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

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

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 site that maintains all the necessary *infrastructures* for benchmarking: i.e., validators, data formats, instances, reference scores, lower bounds, and solutions. 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 [22].

CTT is a practically-important and widely-studied problem; nevertheless, if we go through the research literature on it, we see that many papers create a new formulation and 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 an important 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.

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

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

The International Timetabling Competitions, ITC-1 in 2002-03 and ITC-2 in 2007-08, have been organized specifically with the aim of creating the common ground for comparison. 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 published by the university, and not on the basis of enrollment data. In this work, we focus on this 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 the possible formulations, has been used as one of the tracks of ITC-2, 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.

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. Thus also trying to help in bridging the gap between theory and practice [14].

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.

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, and solutions. In fact, we believe that all the above mentioned tools are mandatory in order to foster the necessary sharing and cross-fertilization [23]. 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[1], 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 site (`http://tabu.diegm.uniud.it/ctt`) from which all datasets can be downloaded and solutions can be validated. All updates and news about the problem will be posted there.

[1] The solver that we use for our solutions is a dynamic tabu search algorithm.

2 Problem Definition

The basic features of CB-CTT are presented in the ITC-2 web site and in the corresponding technical report [8]. However, the model has to be extended to consider the addition components. In order to make this paper self-contained, we present here the full extended 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 in 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 be not 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.

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 component 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 or equal than 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 (CurriculumCompactness 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 (CurriculumCompactness 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. Two lectures are grouped if they are both adjacent and in the same room. *Each non-grouped lecture counts as 1 violation.*

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 [10,11] and it has been used also by Burke *et al* [3,4] with an IP-based approach. The second one has been specifically designed for ITC-2.

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

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

(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 Benchmarking

We present in this section all the information that are needed to benchmark the algorithms, and thus to share the problem with other researchers.

3.1 Instances

All instances are downloadable from the problem web site. The set of instances includes the 4 instances used in [10], 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 among 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 this and similar cases, some information has been added in an arbitrary way. Nevertheless, most of the information is composed of real data.

We have added 7 new instances, also coming from real world settings, which come mainly from different institutions, and thus represent new cases to be dealt with. We name these instances as `DDS1`, ..., `DDS7`.

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

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 quantities. In details, it shows: 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).

The feature Co (conflicts) counts the pairs of lectures that cannot be scheduled at the same time (same course, same teacher, and same curriculum) divided by the total number of distinct pairs of lectures.

Notice that Co and TA are computed for single lecture rather than at course level, so as to take into account the fact that courses have different number of lectures.

3.2 Data formats

The `.ctt` data format used for ITC-2 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.[2]

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

Finally, we provide the translators from one format to the other.

3.3 Validators

In order to allow other researchers to test whether they correctly evaluate their solutions, we provide a solution validator along the lines of the one for ITC-2 track 3. This tool is a simple program that requires three command-line arguments: the formulation, the input file, and the solution file. 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).

Finally, we also developed a validator for the instance files, so as to guide the process for other researchers to add their own instances to the benchmark set. This input validator 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 to a non-existing course, this creates an error, wherever, if a course does not belong to at least one curriculum, this generates a warning.

The use of the input validator should avoid 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.

[2] 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 blank.

3.4 Reference Results

Our solutions are obtained by a dynamic tabu search algorithm. In details, the solver uses a short-term tabu prohibition with variable-size tabu length, but also includes the dynamic modification of the weights of the cost components (including hard ones) based on the number of violations. The neighborhood relation used is the simple one that moves one lecture to a different period and/or a different room. The main parameters, namely the tabu length range and the rate of modification of the weights have been selected based on statistical tests.

We present here some of our (preliminary) results, with the aim of providing some reference values for everybody interested in the problem. All corresponding solution files are available from the web site.

Table 3 shows the results obtained using the competition timeout (360s on our PC) and taking the best out of 40 repetitions. For all instances and all formulations we obtained a feasible solution, and thus Table 3 reports only the total (soft) penalty.

We show only the best solution rather than the full distribution, because the scope is to have this reference results instead of describing the results in a statistically principled way.

In general, publishing on the web the best scores obviously gives only a partial view of the results of a research, which indeed involves also distributions, running times, and many other features. However, in our opinion, this information pertains to the papers written by the authors, and it is outside the scope of the website, that has the objective of collecting, validating, and classifying the results.

4 Related Work

The inspiration and the guidelines for this work come from Johnson's seminal paper [12] 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* [6] 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 [5] extracted from real data, which represent a large variety of different situations. Many researches (see, e.g., [1, 7, 9]) have adopted one of Carter's formulations using his instances, and also have added new instances and new formulations[3]. 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 [16] by Liam Merlot. More recently, Rong Qu has created a new web site [19] 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 ITC-2 (track 1) together with the online validator to test solutions [15].

[3] Incidentally, as documented in detail in [20], at some time two slightly different versions of Carter's datasets were available on the web.

Form./ Instance	comp01	comp02	comp03	comp04	comp05	comp06	comp07
UD1	4	35	52	21	244	27	13
UD2	5	75	93	45	326	62	38
UD3	8	34	45	2	434	14	12
UD4	6	49	371	20	408	28	35
UD5	11	270	206	92	1269	202	213

Form./ Instance	comp08	comp09	comp10	comp11	comp12	comp13	comp14
UD1	24	61	10	0	268	38	30
UD2	50	119	27	0	358	77	59
UD3	8	18	2	0	140	32	2
UD4	20	47	18	0	147	51	21
UD5	102	208	190	0	665	195	138

Form./ Instance	comp15	comp16	comp17	comp18	comp19	comp20	comp21
UD1	46	28	44	41	36	25	69
UD2	87	47	86	71	74	54	117
UD3	30	12	14	10	34	24	30
UD4	42	23	38	32	39	39	60
UD5	259	182	216	150	224	309	247

Form./ Instance	test1	test2	test3	test4
UD1	214	8	36	43
UD2	234	17	86	132
UD3	200	0	18	24
UD4	213	4	22	37
UD5	245	24	108	173

Form./ Instance	DDS1	DDS2	DDS3	DDS4	DDS5	DDS6	DDS7
UD1	238	0	0	233	0	5	0
UD2	1024	0	0	261	0	11	0
UD3	5944	128	22	3988	56	2	40
UD4	2593	78	11	6735	165	10	29
UD5	9677	93	22	13064	128	185	97

Table 3 Reference results

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 [2,13,18], a remarkable effort for CTT in this direction has been made by Müller and Murray: they published on the web [21] all their instances and the source code of the solution methods [17]. 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.

5 Discussion, Conclusions, and Future Work

In [23] we advocated for the necessity of developing a web application, the so-called PMS (*problem management system*), rather than a simple web site for managing an optimization problem. Our current web site is the first step toward the development

of the complete PMS, which is currently under development. At present, the most important feature that is still missing is the possibility for other researchers to add (validated) solutions and new instances in an automatic way. Another remarkable feature that is under development is the visualization of the solution with the highlight of the violations and penalties.

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

- 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.
- 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.
- 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 unforeseen activities).
- Weight assigned to soft violations are not fixed. The have a complex penalty scheme, which depends also on the number of students in the curriculum.

These features are among the first 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 the other members of the team of ITC-2 for their work for the organization of the competition.

Finally, we also thank Jakub Mareček and Harald Michalsen for fruitful discussions about the formulation of Course Timetabling.

References

1. E. Burke and J. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.

7. 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.
8. 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/`.
9. 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.
10. 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.
11. 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. DOI: 10.1007/s10852-005-9032-z.
12. 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`.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. R. Qu, E. Burke, B. McCollum, L. Merlot, and S.Y. Lee. The state of the art of examination timetabling. Technical Report NOTTCS-TR-2006-4, School of CSiT, University of Nottingham, UK, 2006.
21. Tomáš Müller and Keith Murray. University course timetabling & student scheduling. URL: `http://www.unitime.org`, 2008. Viewed: January 24, 2008, Updated: November 28, 2007.
22. Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.
23. 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.