Noname manuscript No.
(will be inserted by the editor)

Benchmarking Curriculum-Based Course Timetabling: Formulations, Data Formats, Instances, Validation, Visualization, and Results

Alex Bonutti · Fabio De Cesco · Luca Di Gaspero · Andrea Schaerf

Abstract We propose a set of formulations for the Curriculum-Based Course Timetabling problem, with the aim of "capturing" many real-world formulations, and thus encouraging researchers to "reduce" their specific problems to one of them, gaining the opportunity to compare and assess their results. This work is accompanied by a web application that maintains all the necessary *infrastructures* for benchmarking: i.e., validators, data formats, instances, reference scores, lower bounds, solutions, and visualizers. All instances proposed here are based on real data from various universities, and they represent a variety of possible situations.

1 Introduction

Course timetabling (CTT) consists of the weekly scheduling of the lectures of a set of university courses within a given number of rooms and time periods, satisfying various constraints due to *conflicts* and other features [26]. CTT is a practically-important and widely-studied problem; nevertheless, looking at the research literature on the problem, we see that many papers create a new formulation (see e.g. [1,9]) and often do not consider the previously-defined ones. This is due to the indisputable truth that, like for other timetabling problems, it is impossible to write a CTT formulation that suits all cases: every institution (and even every department) has its own rules, features, costs, and fixations.

On the other hand, when we think about research, measurability is a fundamental issue, and thus this inconvenient situation should not be taken as an excuse to refrain from performing fair comparisons and, more generally, from assessing (or benchmarking) the absolute quality of a solution method proposed.

A. Bonutti, L. Di Gaspero, and A. Schaerf
DIEGM, University of Udine
via delle Scienze 208, I-33100, Udine, Italy
E-mail: alex.bonutti@virgilio.it, l.digaspero@uniud.it, schaerf@uniud.it

F. De Cesco
EasyStaff S.r.l.
via Oderzo 1, I-33100, Udine, Italy
E-mail: fabio@easystaff.it

The International Timetabling Competitions, ITC2002 in 2002-03 and ITC2007 in 2007-08, have been organized specifically with the aim of creating the common ground for comparison [19]. The success of the competitions confirms that there is a wide awareness in the community of the need for such ground.

Among the different versions of CTT, one distinguishing feature is whether student enrollments are available (and reliable) or not. In the negative case, conflicts between courses are set according to the *curricula*, which can be either published by the university or designed based on experience from previous years. In this work, we focus on this latter case in which conflicts are based on curricula, and the resulting problem is named Curriculum-Based Course Timetabling (CB-CTT).

The CB-CTT problem, in one of its possible formulations, has been used as one of the tracks of ITC2007, namely track 3. The competition formulation of CB-CTT have been designed with the twofold objective of being, on the one side, very realistic and, on the other side, simple and general enough to attract many researchers. In fact, it has been created by starting form a real formulation and stripping out features and cost components in a careful way so as to maintain the general flavor of the problem but removing a lot of overwhelming details. Despite this effort, the competition formulation of CB-CTT has been criticized by some of the participants for not being well-designed, in the sense that the cost components used are not the most important or they do not penalize the right patterns.

In order to answer to these type of (reasonable) critics and to push forward the spirit of the competition, we propose here a larger set of possible formulations for CB-CTT. Our overall objective is thus to create a *portfolio* of "standard" versions of the problem that could be accepted by a larger community of researchers. Thus also trying to help in bridging the gap between theory and practice [17].

Obviously, we do not believe that we could "hit" exactly the formulation adopted by others; nevertheless, we hope to make simpler the "reduction" process from *ad hoc* formulations, considered by researchers for their institution, to an already existing one. The reduction would allow the researcher to work on both her/his formulation and the "standard" one, for which results are already available and thus the assessment process would be much easier. For example, the work of Nurmi and Kyngäs [15] goes into this direction, by translating one of our formulations to their problem so as to test and compare their algorithm on our instances.

In addition, if this portfolio starts to get a footing, new formulations could be added, by others or by ourselves, based on the feedback obtained from the timetabling community. The existence of standard formulations should also stimulate the application of other techniques and the design of new ones that could compete with the existing ones for the best results.

In order to give a chance to this ambitious plan to become effective, we provide along with the problem formulations all the necessary *infrastructures* for benchmarking: i.e., validators, data formats, instances, scores, lower bounds, solutions, and visualization tools. In fact, we believe that all the aforementioned tools are mandatory in order to foster the necessary sharing and cross-fertilization [27]. For example, the validator is needed to double-check that all the information has been made clear and unambiguous; and the solutions (along with the corresponding scores) provide a set of reference results, which in our opinion are very useful for a first evaluation of the own solutions of the perspective users.

This paper is therefore accompanied by a web application (`http://tabu.diegm.uniud.it/ctt`) from which many operations can be performed, as explained in Sec-

tion 4; for example, all instances can be downloaded and solutions can be validated through the web application. Moreover, all updates and news about the problem will be posted there.

2 Problem Definition

The basic features of CB-CTT are presented in the ITC2007 web site and in the corresponding technical report [10]. However, the model has to be extended in order to include the features related to the addition components. In order to make this paper self-contained, we present here the full model. The problem consists of the following basic entities:

Days, Timeslots, and Periods. We are given a number of *teaching days* in the week (typically 5 or 6). Each day is split into a fixed number of *timeslots*, which is equal for all days. A *period* is a pair composed of a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots.

Courses and Teachers. Each course consists of a fixed *number of lectures* to be scheduled in distinct periods, it is attended by a given *number of students*, and is taught by a *teacher*. For each course there is a minimum number of days that the lectures of the course should be spread in, moreover there are some periods in which the course cannot be scheduled.

Rooms. Each *room* has a *capacity*, expressed in terms of number of available seats, and a *location* expressed as an integer value representing a separate building. Some rooms may not be suitable for some courses (because they miss some equipment).

Curricula. A *curriculum* is a group of courses such that any pair of courses in the group have students in common. Based on curricula, we have the *conflicts* between courses and other soft constraints.

Curricula are published by the university in the *Student Guide*. However, the information published in the guide is usually expressed in terms of *mandatory* and *optional* courses. In order to obtain the curricula in the form proposed here, some preprocessing is necessary. In particular, the person responsible for the timetabling, mainly based on experience of previous years, has to divide optional courses in *groups* that are normally taken together by the students. Each group of optional course, together with the mandatory ones, forms a curriculum in our meaning.

The solution of the problem is an assignment of a period (day and timeslot) and a room to all lectures of each course.

We split the cost components into two sets: the basic ones, which belong to all formulations and constitute the core of CB-CTT, and the optional ones, which are considered only in some formulations.

2.1 Basic Cost Components

Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods. A violation occurs if a lecture is not scheduled or two lectures are in the same period.

Conflicts: Lectures of courses in the same curriculum *or taught by the same teacher* must be all scheduled in different periods. Two conflicting lectures in the same period represent one violation. Three conflicting lectures count as 3 violations: one for each pair.

RoomOccupancy: Two lectures cannot take place in the same room in the same period. Two lectures in the same room at the same period represent one violation. Any extra lecture in the same period and room counts as one more violation.

Availability: If the teacher of the course is not available to teach that course at a given period, then no lecture of the course can be scheduled at that period. Each lecture in a period unavailable for that course is one violation.

RoomCapacity: For each lecture, the number of students that attend the course must be less than or equal the number of seats of all the rooms that host its lectures.

All the basic cost components are *hard constraint* for all formulations, except for RoomCapacity which is *soft*, and each student above the capacity counts as 1 point of penalty. The weight for this component is 1 in all formulations, and it can be interpreted as the basic unit of penalty.

2.2 Optional Cost Components

MinWorkingDays: The lectures of each course must be spread into the given minimum number of days. *Each day below the minimum counts as 1 violation.*

IsolatedLectures (Compactness v. 1): Lectures belonging to a curriculum should be adjacent to each other (i.e., in consecutive periods). For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day. *Each isolated lecture in a curriculum counts as 1 violation.*

Windows (Compactness v. 2): Lectures belonging to a curriculum should not have time windows (i.e., periods without teaching) between them. For a given curriculum we account for a violation every time there is one windows between two lectures within the same day. *Each time window in a curriculum counts as many violation as its length (in periods).*

RoomStability: All lectures of a course should be given in the same room. *Each distinct room used for the lectures of a course, but the first, counts as 1 violation.*

StudentMinMaxLoad: For each curriculum the number of daily lectures should be within a given range. *Each lecture below the minimum or above the maximum counts as 1 violation.*

TravelDistance: Students should have the time to move from one building to another one between two lectures. For a given curriculum we account for a violation every time there is an *instantaneous move*: two lectures in rooms located in different building in two adjacent periods within the same day. *Each instantaneous move in a curriculum counts as 1 violation.*

RoomSuitability: Some rooms may be not suitable for a given course because of the absence of necessary equipment (projector, amplification, ...). *Each lecture of a course in an unsuitable room counts as 1 violation.*

DoubleLectures: Some courses require that lectures in the same day are grouped together (double lectures). For a course that requires grouped lectures, every time there is more than one lectures, a lecture non-grouped to another is not allowed.

Problem Formulation: Cost Component	UD1	UD2	UD3	UD4	UD5
Lectures	H	H	H	H	H
Conflicts	H	H	H	H	H
RoomOccupancy	H	H	H	H	H
Availability	H	H	H	H	H
RoomCapacity	1	1	1	1	1
MinWorkingDays	5	5	—	1	5
IsolatedLectures	1	2	—	—	1
Windows	—	—	4	1	2
RoomStability	—	1	—	—	—
StudentMinMaxLoad	—	—	2	1	2
TravelDistance	—	—	—	—	2
RoomSuitability	—	—	3	H	—
DoubleLectures	—	—	—	1	—

Table 1 Problem formulation descriptions

Two lectures are grouped if they are both adjacent and in the same room. *Each non-grouped lecture counts as 1 violation.*

Notice that IsolatedLectures and Windows are two different ways to account for the curriculum compactness, which is meant to reflect the time wasted by students for travelling to the university and waiting between lectures. As an example of the difference, a single lecture in a day is penalized by IsolatedLectures but not by Windows. As another example, two split lectures in a day are penalized always as 2 violations by IsolatedLectures, whereas they are penalized by Windows proportionally to the width of the time window between them.

2.3 Formulations

With the term *formulation* here we mean a specific set of cost components, along with the weights assigned to each of them. The weights are necessary because we consider only weighted-sum single-objective functions, as multi-objective formulations and Pareto optimality issues are out of the scope of this work.

Up to now, only two formulations have been proposed: The first one has been introduced in our previous work [12,13] and it has been used also by Burke *et al* [4,5] with an IP-based approach. The second one has been specifically designed for ITC2007.

We name these formulations as UD1 and UD2, respectively (UD for Udine), and we introduce three new ones, UD3, UD4, and UD5. Among all possible formulations (exponentially many w.r.t. the number of cost components), these ones have been designed with the aim of capturing different university settings. Once the "machinery" is consolidated, new formulations can be very easily added in response to stimuli from the community.

Table 1 presents the formulations in terms of which cost components they include. For each pair formulation/component we write in the cell the weight associated to the component in the formulation, or 'H' if the component is hard. A dash sign '—' means that the component is not included in the formulation. The horizontal line separates the basic components from the optional ones.

It is clear that the weight used in the formulations are quite arbitrary. They are used to create common single-objective formulations, but it is understood that multi-objective ones would be more appropriate in many cases.

3 Problem Data

We present in this section the data needed to share the problem with other researchers, and thus to compare the algorithms.

3.1 Instances

The set of instances includes the 4 instances used in [12], called `test1`, ..., `test4`, and the 21 proposed for the competition, called `comp01`, ..., `comp21`, which are all real cases taken mainly from our university. For many of those instances, the additional information needed in the new formulations was not available, not reliable, or not meaningful. For example, for many instances, rooms were all in the same building, therefore the travel distance was always 0. In these cases and in similar ones, some information has been added in an arbitrary way. Nevertheless, most of the information is composed of real data.

Recently we have added to the portfolio 7 new real world instances, which come mainly from other Italian universities, and thus represent new cases to be dealt with. We name these instances as `DDS1`, ..., `DDS7`. Finally, we have one small instance, called `toy`, that in our case turned out very useful for debugging and testing. This instance is build in such a way that for all formulations it is easy to find a zero-cost solution and it can be verified also by human inspection.

Table 2 shows the main features of all instances together with some statistical indicators: courses (C), total lectures (L), rooms (R), periods per day (PpD), days (D), curricula (Cu), min and max lectures per day per curriculum (MML), average number of conflicts (Co), average teacher availability (TA), average number of lectures per curriculum per day (CL), average room occupation (RO).

More precisely:

- *conflict density* (Co) counts the pairs of lectures that cannot be scheduled at the same time (same course, same teacher, or same curriculum) divided by the total number of distinct pairs of lectures;
- *teacher's availability* (TA) is the average teacher's availability per lecture calculated on the total number of distinct pairs lectures/periods;
- *room's suitability* is the average room's availability calculated on the total number of distinct pairs lectures/rooms;
- *lectures per day per curriculum* (CL) is the average number of daily lectures;
- *room occupation* (RO) is the total number of lectures divided by the total number of distinct pairs room/period.

Notice that Co and TA are computed for each single lecture rather than at course level, in order to consider the fact that courses have different number of lectures.

Instance	C	L	R	PpD	D	Cu	MML	Co	TA	CL	RO
comp01	30	160	6	6	5	14	2-5	13.2	93.1	3.24	88.9
comp02	82	283	16	5	5	70	2-4	7.97	76.9	2.62	70.8
comp03	72	251	16	5	5	68	2-4	8.17	78.4	2.36	62.8
comp04	79	286	18	5	5	57	2-4	5.42	81.9	2.05	63.6
comp05	54	152	9	6	6	139	2-4	21.7	59.6	1.8	46.9
comp06	108	361	18	5	5	70	2-4	5.24	78.3	2.42	80.2
comp07	131	434	20	5	5	77	2-4	4.48	80.8	2.51	86.8
comp08	86	324	18	5	5	61	2-4	4.52	81.7	2	72
comp09	76	279	18	5	5	75	2-4	6.64	81	2.11	62
comp10	115	370	18	5	5	67	2-4	5.3	77.4	2.54	82.2
comp11	30	162	5	9	5	13	2-6	13.8	94.2	3.94	72
comp12	88	218	11	6	6	150	2-4	13.9	57	1.74	55.1
comp13	82	308	19	5	5	66	2-3	5.16	79.6	2.01	64.8
comp14	85	275	17	5	5	60	2-4	6.87	75	2.34	64.7
comp15	72	251	16	5	5	68	2-4	8.17	78.4	2.36	62.8
comp16	108	366	20	5	5	71	2-4	5.12	81.5	2.39	73.2
comp17	99	339	17	5	5	70	2-4	5.49	79.2	2.33	79.8
comp18	47	138	9	6	6	52	2-3	13.3	64.6	1.53	42.6
comp19	74	277	16	5	5	66	2-4	7.45	76.4	2.42	69.2
comp20	121	390	19	5	5	78	2-4	5.06	78.7	2.5	82.1
comp21	94	327	18	5	5	78	2-4	6.09	82.4	2.25	72.7
test1	46	207	12	4	5	26	2-4	5.25	97.6	1.97	86.2
test2	52	223	12	4	5	30	2-4	5.57	86.1	2.11	92.9
test3	56	252	13	4	5	55	2-4	5.89	78.1	2	96.9
test4	55	250	10	5	5	55	2-4	5.98	76.8	2	100
DDS1	201	900	21	15	5	99	3-7	4.58	21.3	5.18	57.1
DDS2	82	146	11	11	6	11	3-6	23.2	34.8	4.06	20.1
DDS3	50	206	8	11	5	9	3-6	12.4	58.8	4.76	46.8
DDS4	217	972	31	10	5	105	3-6	2.85	91.4	3.78	62.7
DDS5	109	560	18	12	6	44	3-6	2.19	66	1.89	43.2
DDS6	107	324	17	5	5	62	2-4	5.79	77.8	2.38	76.2
DDS7	49	254	9	10	6	37	3-6	14	89	3.01	47
toy	4	16	3	4	5	2	2-3	75	90	2.1	26.7

Table 2 Description of the instances

3.2 Data Formats

The `.ctt` data format used for ITC2007 could not be used as is for formulations UD3, UD4, and UD5, because it needs to be extended for adding the extra data necessary for the new features and cost components. We believe that it would be too complicated to maintain separated data formats for each formulation, therefore we decide to create an extended format that accommodates all the features. Researchers that are interested in one specific formulation can simply ignore the unnecessary additional information from the input files.

The outcome is the `.ectt` file format (`e` for extended) which largely resembles the `.ctt` one, but with extra data added in various points: header, courses, rooms, and constraints sections. For brevity, it is not presented here but it is fully described in the web site.[1]

In addition, we propose an XML format, along with its DTD (Document Type Definition), that contains the same information. We post on the website all instances

[1] It also corrects an incongruity in the use of the separators in the `.ctt` format: in `.ectt` files data fields are always separated by one single space character.

also in this alternative format, for the convenience of researchers that are accustomed to use this (very flexible) technology.

4 Problem Management System

The CB-CTT web site (`http://tabu.diegm.uniud.it/ctt`), is what we call a *problem management system* (PMS); that is, a web application which provides all the tools for the complete benchmarking of the underlying optimization problem.

The PMS aim is to join the common web sites features with more advanced ones, such as solution validation, instance generator, and an online solver, which could help the community of researchers. In [27] we advocated for the necessity of developing a system which could manage an optimization problem, because we all know the importance of a benchmarking activity in the research field, especially in the optimization area.

In the public section of the PMS the basic functions are available: description of the problem (data formats and instances), online validator, visualization of solutions and instance statistics. For registered users, the application becomes more interactive because users can not only add their solutions, lower bounds and new instances, but also generate new instances and ask for a solution.

We think that with the current version of the PMS we achieved our main goal, but we are aware that with the feedback and the interaction with the research community our application could still be improved. In the following sections we describe in more details the features available in our PMS.

4.1 Validate and Insert Solutions

In order to allow other researchers to check whether they correctly evaluate their solutions, we provide a solution validator along the lines of the one developed for ITC2007 track 3. This tool is a simple program that requires three command-line arguments: the formulation, the input file name, and the solution file name. The result of the validation is an output of all specific violations (one per line), and a summary report that shows the costs for each component plus the total one. The validator can be downloaded form the web site as C++ source code. We provide it open-source so that it can be inspected, improved, and extended by the community.

In addition, we also set up a web-based validator that allows the user to upload a solution file, select an existing instance and obtain the same report information (without having to compile and execute the validator). If a user is registered and logged in, upon validation, she/he can also insert her/his solutions in our repository, along with the associated score and time-stamp.

4.2 Visualize Instances and Solutions

All instances listed in 3.1 are downloadable, in each of the formats allowed. We organized all the instances in families, with one table per family, from which every user can also see all the statistics of every instance, including those described in table 2.

Contributed solutions are organized so as to be presented in various ways: by formulation, by author, or by score. Another feature is the visualization of the contributed solutions in the graphical timetable format, which gives a clearer feeling of how good or bad a solution really is. Users can also visualize the solution with the highlights of the violations and penalties (hard and soft).

4.3 Insert Lower Bounds

In order to allow for a better understanding of the structure of the CB-CTT instances, we provide a section of our PMS where users can upload the lower bounds of any instance and formulation.

We do not have any lower bound validator or mathematical software to prove the correctness of the lower bounds, so we accept any data uploaded (relying on users' trustfulness), as long as it is below the cost of all solutions. To this regard, along with the numeric values the user can also insert a log file, which could be used to reconstruct/prove the lower bound.

4.4 Insert New instances

We developed a checker for the instance files, so as to guide the process for other researchers to add their own instances to the benchmark set. This program reads an instance file (in `.ectt` or XML format) and checks if the data is coherent and correctly formatted, and it issues errors, warnings, and statistics. For example, if there is a constraint on a non-existing course, this arises an error; instead, if a course does not belong to at least one curriculum, this generates a warning.

The use of the input checker provides against the publication of non well-defined instances, that could create ambiguity on the computation of the cost. In any case, new instances would be published on the web only upon approval of the administrator. The PMS provides also the translation of the instances from `.ectt` or XML format to the other one (and to `.ctt`).

4.5 Random Instance Generator

The generator produces problem instances which could be used for experimentation purposes, with different characteristics for different values of the given parameters. The user specifies through a web form the values for the input features, and the PMS the requests the number of randomly generated instances. These random instances are not meant to be inserted in the repository, which is (and should remain) composed exclusively of real data.

The generator is available but a new version is under development, because we are planning trying to improve the control parameters of the program, avoiding the possibility to insert in the input form those specifications that can be fulfilled only by instances with errors or warnings. At present, the generator refrains from generating violations about repeated courses in a curriculum, repeated curricula, and total lectures of a curriculum. Instead, the most frequent warning is the presence of courses which not belong to any curricula (4.4), which could occur if the number of requested curricula is too small compared to the number of requested courses.

5 Related Work

The inspiration and the guidelines for this work come from Johnson's seminal paper [14] that emphasizes, among other features, the importance of the measurability of the experimental results in computer science.

The motivations come also from the observation of the evolution in the last decade of the literature on the "twin" problem, namely the examination timetabling problem (ETT). For ETT, Carter *et al* [7] proposed in 1996 a set of formulations which differ from each other based on some components of the objective function. Carter also made available a set of benchmark instances [6] extracted from real data, which represent a large variety of different situations. Many researches (see, e.g., [2,8,11]) have adopted one of Carter's formulations using his instances, and also have added new instances and new formulations[2]. For the most complex new formulations, additional data have been added by other researchers, mainly in an arbitrary but realistic way. Available formulations and instances, and the corresponding best results, up to 2003, have been published on the web [20] by Liam Merlot. More recently, Rong Qu has created a new web site [23] that allows the visitors to download an executable that validates ETT solutions (using a raw fixed-structure output format). Up to now, the executable validates only solutions for the basic version of ETT. Finally, a new version of the problem has been defined for ITC2007 (track 1) together with the online validator to test solutions [18].

Learning from the reported experience with ETT, we hope to give to CTT a similar, but hopefully more systematic, development that could converge rapidly to a mature state. Besides some earlier attempts to define a standard timetabling language [3,16, 22], a remarkable effort for CTT in this direction has been made by Müller and Murray: they published on the web [25] all their instances and the source code of the solution methods [21]. Their approach is however somewhat complementary to our, as they provide directly the general problem formulation with all its complex details. Our idea instead has been to start with simpler formulations and add complexity in a controlled way only when (and if) the time is mature for it. In our opinion, both can be useful for the evolution of the field.

6 Discussion, Conclusions, and Future Work

At present (June 30^{st}, 2009), the application has 24 users (excluding ourselves), 194 solutions, and 45 lower bounds inserted by other researchers. All the best known published solutions, up to our knowledge, have been inserted. We believe that the system is on its way to become the "official" reference for authors to see the current results and for referees to check the reliability of results claimed in the submitted papers.

We are now working on an online solving facility that could run available solvers on selected instances (either randomly generated or real world). This feature is currently under testing, and we are about to invite other researchers to contribute with their software. For the users that are willing to provide us their code, we plan for the future to add the possibility of validating also the running times of their solutions.

The other current work is on improving the random instance generator. The aim is to make it more configurable and more effective.

[2] Incidentally, as documented in detail in [24], at some time two slightly different versions of Carter's datasets were available on the web.

Regarding the problem, the full-fledged formulation used at the University of Udine, with respect to the five proposed here, still contains many extra features:

Preassignments: Some lecture can be preassigned to a specific room or a specific period.
Lunch break: There is a cost component dealing with the lunch break for students: at least one slot among those around the lunch time should be free.
Room too big: If a room is too big for a class, this is also penalized (this is not only for the unpleasant feeling that an empty room provokes, but also to save big rooms for activities which were unforeseen at the time the timetable is computed).
Flexibility: For most cost components, the hard and the soft version are both available to the user. For example, it is possible to set soft conflicts for courses and soft unavailability for teachers.
Configurability: Weight assigned to soft violations are not fixed. They have a complex penalty scheme, which depends also on the number of students in the curriculum.

Besides the above ones, many other cost components have emerged from other universities. The most common ones are the following:

Commuters: Some teachers must be assigned to consecutive days.
TeacherMinMaxLoad: Load limits similar to student's ones might be imposed also on teachers.
Day patterns: Courses might require specific patterns of lectures (e.g., lecture in the morning, labs in the afternoon).
Video-conference: A single lecture might require more than one room.
External rooms: Some lecture might be given in rooms that are outside the control of the system.
Simultaneity: It is possible that some courses are requested to be scheduled exactly at the same time (e.g., lab. sections of the same course).

All these features are among the candidates to be inserted in future formulations, depending also on the general interest they rise.

Acknowledgments

We thank Marco Chiarandini for his comments of an earlier draft of this paper that helped us to improve it. We are grateful to Barry McCollum and all the other members of the team of ITC2007 for their work for the organization of the competition. We also thank Martin Geiger, Gerald Lach, Zhipeng Lü, Jin-Kao Hao, and Tomáš Muller for using the system and giving us comments to improve it. Finally, we also thank Jakub Mareček and Harald Michalsen for fruitful discussions about the formulation and the solution of the problem.

References

1. Pasquale Avella and Igor Vasil'ev. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8:497–514, 2005.
2. E. Burke and J. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

3. E. Burke, P. Pepper, and J. Kingston. A standard data format for timetabling instances. In E. Burke and M. Carter, editors, *Proc. of the 2nd Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-97), selected papers*, volume 1408 of *Lecture Notes in Computer Science*, pages 213–222, Berlin-Heidelberg, 1997. Springer-Verlag.
4. Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. On a clique-based integer programming formulation of vertex colouring with applications in course timetabling. Technical Report NOTTCS-TR-2007-10, The University of Nottingham, Nottingham, 2007.
5. Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. Penalising patterns in timetables: Novel integer programming formulations. In Stefan Nickel and Jörg Kalcsics, editors, *Operations Research Proceedings 2007*, Operations Research Proceedings, Berlin, 2008. Springer.
6. M. W. Carter. Carter's test data. URL: `ftp://ftp.mie.utoronto.ca/pub/carter/testprob/`, 2005. Viewed: March 13, 2007, Updated: June 7, 2005.
7. M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 74:373–383, 1996.
8. S. Casey and J. Thompson. Grasping the examination scheduling problem. In Edmund Burke and Patrick De Causmaecker, editors, *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers*, volume 2740 of *Lecture Notes in Computer Science*, pages 232–244, Berlin-Heidelberg, 2003. Springer-Verlag.
9. S. Daskalaki, T. Birbas, and E. Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153:117–135, 2004.
10. Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0/1, School of Electronics, Electrical Engineering and Computer Science, Queens University, Belfast (UK), August 2007. ITC-2007 site: `http://www.cs.qub.ac.uk/itc2007/`.
11. Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetabling. In E. Burke and W. Erben, editors, *Proc. of the 3rd Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2000), selected papers*, volume 2079 of *Lecture Notes in Computer Science*, pages 104–117. Springer-Verlag, Berlin-Heidelberg, 2001.
12. Luca Di Gaspero and Andrea Schaerf. Multi-neighbourhood local search with application to course timetabling. In Edmund Burke and Patrick De Causmaecker, editors, *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers*, volume 2740 of *Lecture Notes in Computer Science*, pages 262–275, Berlin-Heidelberg, 2003. Springer-Verlag.
13. Luca Di Gaspero and Andrea Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modeling and Algorithms*, 5(1):65–89, 2006.
14. D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society, 2002. Available from `http://www.research.att.com/~dsj/papers.html`.
15. Jari Kyngäs Kimmo Nurmi. A conversion scheme for turning curriculum-based timetabling problem into school timetabling problem. In E. Burke and M. Gendreau, editors, *Proc. of the 7th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2008)*, 2008.
16. Jeffrey H. Kingston. Modelling timetabling problems with STTL. In E. Burke and W. Erben, editors, *Proc. of the 3rd Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2000), selected papers*, volume 2079 of *Lecture Notes in Computer Science*, pages 309–321. Springer-Verlag, Berlin-Heidelberg, 2001.
17. Barry McCollum. A perspective on bridging the gap in university timetabling. In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006), selected papers*, volume 3867 of *Lecture Notes in Computer Science*, pages 3–23, Berlin-Heidelberg, 2007. Springer-Verlag.
18. Barry McCollum, Paul McMullan, Edmund K. Burke, Andrew J. Parkes, and Rong Qu. The second international timetabling competition: Examination timetabling track. Technical Report QUB/IEEE/Tech/ITC2007/Exam/v4.0/17, Queens University, Belfast (UK), September 2007.

19. Barry McCollum, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu, , and Edmund K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 2009. In press.
20. Liam Merlot. Public exam timetabling data sets. URL: `http://www.or.ms.unimelb.edu.au/timetabling`, 2005. Viewed: March 13, 2007, Updated: October 13, 2003.
21. Keith S. Murray, Tomás Müller, and Hana Rudová. Modeling and solution of a complex university course timetabling problem. In *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006), selected papers*, pages 189–209, 2007.
22. E. Özcan. Towards an XML-based standard for timetabling problems: TTML. In G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, editors, *Proc. of the 1st Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA-03), selected papers*, pages 163–185. Springer, 2005.
23. R. Qu. The exam timetabling site. URL: `http://www.cs.nott.ac.uk/~rxq/ETTP.htm`, 2006. Viewed: March 13, 2007, Updated: July 8, 2006.
24. R. Qu, E. Burke, B. McCollum, L. Merlot, and S.Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.
25. Tomáš Müller and Keith Murray. University course timetabling & student scheduling. URL: `http://www.unitime.org`, 2008. Viewed: January 24, 2008, Updated: November 28, 2007.
26. Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.
27. Andrea Schaerf and Luca Di Gaspero. Measurability and reproducibility in timetabling research: Discussion and proposals. In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006), selected papers*, volume 3867 of *Lecture Notes in Computer Science*, pages 40–49, Berlin-Heidelberg, 2007. Springer-Verlag.