In: *International conference on the practice and theory of automated timetabling* (pp. 329–351). Springer.

Sahargahi, V., & Drakhshi, M. (2016). Comparing the methods of creating educational timetable. *International Journal of Computer Science and Network Security (IJCSNS)*, *16*, 26.

Smith-Miles, K., James, R., Giffin, J., & Tu, Y. (2009). *Understanding the relationship between scheduling problem structure and heuristic performance using knowledge discovery* (p. 3). LION: Learning and Intelligent Optimization.

Smith-Miles, K., & Lopes, L. (2011). Generalising algorithm performance in instance space: A timetabling case study. In: *International conference on learning and intelligent optimization* (pp. 524–538). Springer.

Smith-Miles, K., & Tan, T. T. (2012). Measuring algorithm footprints in instance space. In: *2012 IEEE congress on evolutionary computation (CEC)* (pp. 1–8). IEEE.

Soghier, A., & Qu, R. (2013). Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. *Applied Intelligence*, *39*, 438–450.

Soria-Alcaraz, J. A., Ochoa, G., Swan, J., Carpio, M., Puga, H., & Burke, E. K. (2014). Effective learning hyper-heuristics for the course timetabling problem. *European Journal of Operational Research*, *238*, 77–86.

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, *41*, 647–665.

Teoh, C. K., Wibowo, A., & Ngadiman, M. S. (2015). Review of state of the art for metaheuristic techniques in academic scheduling problems. *Artificial Intelligence Review*, *44*, 1–21.

Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, *5*, 1205–1224.