TOP (2015) 23:313–349
DOI 10.1007/s11750-015-0366-z

INVITED PAPER

An overview of curriculum-based course timetabling

Andrea Bettinelli · Valentina Cacchiani · Roberto Roberti · Paolo Toth

Published online: 19 March 2015

Abstract In 2007, the Second International Timetabling Competition (ITC-2007) has been organized and a formal definition of the *Curriculum-Based Course Timetabling* (CB-CTT) problem has been given, by taking into account several real-world constraints and objectives while keeping the problem general. CB-CTT consists of finding the best weekly assignment of university course lectures to rooms and time periods. A feasible schedule must satisfy a set of *hard* constraints and must also take into account a set of *soft* constraints, whose violation produces penalty terms to be minimized in the objective function. From ITC-2007, many researchers have developed advanced models and methods to solve CB-CTT. This survey is devoted to review the main works on the topic, with focus on mathematical models, lower bounds, and exact and heuristic algorithms. Besides giving an overview of these approaches, we highlight interesting extensions that could make the study of CB-CTT even more challenging and closer to reality.

This invited paper is discussed in the comments available at doi:10.1007/s11750-015-0362-3, doi:10.1007/s11750-015-0363-2, doi:10.1007/s11750-015-0364-1, doi:10.1007/s11750-015-0365-0.

A. Bettinelli · V. Cacchiani · P. Toth (✉)
DEI, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
e-mail: paolo.toth@unibo.it

A. Bettinelli
e-mail: andrea.bettinelli@unibo.it

V. Cacchiani
e-mail: valentina.cacchiani@unibo.it

R. Roberti
Department of Transport, Technical University of Denmark,
2800 Kgs Lyngby, Denmark
e-mail: rorob@transport.dtu.dk

Keywords University timetabling · Curriculum-based course timetabling · Survey · Models · Exact algorithms · Heuristic algorithms

Mathematics Subject Classification 90-02 · 90C10 · 90C57 · 90C59

1 Introduction

Timetabling is an important and active area of research with applications in several fields such as university, transportation, high school, hospital, sport, employees, etc. It describes a variety of usually NP-hard optimization problems with a considerable practical impact. Many advanced techniques and real-world applications have been published in the proceedings of the biannual International Conference on the Practice and Theory of Automated Timetabling (PATAT http://www.patatconference.org/).

Roughly speaking "timetabling" can be defined as the problem of assigning a set of activities to resources (Wren 1996; Schaerf 1999; Carter 2013), such as time slots, personnel and locations, under a complex set of constraints, usually called *hard constraints*, which vary according to the specific problem. The goal is to find a feasible assignment of all the activities to the resources, while minimizing the weighted sum, according to given weights, of the penalties corresponding to the violations of the *soft constraints*.

In the recent years, there has been an increasing interest in *Automated University Timetabling*, with many surveys published on the topic: Burke et al. (1997), Schaerf (1999), Burke and Petrovic (2002), Petrovic and Burke (2004), McCollum and Ireland (2006), McCollum (2007), Lewis (2008), Kingston (2013), MirHassani and Habibi (2013), Kristiansen and Stidsen (2013), Babaei et al. (2014), Pillay (2014).

The interest in this research area is also due to the International Timetabling Competitions organized on this topic. The first International Timetabling Competition (ITC-2002) was organized in 2002 by Paechter and Gambardella of the European Metaheuristics Network and was sponsored by PATAT. The aim of this competition was to attract researchers and to establish a precise problem definition and a set of randomly generated benchmark instances for the university course timetabling problem, to allow the comparison of the performances of the various proposed methods. The success of this first competition is evident from the number of published papers on the topic (see e.g. Abdullah et al. 2007; Chiarandini et al. 2006; Di Gaspero and Schaerf 2006; Kostuch 2005; Landa-Silva and Obit 2008; Lewis et al. 2007).

Following the success of ITC-2002, the second ITC (ITC-2007) was organized in 2007 (Di Gaspero et al. 2007; Bonutti et al. 2012; McCollum et al. 2010). As explained in Di Gaspero et al. (2007), a main innovation of ITC-2007 was to distinguish university timetabling into three categories, and define for each of them one track for the competition. These tracks correspond to *Examination Timetabling*, *Post-Enrolment-based Course Timetabling* (PE-CTT) and *Curriculum-Based Course Timetabling* (CB-CTT). The first problem deals with the definition of examination timetables, i.e., it requires to assign exams to time slots within a given examination session while satisfying a set of hard constraints (e.g. students cannot give more than one exam at the same time, the number of students giving the exam cannot exceed the capacity of the room, etc.).

The other two problems fall in the category of *Course Timetabling*: in PE-CTT students enroll for courses before the determination of the timetable, while in CB-CTT a curriculum predefines the sets of lectures that students are supposed to follow. This is not the only difference between the two problems: an important one is that in PE-CTT each course is a single event, while in CB-CTT a course consists of a series of lectures. As a consequence, constraints and objectives are also different. In addition, CB-CTT does not involve *student sectioning*, i.e., assigning students to individual sections of a course (see e.g. Müller and Murray 2010). This is an important step in course timetabling that has been neglected in CB-CTT to keep the problem not too complex. Both variants are important in real-life applications, and one or the other is used depending on the organization of the considered university. We refer the reader to McCollum et al. (2010) and Di Gaspero et al. (2007) for further details on these tracks and on the ITC-2007 rules.

Due to the very large amount of research on the three tracks and on university timetabling for specific real-life case studies, we focus, in this survey, on CB-CTT as defined in McCollum et al. (2010) for the third track of ITC-2007, and highlight the most successful models and approaches proposed to solve it. With respect to previous surveys on Automated University Timetabling, the main difference is that we focus on the well-defined CB-CTT according to the definition given in Sect. 2. This definition has been proposed in ITC-2007 and since then many works on successful methods to solve CB-CTT have been published, which make CB-CTT a well-established topic of research.

The paper is organized as follows. In Sect. 2, we report the definition of CB-CTT. In Sect. 3, we present mathematical models, lower bounds and exact algorithms, while Sect. 4 is devoted to the description of heuristic algorithms. Finally, we describe extensions of the problem in Sect. 5 and conclude with perspective on possible future research directions in Sect. 6.

2 Problem description

This section is devoted to the description of CB-CTT, as formalized for ITC-2007. We first present the hard constraints and the soft constraints. Then, we describe the benchmark instances introduced in Bonutti et al. (2012) (see Sect. 2.1) and conclude this section (see Sect. 2.2) with the notation used in the remainder of this paper.

In CB-CTT we are given sets of:

- *time periods*: the time horizon (typically 5 or 6 teaching days) is divided in *days* and each day is divided in a fixed number of *time slots*; a time period is a pair (day, time slot);
- *courses* and *teachers*: each course consists of a given number of *lectures*, is taught by a teacher and is attended by a given number of *students*. It can belong to some university curricula and its lectures should be spread across a *minimum number of days*. In addition, for each course, a set of *unavailable time periods*, i.e., time periods in which the teacher of the course is not available, is given;
- *curricula*: a curriculum corresponds to a set of courses that must be taken by some students;

– *rooms*: each room is characterized by a *capacity*, which represents the number of seats in the room.

CB-CTT consists of finding the best assignment of course lectures to rooms and time periods, while satisfying the following *hard constraints*:

– *lectures*: all the lectures of a course must be scheduled and they must be assigned to distinct time periods;
– *room occupancy*: each room can host at most one lecture per time period;
– *conflicts*: lectures of courses belonging to the same curriculum or taught by the same teacher cannot be scheduled in the same time period;
– *availabilities*: unavailable time periods for the teacher of the course cannot be used for scheduling a lecture of that course.

The goal is the minimization of the weighted sum (according to given weights) of multiple objective functions, representing the costs for the violation of the following *soft constraints*:

– *minimum number of working days*: for each course, a penalty is given for each day below the minimum number of working days;
– *curriculum compactness*: it is preferable that the lectures of a curriculum are consecutive, without any empty time period in between; thus, a penalty is given for each *isolated lecture*, i.e., a lecture not adjacent to any other lecture of the same curriculum in the same day;
– *room capacity*: a penalty is given for each student that cannot have a seat in the room assigned to the course lecture;
– *room stability*: it is preferable that a course is always taught in the same room, thus, a penalty is given for each additional room used for a course.

CB-CTT as defined above corresponds to the university course timetabling problem arising in many Italian universities and also in international ones. Even if additional constraints can appear in some cases, it has been simplified as described above for ITC-2007 in order to maintain a certain level of generality (see Di Gaspero et al. 2007). In Bonutti et al. (2012), a set of variants, called `UD1`, `UD2`, `UD3`, `UD4`, and `UD5`, of CB-CTT have been proposed. The problem defined above corresponds to `UD2`, which is the most studied variant. We mention that `UD1` corresponds to `UD2` without the soft constraint of room stability and with a different weight for the penalization of the curriculum compactness. The other three variants have been introduced with the aim of including many real-world soft constraints. These and other possible extensions of CB-CTT are discussed in Sect. 5. In this survey we focus on `UD2`.

The CB-CTT problem is NP-hard since it has as a core problem a *graph coloring* problem (see Burke et al. 2010b), which is a well-known NP-hard problem. Let us consider a graph having one vertex for each lecture and one edge for each pair of lectures that cannot be scheduled simultaneously (i.e., either they belong to the same course or to courses of the same curriculum or they must be taught by the same teacher). Let us consider one color for each time period. The core problem of CB-CTT is to assign one color to each vertex, so that adjacent vertices are assigned different colors. In addition, a constraint is imposed on the maximum number of times each color can be used, which is set equal to the number of available rooms. This leads to the so-called *bounded coloring problem* (see Hansen et al. 1993).

2.1 Benchmark instances

A fundamental outcome of ITC-2007 was the definition of a wide set of real-world instances to allow the comparison of different solution methods. These instances are publicly available on the website http://satt.diegm.uniud.it/ctt, maintained by the organizers of ITC-2007. A first set of instances, defined in Di Gaspero and Schaerf (2003), consists of four instances denoted by `test*`. The main set of instances, defined in Di Gaspero et al. (2007) for ITC-2007, consists of 21 instances denoted by `comp*`. A third set of instances was defined in Bonutti et al. (2012) and consists of seven instances denoted by `DDS*`. Very recently, additional instances have been proposed, which are identified by `erlangen*`,[1] `Udine*` and `EA*`.[2]

On the same website, the best-known heuristic solution values and lower bound values are also reported and continuously updated. In addition, the website provides an online solution validator, allows the visualization of the solutions and gives information on the instance statistics (see also Bonutti et al. 2012 for a detailed analysis of the instance characteristics). To produce comparable results, a benchmarking program is provided on the website, which can be used to set the time limit of the algorithm execution. In particular, this program was used to allocate a time limit to each of the competitor algorithms of the ITC-2007 competition: each competitor had to run the program on his machine and, when the program halted, the corresponding execution time is used to define the time limit, called *CPU time unit*, for the considered machine. In addition, the results for the competition were obtained by performing a random sample of 10 runs of each algorithm on all the available instances.

We wish to stress that although the availability of the benchmark instances and of the benchmarking program has certainly provided a way to come very close to fair computational comparisons of different methods, the settings, machines and computing time limits vary significantly between different works. For this reason, when reporting tables that include computational results taken from different papers, we always show the corresponding computing time and used computer, as well as the number of runs and (when used) the general-purpose Mixed Integer Linear Programming (MILP) solver (see Sects. 3.8 and 4.3).

Two instance generators have also been developed: the first one is by Burke et al. (2010b), based on the benchmark instances proposed in Di Gaspero and Schaerf (2003), while the second one is an improvement by Lopes and Smith-Miles (2010, 2013), who base their work on a deeper insight on the features of the instances. In the latter paper, the authors propose a methodology for tuning instance generators of problems that contain knapsack or graph coloring problems as a core problem. Their goal is to produce instances that are similar to the real-life ones and also able of highlighting different behaviors of the solvers. The proposed methodology is applied to CB-CTT. The generated instances can be used to tune an algorithm, to avoid overtuning phenomena on the set of benchmark instances. In Bellio et al. (2014), for example, a simulated annealing algorithm is tuned on the generated instances and

[1] Contributed by Moritz Mühlenthaler.

[2] http://www.easystaff.it.

turns out to obtain comparable or even better results than algorithms tuned on the ITC-2007 instances.

2.2 Notation

In this section, we report the notation used in the remainder of the paper.

- CB-CTT: problem
- $\mathbb{B}$: binary range
- $\mathbb{Z}$: integer range
- $\mathscr{C}$: set of courses
- $\mathscr{Q}$: set of curricula
- $\mathscr{D}$: set of days
- $\mathscr{H}$: set of time periods
- $\mathscr{R}$: set of rooms
- $\mathscr{T}$: set of teachers
- $\mathscr{C}_q$: set of courses belonging to curriculum $q \in \mathscr{Q}$
- $\mathscr{C}_t$: set of courses hold by teacher $t \in \mathscr{T}$
- $\mathscr{H}_d$: set of time periods of day $d \in \mathscr{D}$
- $\mathscr{Q}_c$: set of curricula involving course $c \in \mathscr{C}$
- d_h: day of time period $h \in \mathscr{H}$
- lct_c: number of lectures of course $c \in \mathscr{C}$
- mwd_c: minimum working days of course $c \in \mathscr{C}$
- st_c: number of students attending course $c \in \mathscr{C}$
- cap_r: capacity of room $r \in \mathscr{R}$
- W^{CC}: penalty for curriculum compactness
- W^{RC}: penalty for room capacity
- W^{RS}: penalty for room stability
- W^{WD}: penalty for minimum working days

3 Mathematical models, lower bounds and exact algorithms

The aim of this section is to illustrate the main mathematical models and algorithms that can be found in the literature to solve CB-CTT to optimality and to compute lower bounds. Most of the mathematical models presented in this section are used either to derive lower bounds to CB-CTT or to compute upper bounds within a heuristic framework or to both aims. Only a few exact algorithms have been proposed in the literature. Since most of the works that contribute on mathematical models also provide methods for computing lower bounds and rarely give a framework for an exact solution of CB-CTT, we describe them all together in this section, specifying whether the method is exact or employed for lower/upper bound computation. Section 3.1 describes the compact formulation of Burke et al. (2010a) and some ideas to derive various lower bounds. In Sect. 3.2, we sketch the main ideas that characterize the branch-and-cut algorithm of Burke et al. (2012). The two-stage approach of Lach and Lübbecke (2012) is illustrated in Sect. 3.3. Section 3.4 describes the *divide-and-conquer* method

of Hao and Benlic (2011). A column generation algorithm developed by Cacchiani et al. (2013) is illustrated in Sect. 3.5. A different approach based on *Satisfiability* (SAT) solvers was proposed by Asín Achá and Nieuwenhuis (2014) and is presented in Sect. 3.6. In this section, we focus on the CB-CTT as defined in Sect. 2. In the literature, other university timetabling problems can be found; recently published lower bounds and exact methods for such problems are described in Carter (2001), Daskalaki et al. (2004), Avella and Vasil'ev (2005), Daskalaki and Birbas (2005), Qualizza and Serafini (2005), MirHassani (2006), Al-Yakoob and Sherali (2007), Schimmelpfeng and Helber (2007), Lach and Lübbecke (2008), Van Den Broek et al. (2009), Miranda (2010) and Phillips et al. (2015).

3.1 The compact formulation of Burke et al. (2010a) and various possible lower bounds

Burke et al. (2008, 2010a) introduced a compact *Integer Linear Programming* (ILP) formulation, hereafter called *Monolithic*, that can be solved with a generic ILP solver and provides an optimal solution of the CB-CTT, assuming that enough time is given to the ILP solver for its resolution. Five sets of variables are used:

- $x_{chr} \in \mathbb{B}$ equals 1 if course $c \in \mathscr{C}$ is assigned to room $r \in \mathscr{R}$ at period $h \in \mathscr{H}$ (we assume $x_{chr} = 0$ whenever course c cannot take place at time period h and/or cannot take place in room r);
- $v_{cd} \in \mathbb{B}$ equals 1 if at least a lecture of course $c \in \mathscr{C}$ is scheduled on day $d \in \mathscr{D}$;
- $u_c \in \mathbb{Z}$ represents the number of days course $c \in \mathscr{C}$ is short of mwd_c;
- $z_{hq} \in \mathbb{B}$ equals 1 if there is an isolated lecture of curriculum $q \in \mathscr{Q}$ at time period $h \in \mathscr{H}$;
- $y_{cr} \in \mathbb{B}$ equals 1 if at least a lecture of course $c \in \mathscr{C}$ takes place in room $r \in \mathscr{R}$.

The Monolithic formulation reads as follows:

$$\min \quad W^{RC} \sum_{r\in\mathscr{R}} \sum_{\substack{c\in\mathscr{C}:\\ st_c>\text{capr}}} \sum_{h\in\mathscr{H}} (st_c - cap_r)x_{chr} + W^{WD} \sum_{c\in\mathscr{C}} u_c$$
$$+ W^{CC} \sum_{h\in\mathscr{H}} \sum_{q\in\mathscr{Q}} z_{hq} + W^{RS} \sum_{c\in\mathscr{C}} \left(\sum_{r\in\mathscr{R}} y_{cr} - 1 \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{h\in\mathscr{H}} \sum_{r\in\mathscr{R}} x_{chr} = lct_c \qquad c \in \mathscr{C} \tag{2}$$

$$\sum_{c\in\mathscr{C}} x_{chr} \leq 1 \qquad h \in \mathscr{H},\ r \in \mathscr{R} \tag{3}$$

$$\sum_{r\in\mathscr{R}} x_{chr} \leq 1 \qquad c \in \mathscr{C},\ h \in \mathscr{H} \tag{4}$$

$$\sum_{r\in\mathscr{R}} \sum_{c\in\mathscr{C}_t} x_{chr} \leq 1 \qquad h \in \mathscr{H},\ t \in \mathscr{T} \tag{5}$$

$$\sum_{r\in\mathscr{R}} \sum_{c\in\mathscr{C}_q} x_{chr} \leq 1 \qquad h \in \mathscr{H},\ q \in \mathscr{Q} \tag{6}$$

$$\sum_{r\in\mathscr{R}} x_{chr} \le v_{cd_h} \qquad c \in \mathscr{C},\ h \in \mathscr{H} \tag{7}$$

$$\sum_{r\in\mathscr{R}} \sum_{h\in\mathscr{H}_d} x_{chr} \ge v_{cd} \qquad c \in \mathscr{C},\ d \in \mathscr{D} \tag{8}$$

$$\sum_{d\in\mathscr{D}} v_{cd} \ge mwd_c - u_c \qquad c \in \mathscr{C} \tag{9}$$

$$\sum_{c\in\mathscr{C}_q} \sum_{r\in\mathscr{R}} (x_{chr} - x_{c,h-1,r} - x_{c,h+1,r}) \le z_{hq} \qquad q \in \mathscr{Q},\ h \in \mathscr{H} \tag{10}$$

$$x_{chr} \le y_{cr} \qquad c \in \mathscr{C},\ h \in \mathscr{H},\ r \in \mathscr{R} \tag{11}$$

$$\sum_{h\in\mathscr{H}} x_{chr} \ge y_{cr} \qquad c \in \mathscr{C},\ r \in \mathscr{R} \tag{12}$$

$$x_{chr} \in \mathbb{B} \qquad c \in \mathscr{C},\ h \in \mathscr{H},\ r \in \mathscr{R} \tag{13}$$

$$v_{cd} \in \mathbb{B} \qquad c \in \mathscr{C},\ d \in \mathscr{D} \tag{14}$$

$$u_c \in \mathbb{Z} \qquad c \in \mathscr{C} \tag{15}$$

$$z_{hq} \in \mathbb{B} \qquad h \in \mathscr{H},\ q \in \mathscr{Q} \tag{16}$$

$$y_{cr} \in \mathbb{B} \qquad c \in \mathscr{C},\ r \in \mathscr{R}. \tag{17}$$

The objective function (1) asks for minimizing the violation of soft constraints. Constraints (2)–(6) model hard constraints by stating that each lecture of course c must be scheduled, no more than a course can take place in the same room at the same time period, a course cannot be assigned to more than a room at a given time period, and courses of a teacher or belonging to the same curriculum cannot be scheduled simultaneously. Constraints (7)–(8) impose on variable v_{cd} to be equal to 1 if and only if there is at least a lecture of course $c \in \mathscr{C}$ held on day $d \in \mathscr{D}$. Constraints (9) link variables $\mathbf{v}$ and $\mathbf{u}$. Constraints (10) model curriculum compactness constraints; notice that, whenever h corresponds the first (last, resp.) time period of day d, variables $x_{c,h-1,r}$ ($x_{c,h+1,r}$, resp.) do not exist and are assumed to be 0. Room stability is modeled through constraints (11)–(12). Integrality on the variables is imposed by constraints (13)–(17).

As illustrated by Burke et al. (2010b), the Monolithic formulation (1)–(17) can be interpreted as a *supernodal* formulation derived from the standard formulation of graph coloring problems, where there is a binary variable x_{ehr} for each lecture (an event) instead of each course c. The resulting formulation has many more variables and much worse computational performance than the Monolithic formulation (the reader is referred to Burke et al. 2010b for detailed computational results).

The Monolithic formulation can be solved with any generic ILP solver to obtain an optimal solution. Nonetheless, non-trivial instances cannot be solved, not even allowing the ILP solver to run for days of computing times. Therefore, Burke et al. (2010a) proposed various lower bounds, derived from the Monolithic formulation, that can be computed in reasonable amounts of computing times. These lower bounds, described in Sects. 3.1.1 and 3.1.2, are used within a heuristic algorithm, proposed in Burke et al. (2010a), which is described in Sect. 4.1. The results obtained with the

Monolithic formulation (1)–(17) and the heuristic algorithm based on the alternative formulations will be presented in Sects. 3.8 and 4, respectively.

3.1.1 First lower bound derived from the monolithic formulation: Surface1

The first idea suggested by Burke et al. (2010a) to solve hard CB-CTT instances to optimality or, at least, to compute good lower bounds starting from the Monolithic formulation is to ignore the penalties for violating room capacity and room stability (that is, by setting $W^{RC} = 0$ and $W^{RS} = 0$) and to add extra-constraints to bound the number of rooms used at any single time period. This translates into an ILP model, hereafter called *Surface1*, with much fewer variables and constraints than the Monolithic formulation: **y** variables can be ignored and the index r in the **x** variables is no longer necessary. The computational results that will be reported in Sect. 3.8 indicate that the resulting model solved by a general-purpose ILP solver provides, in much shorter computing times, lower bounds comparable or even better than the ones achieved by the Monolithic formulation.

3.1.2 Second lower bound derived from the monolithic formulation: Surface2

The Surface1 formulation can be thought of an extreme aggregation of the $|\mathscr{R}|$ rooms into a single multi-room of multiplicity $|\mathscr{R}|$ and capacity equal to the size of the largest room. An intermediate aggregation is to divide the rooms into two sets of multi-rooms, having capacity smaller or greater than a given threshold, to limit the number of variables and constraints in the resulting formulation but still considering all four penalties of soft constraints in the objective function. The resulting formulation, hereafter called *Surface2* (see Burke et al. 2010a for its complete description) provides lower bounds that are, on average, better than the ones achieved by Surface1 (see Sect. 3.8 for a detailed analysis).

3.2 The exact branch-and-cut algorithm of Burke et al. (2012)

Burke et al. (2012) described an exact branch-and-cut algorithm to solve CB-CTT to optimality. The proposed algorithm solves two instances (`comp01` and `comp11`) to proven optimality. For the remaining instances, the obtained lower bounds are shown in Table 1 (see Sect. 3.8). This branch-and-cut algorithm starts from an ILP model whose optimal solution cost is a valid lower bound to CB-CTT and that uses the same **x**, **v**, **u** and **y** variables and most of the constraints defined for the Monolithic formulation. Such model differs from the Monolithic one for the definition of the **z** variables and the constraints used to model curriculum compactness; indeed, binary variables z_{hq} for a given curriculum $q \in \mathscr{Q}$ and a time period $h \in \mathscr{H}$ are replaced by an integer variable $z_{dq} \in \mathbb{Z}$ that equals the number of isolated lectures of curriculum $q \in \mathscr{Q}$ on day $d \in \mathscr{D}$, and constraints (10) are replaced with the following constraints

$$\sum_{c \in \mathscr{C}_q} \sum_{r \in \mathscr{R}} (x_{chr} - x_{c,h-1,r} - x_{c,h+1,r}) \leq z_{d_h q} \qquad q \in \mathscr{Q},\ h \in \mathscr{H}. \tag{18}$$

In the following, we refer to the ILP model (1)–(9), (11)–(17), (18), as *Relaxed Monolithic* model.

Burke et al. (2012) observed that, by simply introducing constraints (18), the penalty paid for having isolated lectures for a given curriculum is at most 1 while the number of these lectures can be arbitrarily high. To achieve a valid formulation, Burke et al. (2012) proposed to add, to the Relaxed Monolithic model, an exponential (in the number of periods per day) number of inequalities, that the authors called *Cuts from Event/Free-Period Patterns* or, even, *Type 1 Cuts*, to stress that they are required to guarantee the correctness of the resulting formulation. For a given curriculum $q \in \mathscr{Q}$, let $\mathscr{B}_q$ be the index set of all n-dimensional vectors $\mathbf{b}^\beta \in \{1, -1\}^n$, $\beta = 1, \ldots, |\mathscr{B}_q|$, where n is the number of daily time periods and b_h^β, $h = 1, \ldots, n$, indicates if a lecture of curriculum q is scheduled at time period h ($b_h^\beta = 1$) or not ($b_h^\beta = -1$). For each vector $\mathbf{b}^\beta$, $\beta = 1, \ldots, |\mathscr{B}_q|$ of curriculum $q \in \mathscr{Q}$, let us indicate by π_β the number of corresponding isolated lectures and by o_β the number of schedule lectures (i.e., $o_\beta = |\{b_h^\beta : b_h^\beta = 1,\ h = 1, \ldots, n\}|$). Then, curriculum compactness constraints can be modeled by replacing constraints (10) with the following Type 1 Cuts

$$\pi_\beta \left(1 - o_\beta + \sum_{h=1}^{n} \left(b_i \sum_{c \in \mathscr{C}_q} \sum_{r \in \mathscr{R}} x_{chr}\right)\right) \leq z_{dq} \qquad q \in \mathscr{Q},\ d \in \mathscr{D},\ \beta = 1, \ldots, |\mathscr{B}_q|.$$

Burke et al. (2012) also introduced three classes of valid inequalities to improve the linear relaxation of the Relaxed Monolithic model.

1. *Implied Bounds (IB)* cuts explicitly define lower and/or upper bounds on the variables. In particular

$$1 \leq \sum_{d \in \mathscr{D}} v_{cd} \leq lct_c \qquad c \in \mathscr{C}$$

$$1 \leq \sum_{r \in \mathscr{R}} y_{cr} \leq lct_c \qquad c \in \mathscr{C}$$

$$y_{cr} \leq \sum_{h \in \mathscr{H}} x_{chr} \leq lct_c y_{cr} \qquad c \in \mathscr{C},\ r \in \mathscr{R}$$

$$u_c \leq mwd_c - 1 \qquad c \in \mathscr{C}.$$

2. *Days of Instruction (DI)* cuts take into account soft constraints penalizing insufficient number of distinct days of instruction per course and stipulate a link between $\mathbf{x}$ and $\mathbf{u}$ variables. In particular, the number of lectures of a given course $c \in \mathscr{C}$ taking place on a given day $d \in \mathscr{D}$ cannot be higher than one plus the number of lectures not necessary to maintain the spread of lectures throughout the week ($lct_c - mwd_c$), if penalty u_c is set to 0, namely

$$\sum_{h \in \mathscr{H}_d} \sum_{r \in \mathscr{R}} x_{chr} \leq 1 + lct_c - mwd_c + u_c \qquad c \in \mathscr{C},\ d \in \mathscr{D}. \tag{19}$$

Cuts (19) can be extended to cover an arbitrary subset $\widehat{\mathscr{D}}$ of days

$$\sum_{d \in \widehat{\mathscr{D}}} \sum_{h \in \mathscr{H}_d} \sum_{r \in \mathscr{R}} x_{chr} \leq |\widehat{\mathscr{D}}| + lct_c - mwd_c + u_c \qquad c \in \mathscr{C},\ \widehat{\mathscr{D}} \subset \mathscr{D}. \tag{20}$$

3. *Clique (CLI)* cuts state that at most a lecture of properly selected subsets of courses can take place at a given time period in any room. Let $\mathscr{S}$ be a collection of subsets of courses, where each subset $S \subseteq \mathscr{C}$, $S \in \mathscr{S}$, corresponds to a complete subgraph in the course-based conflict graph having a node for each course and an edge between two nodes if the two courses cannot be scheduled simultaneously (that is, the two courses are given by the same teacher and/or belong to the same curriculum). For each subset $S \in \mathscr{S}$ and each time period $h \in \mathscr{H}$, the following *Clique* cut is valid

$$\sum_{c \in S} \sum_{r \in \mathscr{R}} x_{chr} \leq 1 \qquad h \in \mathscr{H},\ S \in \mathscr{S}.$$

3.3 The two-stage ILP method of Lach and Lübbecke (2012)

Lach and Lübbecke (2012) approached CB-CTT through an ILP model. Instead of solving a formulation with three-index variables for the course/room/time period assignment (that is, x_{chr} variables of the Monolithic formulation), they decomposed the problem in two stages corresponding to two different ILP models. The goal of the first stage is to assign lectures to time periods, without explicitly considering rooms and minimizing penalties for room capacity, curriculum compactness, and minimum working days. In the second stage, the assignment of lectures to rooms is performed and room stability comes into play.

The proposed method consists of solving the two ILP models in sequence. Once the first stage assigns lectures to time periods, the second stage starts from such an assignment to assign the lectures of each time period to rooms. Under some conditions, the final solution is a global optimum, provided that enough time is given to solve the two models to optimality, but the achieved solution is usually a heuristic one. In particular, since the soft constraint of room stability is taken into account only in the second stage, the model is exact only for `UD1` (in which room stability is neglected). The proposed method obtains the optimal solutions for `test1`–`test4` instances of `UD1`. Nonetheless, the optimal solution of the first stage always provides a valid lower bound to CB-CTT.

Some additional notation is necessary to describe the ILP model solved in the first stage. Let $\mathscr{S}$ be the set of all different room capacities. Moreover, let $\mathscr{C}_{>s}$ denote the set of courses with demand larger than $s \in \mathscr{S}$ (i.e., $\mathscr{C}_{>s} = \{c \in \mathscr{C} : st_c > s\}$) and $\mathscr{R}_{>s}$ denote the set of rooms with capacity larger than s (i.e., $\mathscr{R}_{>s} = \{r \in \mathscr{R} : cap_r > s\}$). Six sets of variables are used:

- $x_{ch} \in \mathbb{B}$ equals 1 if a lecture of course $c \in \mathscr{C}$ is given at time period $h \in \mathscr{H}$;
- $w_{chs} \in \mathbb{B}$ equals 1 if the lecture course $c \in \mathscr{C}$ given at time period $h \in \mathscr{H}$ takes place in a room of capacity smaller than $s \in \mathscr{S}$;

– $r_{hq} \in \mathbb{B}$ equals 1 if there is an isolated lecture of curriculum $q \in \mathscr{Q}$ at time period $h \in \mathscr{H}$;
– $\mathbf{u}$, $\mathbf{v}$, $\mathbf{z}$ as introduced for the Monolithic formulation in Sect. 3.1.

The ILP model solved in the first stage is:

$$\min \quad W^{RC} \sum_{s \in \mathscr{S}} \sum_{c \in \mathscr{C}_{>s}} \sum_{h \in \mathscr{H}} (st_c - cap_r) w_{chs} + W^{CC} \sum_{h \in \mathscr{H}} \sum_{q \in \mathscr{Q}} r_{hq} + W^{WD} \sum_{c \in \mathscr{C}} u_c \tag{21}$$

$$\text{s.t.} \quad \sum_{h \in \mathscr{H}} x_{ch} = lct_c \qquad c \in \mathscr{C} \tag{22}$$

$$\sum_{c \in \mathscr{C}} x_{ch} \leq |\mathscr{R}| \qquad h \in \mathscr{H} \tag{23}$$

$$x_{ch} \geq w_{chs} \qquad s \in \mathscr{S},\ h \in \mathscr{H},\ c \in \mathscr{C}_{>s} \tag{24}$$

$$\sum_{c \in \mathscr{C}_{>s}} x_{ch} - w_{chs} \leq |\mathscr{R}_{>s}| \qquad s \in \mathscr{S},\ h \in \mathscr{H} \tag{25}$$

$$\sum_{h \in \mathscr{H}_d} x_{ch} \geq v_{cd} \qquad c \in \mathscr{C},\ d \in \mathscr{D} \tag{26}$$

$$\sum_{r \in \mathscr{R}} x_{chr} \leq v_{cd_h} \qquad c \in \mathscr{C},\ h \in \mathscr{H} \tag{27}$$

$$\sum_{d \in \mathscr{D}} v_{cd} \geq mwd_c - u_c \qquad c \in \mathscr{C} \tag{28}$$

$$\sum_{q \in \mathscr{C}_q} x_{ch} = r_{hq} \qquad q \in \mathscr{Q},\ h \in \mathscr{H} \tag{29}$$

$$r_{hq} - r_{h-1,q} - x_{h+1,r} \leq z_{hq} \qquad q \in \mathscr{Q},\ h \in \mathscr{H} \tag{30}$$

$$\sum_{c \in \mathscr{C}_t} x_{ch} \leq 1 \qquad h \in \mathscr{H},\ t \in \mathscr{T} \tag{31}$$

$$x_{ch} \in \mathbb{B} \qquad c \in \mathscr{C},\ h \in \mathscr{H} \tag{32}$$

$$w_{chs} \in \mathbb{B} \qquad s \in \mathscr{S},\ c \in \mathscr{C}_{>s},\ h \in \mathscr{H} \tag{33}$$

$$v_{cd} \in \mathbb{B} \qquad c \in \mathscr{C},\ d \in \mathscr{D} \tag{34}$$

$$u_c \in \mathbb{Z} \qquad c \in \mathscr{C} \tag{35}$$

$$z_{hq}, r_{hq} \in \mathbb{B} \qquad h \in \mathscr{H},\ q \in \mathscr{Q}. \tag{36}$$

The objective function (21) allows to minimize the penalties for room occupancy, curriculum compactness, and minimum working days. Constraints (22) force the solution to schedule all lct_c lectures of each course $c \in \mathscr{C}$. No more than $|\mathscr{R}|$ lectures can be scheduled at a given time period $h \in \mathscr{H}$ (see (23)). The relationship between the $\mathbf{x}$ and $\mathbf{w}$ variables is stated by constraints (24)–(25). The correct setting of $\mathbf{v}$ variables is assured by constraints (26)–(27). Penalties for minimum working days are

paid because of constraints (28). Constraints (29) establish the relationship between **x** and **r** variables. The correct penalty for curriculum compactness is guaranteed by constraints (30). No two simultaneous lectures given by the same teacher can be scheduled because of constraints (31). Integrality is imposed on the variables by constraints (32)–(36).

The ILP model of the second stage corresponds to the standard formulation of a (one-sided perfect) matching in a bipartite graph with additional constraints to take room stability into account. The bipartite graph is defined on the values of the **x** variables in the optimal solution found in the first stage. For a detailed description of the second stage, the reader is referred to Lach and Lübbecke (2012).

3.4 The divide-and-conquer approach of Hao and Benlic (2011)

To derive valid lower bounds for CB-CTT, Hao and Benlic (2011) developed a partition-based approach that follows the divide-and-conquer principle. From the original problem, a selected subset of the constraints are eliminated in such a way that the resulting problem is decomposable into smaller independent subproblems. Each subproblem can be solved using the ILP model (21)–(36) proposed by Lach and Lübbecke (2012). A valid lower bound to the whole CB-CTT is then obtained by summing up the optimal solution costs of the individual subproblems.

The rational of the proposed approach can be more formally described as follows. Let $\{X_p\}_{1 \le p \le k}$ denote a partition of the courses of the set $\mathscr{C}$ in k classes (i.e., $\cup_{p=1}^{k} X_p = \mathscr{C}$ and $X_i \cap X_j = \varnothing, i, j \in \{1, \ldots, k\}, i \neq j$). Let $P(X_p)$, $p = 1, \ldots, k$, denote the p-th subproblem derived from model (21)–(36) by keeping only the variables of the set X_p; problem $P(X_p)$ is clearly a relaxation of the whole problem (21)–(36). If LB_p, $p = 1, \ldots, k$, indicates the optimal solution cost of problem $P(X_p)$, then $\sum_{p=1}^{k} LB_p$ represents a valid lower bound to CB-CTT.

The divide-and-conquer approach of Hao and Benlic (2011) consists of three main steps:

1. Generate a partition $\{X_p\}$ of the set of courses $\mathscr{C}$.
2. Solve each subproblem $P(X_p)$, $p = 1, \ldots, k$, with a generic ILP solver to compute lower bound LB_p.
3. Sum up the k values LB_p to achieve the final lower bound to CB-CTT.

The partition defined at Step 1 is achieved by running an *Iterated Tabu Search* (ITS)—we refer the reader to Hao and Benlic (2011) for all the details. The ITS determines a k-partition of the graph $G = \{V, E\}$ defined as follows: the set of nodes V contains a node for each course $c \in \mathscr{C}$, and an edge (c_1, c_2) belongs to E if there exists at least a curriculum $q \in \mathscr{Q}$ containing both courses (i.e., $c_1, c_2 \in \mathscr{C}_q$ for some $q \in \mathscr{Q}$). A k-partition is obtained by iteratively removing, from graph G, some of the edges until the graph contains exactly k connected components. In the end, if any of the edges added to the original edge set E, for curriculum q is removed, all other edges induced by the same curriculum q are removed.

In the problem (21)–(36) corresponding to the k-partition defined at Step 1, a subset of the constraints (29)–(30) corresponding to the eliminated edges can be removed.

Similarly, constraints modeling room occupancy (22)–(24) and conflict constraints (29) that link subproblems of a partition can be relaxed. The resulting problem is decomposable in k different subproblems that can be independently solved.

3.5 The column generation method of Cacchiani et al. (2013)

Cacchiani et al. (2013) presented different formulations for CB-CTT all featuring exponentially many variables. In the following, we focus on the formulation, called *Two Weekly Schedule Types* (2WST), that achieved the best computational results. Such a formulation presents two sets of binary variables. The first set of variables is represented by all possible feasible assignments of lectures to rooms and time periods and considers penalties for room capacity and room stability. The second set of variables is represented by all possible feasible assignments of lectures to time periods and considers penalties for curriculum compactness and minimum working days. To introduce the formulation, we need some additional definitions.

A feasible assignment of lectures to rooms and time periods is a vector $\mathbf{x} \in \mathbb{B}^{|\mathscr{C}|\times|\mathscr{H}|\times|\mathscr{R}|}$ made up of components $x_{chr} \in \mathbb{B}$ representing the assignment of course $c \in \mathscr{C}$ to room $r \in \mathscr{R}$ at time period $h \in \mathscr{H}$. Vector $\mathbf{x}$ has to be such that: all lectures of all courses are scheduled; no more than a lecture takes place in the same room at the same time period; no two lectures given by the same teacher or belonging to the same curriculum are scheduled simultaneously. The cost of such an assignment encompasses room capacity and room stability only. Therefore, the set $\mathscr{X}$ of all feasible assignments courses/time periods/rooms correspond to all vectors $\mathbf{x}$ that satisfy constraints (2) + (3) + (5) + (6) + (13) + (17) and constraints

$$\sum_{h\in\mathscr{H}} x_{chr} \leq lct_c y_{cr} \qquad c \in \mathscr{C},\ r \in \mathscr{R},$$

where variables $\mathbf{x}$ and $\mathbf{y}$ are defined as in Sect. 3.1. The cost of any vector $\mathbf{x} \in \mathscr{X}$ is defined as

$$c(\mathbf{x}) = W^{RC} \sum_{c\in\mathscr{C}} \sum_{h\in\mathscr{H}} \sum_{\substack{r\in\mathscr{R}:\\ cap_r < st_c}} (st_c - cap_r) x_{chr} + W^{RS}\Big(\sum_{c\in\mathscr{C}} \sum_{r\in\mathscr{R}} (y_{cr} - 1)\Big).$$

Similarly, a feasible assignment of lectures to time periods is a vector $\theta \in \mathbb{B}^{|\mathscr{C}|\times|\mathscr{H}|}$ made up of components $\theta_{ch} \in \mathbb{B}$ that indicates if a lecture of course $c \in \mathscr{C}$ is scheduled at time period $h \in \mathscr{H}$. Vector θ has to be such that: all lectures of all courses are scheduled; no two lectures given by the same teacher or belonging to the same curriculum are scheduled simultaneously. The cost of such an assignment covers curriculum compactness and minimum working days only. Therefore, the set Θ of all feasible assignments courses/time periods correspond to all vectors θ that satisfy the following constraints

$$\sum_{h\in\mathscr{H}} \theta_{ch} = lct_c \qquad c \in \mathscr{C}$$

$$\sum_{c \in \mathscr{C}_q} \theta_{ch} \leq 1 \qquad h \in \mathscr{H},\ q \in \mathscr{Q}$$

$$\sum_{c \in \mathscr{C}_t} \theta_{ch} \leq 1 \qquad h \in \mathscr{H},\ t \in \mathscr{T}$$

$$\sum_{h \in \mathscr{H}_d} \theta_{ch} \geq v_{cd} \qquad c \in \mathscr{C},\ d \in \mathscr{D}$$

$$\sum_{d \in \mathscr{D}} v_{cd} + u_c \geq mwd_c \qquad c \in \mathscr{C}$$

$$\sum_{c \in \mathscr{C}_q} (\theta_{ch} - \theta_{c,h-1} - \theta_{c,h+1}) \leq z_{hq} \qquad h \in \mathscr{H},\ q \in \mathscr{Q}$$

$$\theta_{ch} \in \mathbb{B} \qquad c \in \mathscr{C},\ h \in \mathscr{H}$$

$$+ (14) + (15) + (16)$$

where variables $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{z}$ are defined as in Sect. 3.1, and the cost $d(\theta)$ of any vector $\theta \in \Theta$ is defined as

$$d(\theta) = W^{WD} \sum_{c \in \mathscr{C}} u_c + W^{CC} \sum_{h \in \mathscr{H}} \sum_{q \in \mathscr{Q}} z_{hq}.$$

Given the definition of the sets $\mathscr{X}$ and Θ. CB-CTT can be formulated as follows. Let $\xi_\ell \in \mathbb{B}$ be a binary variable equal to 1 if and only if vector $\mathbf{x}^\ell \in \mathscr{X}$ is selected, and let $\varphi_\ell \in \mathbb{B}$ be a binary variable equal to 1 if and only if vector $\theta^\ell \in \Theta$ is selected. The 2WST formulation reads as follows:

$$\min \quad \sum_{\ell \in \mathscr{X}} c_\ell \xi_\ell + \sum_{\ell \in \Theta} d_\ell \varphi_\ell \tag{37}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathscr{X}} \xi_\ell = 1 \tag{38}$$

$$\sum_{\ell \in \Theta} \varphi_\ell = 1 \tag{39}$$

$$\sum_{\ell \in \mathscr{X}} \sum_{r \in \mathscr{R}} x^\ell_{chr} \xi_\ell = \sum_{\ell \in \Theta} \theta^\ell_{ch} \varphi_\ell \qquad c \in \mathscr{C},\ h \in \mathscr{H} \tag{40}$$

$$\xi_\ell \in \mathbb{B} \qquad \ell \in \mathscr{X} \tag{41}$$

$$\varphi_\ell \in \mathbb{B} \qquad \ell \in \Theta. \tag{42}$$

The objective function (37) aims at minimizing the total cost of the selected assignments. Constraint (38) force to select exactly one of the courses/time periods/rooms assignments of the set $\mathscr{X}$. Constraint (39) force to select exactly one of the courses/time periods assignments of the set Θ. The correspondence between the two types of variables is stipulated by constraints (40) that force the φ variables to schedule a lecture of course $c \in \mathscr{C}$ at time period $h \in \mathscr{H}$ if and only if such a lecture

is scheduled in the selected variable ξ_ℓ. Integrality on the variables is imposed by constraints (41)–(42).

Using formulation (37)–(42) for directly solving CB-CTT is not doable, not even for small/medium size CB-CTT instances. Even solving the linear relaxation of (37)–(42) is hard due to the exponential number of variables and the complexity of generating columns. Therefore, Cacchiani et al. (2013) proposed to compute a valid lower bound to CB-CTT by dropping constraints (40) and solving the remaining problem, which is clearly decomposable in the two separate subproblems corresponding to finding the minimum cost assignment of set $\mathscr{X}$ and set Θ. The computed lower bound is given by $\min_{\mathbf{x}\in\mathscr{X}}\{c(\mathbf{x})\}+\min_{\theta\in\Theta}\{d(\theta)\}$. If the problem of finding the min-cost assignment θ is still computationally intractable, Cacchiani et al. (2013) suggested to reformulate the problem as an ILP model with exponentially many variables and to compute the optimal solution cost of its linear relaxation.

3.6 Asín Achá and Nieuwenhuis (2014): Formulations via SAT and MaxSAT

Asín Achá and Nieuwenhuis (2014) proposed different encodings of CB-CTT to be solved by SAT solvers. Such encodings differ for the subsets of hard and soft constraints modeled and for the way these constraints are defined. The proposed encodings can provide either lower or upper bounds to CB-CTT. Optimality can also be proved.

All the encodings are based on a Basic SAT encoding, where all constraints (both hard and soft) of CB-CTT are defined as hard. Therefore, for those instances for which the SAT solver finds a solution, this has a zero-cost. The Basic SAT encoding defines the following four sets of propositional variables:

- x_{ch}: course $c \in \mathscr{C}$ takes place at time period $h \in \mathscr{H}$;
- v_{cd}: course $c \in \mathscr{C}$ takes place on day $d \in \mathscr{D}$;
- y_{cr}: course $c \in \mathscr{C}$ is held in room $r \in \mathscr{R}$;
- z_{hq}: one of the lectures of curriculum $q \in \mathscr{Q}$ takes place at time period $h \in \mathscr{H}$.

The following relationships among the propositional variables are stipulated ($\neg x$ indicates the negation of literal x, and $\vee$ indicates a disjunction):

- If a course $c \in \mathscr{C}$ takes place at a time period $h \in \mathscr{H}$, then it takes place on day d_h

$$\neg x_{ch} \vee v_{cd_h} \qquad c \in \mathscr{C},\ h \in \mathscr{H}.$$

- If a course $c \in \mathscr{C}$ is held on day $d \in \mathscr{D}$, it has to take place at, at least, one of the time periods of the set $\mathscr{H}_d = \{h_1, h_2, \ldots, h_n\}$ of day d

$$\neg v_{cd} \vee x_{ch_1} \vee x_{ch_2} \vee \ldots \vee x_{ch_n} \qquad c \in \mathscr{C},\ d \in \mathscr{D}.$$

- If a lecture of course $c \in \mathscr{C}$ is held at time period $h \in \mathscr{H}$, all the curricula of the set $\mathscr{Q}_c$ occur at time period h

$$\neg x_{ch} \vee z_{hq} \qquad c \in \mathscr{C},\ h \in \mathscr{H},\ q \in \mathscr{Q}_c$$

– If a curriculum $q \in \mathcal{Q}$ takes place at time period $h \in \mathcal{H}$, then at least one of the courses of the set $\mathcal{C}_q = \{c_1, c_2, \ldots, c_n\}$ must occur at time period h

$$\neg z_{hq} \vee x_{c_1 h} \vee x_{c_2 h} \vee \ldots \vee x_{c_n h} \qquad h \in \mathcal{H},\ q \in \mathcal{Q}.$$

Hard and soft constraints of CB-CTT are all encoded as hard constraints through the following clauses, where $eq(k, S)$ ($geq(k, S)$, resp.) represents a set of literals S that are satisfied if and only if exactly (at least, resp.) k of the variables of S are true:

– *Curriculum Clashes*: No two courses $c_1, c_2 \in \mathcal{C}$ for which there exists a curriculum $q \in \mathcal{Q}$ such that $c_1, c_2 \in \mathcal{C}_q$ can be scheduled at the same time period $h \in \mathcal{H}$

$$\neg x_{c_1 h} \vee \neg x_{c_2 h} \qquad h \in \mathcal{H},\ c_1, c_2 \in \mathcal{C} : \mathcal{Q}_{c_1} \cap \mathcal{Q}_{c_2} \neq \varnothing.$$

– *Teacher Clashes*: No two courses c_1 and c_2 given by the same teacher $t \in \mathcal{T}$ may be scheduled at the same time period $h \in \mathcal{H}$

$$\neg x_{c_1 h} \vee \neg x_{c_2 h} \qquad h \in \mathcal{H},\ t \in \mathcal{T},\ c_1, c_2 \in \mathcal{C}_t.$$

– *Room Clashes*: No two courses c_1 and c_2 can be scheduled in the same room $r \in \mathcal{R}$ at the same time period $h \in \mathcal{H}$

$$\neg x_{c_1 h} \vee \neg x_{c_2 h} \vee \neg y_{c_1 r} \vee \neg y_{c_2 r} \qquad h \in \mathcal{H},\ r \in \mathcal{R},\ c_1, c_2 \in \mathcal{C}.$$

– *Time Period Availability*: If a course $c \in \mathcal{C}$ cannot be scheduled at time period $h \in \mathcal{H}$, the one-literal clause $\neg x_{ch}$ is added.
– *Number of Lectures*: Exactly lct_c lectures must be scheduled for course $c \in \mathcal{C}$

$$eq(lct_c, \{x_{ch_1}, x_{ch_2}, \ldots, x_{ch_n}\}) \qquad c \in \mathcal{C},$$

where $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$.
– *Room Capacity*: Each course must be scheduled into a room in which it fits

$$\neg y_{cr} \qquad c \in \mathcal{C},\ r \in \mathcal{R} : cap_r < st_c.$$

– *Minimum Working Days*: For each course $c \in \mathcal{C}$, at least mwd_c literals of the set $v_{cd_1}, v_{cd_2}, \ldots, v_{cd_{|D|}}$ must be true, where $\mathcal{D} = \{d_1, d_2, \ldots, d_{|D|}\}$:

$$geq(mwd_c, \{v_{cd_1}, v_{cd_2}, \ldots, v_{cd_{|D|}}\}) \qquad c \in \mathcal{C}.$$

– *Isolated Lectures*: If some curriculum $q \in \mathcal{Q}$ takes place at a given time period $h \in \mathcal{H}$, then q must also occur at time period $h-1$ (if h is not the first time period of the day) and/or time period $h+1$ (if h is not the last time period of the day)

$$\neg z_{hq} \vee z_{h-1,q} \vee z_{h+1,q} \qquad c \in \mathcal{C},\ h \in \mathcal{H}.$$

– *Room Stability*: Each course must be assigned to exactly one room of the set $\mathscr{R} = \{r_1, r_2, \ldots, r_n\}$

$$eq(1, \{y_{cr_1}, y_{cr_2}, \ldots, y_{cr_n}\}) \qquad c \in \mathscr{C}.$$

Asín Achá and Nieuwenhuis (2014) proposed different encodings based on the Basic one just outlined. In particular, they proposed: (i) to relax the soft constraints of the isolated lectures and to solve the derived model through a *Partial MaxSAT* solver, (ii) to relax also the soft constraints of the minimum working days and to solve the derived model through a *Weighted Partial MaxSAT* solver, (iii) to relax all the soft constraints and to solve the derived model through a *Partial MaxSAT* solver, and (iv) to use a branch-and-bound algorithm for the *Weighted Partial MaxSAT* problem. Among these encodings, the one achieving the best lower bounds consists of relaxing all the four CB-CTT soft constraints (i.e., room capacity, room stability, curriculum compactness, and minimum working days). Each violation of a soft constraint is described by as many clauses as its cost—that is, for each curriculum $q \in \mathscr{Q}$ and each time period $h \in \mathscr{H}$, W^{CC} clauses are added to take into account curriculum compactness; for each course $c \in \mathscr{C}$, W^{WD} clauses are added to take into account minimum working days; and so on. This complete encoding, besides providing valid lower bounds to CB-CTT, allows to achieve the optimal solutions for eleven instances of ITC-2007 (see Sect. 3.8). The best heuristic solutions are instead obtained by considering all the different types of encodings, and selecting, for each instance, the best result achieved. The different heuristic algorithms are tested on the instances of ITC-2007 (`comp01-comp21`) and on additional instances (`test1-test4`) and (`DDS1-DDS7`). The obtained results are presented in Sect. 4.3.

3.7 Banbara et al. (2013): Answer Set Programming

Banbara et al. (2013) used *Answer Set Programming* (ASP) as a modeling language for CB-CTT. ASP is useful for modeling combinatorial problems in computer science and artificial intelligence. Each hard constraint was expressed by integrity constraints and aggregates of ASP. In particular, the authors presented two different encodings called *Direct encoding* and *Linked encoding* and showed that the linked encoding is faster and can be more scalable to the number of courses than the direct encoding. For the soft constraints, they used the predicate $penalty(S, V, C)$, which is intended to express that a soft constraint S is violated by V and its penalty cost is C. The proposed approach is tested on the ITC-2007 instances (`comp01-comp21`), and on `DDS1-DDS7`, `test1-test4` and `erlangen*`. Since no lower bound is reported in the corresponding paper, the obtained results are presented in Sect. 4.3.

3.8 Computational results

In this section, we report a computational comparison of the performance of the different approaches described in Sects. 3.1–3.6. The comparison is made by considering

the 21 instances comp01–comp21 described in Sect. 2.1, that have been used as benchmark to test all exact methods and lower bounds proposed in the literature.

For each of the 21 instances, Table 1 reports the name (Inst) and the best-known upper bound (WBest), taken from the website http://satt.diegm.uniud.it/ctt, which is indicated with an asterisk whenever it corresponds to the optimal solution cost. Then, for the following nine approaches, we report the achieved lower bound and its percentage with respect to the best-known upper bound:

- BMPR10—Column LB0: Monolithic formulation (see Sect. 3.1) proposed by Burke et al. (2010a);
- BMPR10—Column LB1: Surface1 (see Sect. 3.1.1) proposed in Burke et al. (2010a);
- BMPR10—Column LB2: Surface2 (see Sect. 3.1.2) proposed in Burke et al. (2010a);
- BMPR12—Column LB0: branch-and-cut with both Type 1 and Type 2 cuts (see Sect. 3.2) proposed in Burke et al. (2012);
- BMPR12—Column LB1: Surface1 plus Implied Bounds cuts (see Sect. 3.2) reported in Burke et al. (2012);
- LL12: first stage of the two-stage approach of Lach and Lübbecke (2012) (see Sect. 3.3);
- HB11: divide-and-conquer approach of Hao and Benlic (2011) (see Sect. 3.4);
- CCRT13: column generation of Cacchiani et al. (2013) (see Sect. 3.5);
- AN14: Partial MaxSAT of Asín Achá and Nieuwenhuis (2014) (see Sect. 3.6).

For each instance, the best lower bound found by the compared approaches is highlighted in bold. In lines labeled Avg1–14 and Avg1–21, we indicate the number of times each method achieves the best-known lower bound and the corresponding average percentage lower bounds over the first 14 and all 21 instances, respectively. At the time of writing this paper, the website http://satt.diegm.uniud.it/ctt reports, as best-known lower bounds for the five instances comp05, comp09, comp18, comp19, and comp21, the following values 211, 96, 61, 57, and 74, respectively, that, to the best of our knowledge, have not appeared in any published paper available from the literature.

Table 1 indicates that none of the exact approaches developed so far can guarantee to provide really tight lower bounds on all benchmark instances. Considering the different machines used and the different time limits imposed on the algorithms, it is hard to have a fair comparison among the different approaches. Nonetheless, the divide-and-conquer algorithm of Hao and Benlic (2011) (although with very large computing times), the column generation of Cacchiani et al. (2013), and the Partial MaxSAT of Asín Achá and Nieuwenhuis (2014) seem to be, globally, the three best approaches currently available.

However, the choice of the approach to be used depends on the characteristics of the problem to be solved. In case of very constrained problems, SAT-type methodologies seem to be more promising. If the problem does not contain constraints related to a "group of rooms" (such as room stability), a two-stage ILP method, such that proposed by Lach and Lübbecke (2012), has certainly advantages, as room assignment can be performed in a second stage, using a matching procedure. Due to the wide variety of

Table 1 Comparison of the lower bounds achieved by the different approaches described earlier in this section (UD2)

Inst	WBest	BMPR10						BMPR12				LL12		HB11		CCRT13		AN14	
		LB0	%	LB1	%	LB2	%	LB0	%	LB1	%	LB	%	LB	%	LB	%	LB	%
comp01	5*	4	80.00	0	0.00	**5**	100.00	4	80.00	**5**	100.00	4	80.00	4	80.00	**5**	100.00	0	0.00
comp02	24	1	4.17	0	0.00	1	4.17	0	0.00	6	25.00	11	45.83	12	50.00	**16**	66.67	**16**	66.67
comp03	64	25	39.06	25	39.06	33	51.56	0	0.00	43	67.19	25	39.06	38	59.38	**52**	81.25	28	43.75
comp04	35*	8	22.86	**35**	100.00	**35**	100.00	0	0.00	2	5.71	28	80.00	**35**	100.00	**35**	100.00	**35**	100.00
comp05	284	111	39.08	119	41.90	114	40.14	95	33.45	**183**	64.44	108	38.03	**183**	64.44	166	58.45	48	16.90
comp06	27*	12	44.44	13	48.15	16	59.26	0	0.00	6	22.22	10	37.04	22	81.48	11	40.74	**27**	100.00
comp07	6*	0	0.00	**6**	100.00	**6**	100.00	0	0.00	0	0.00	**6**	100.00	**6**	100.00	**6**	100.00	**6**	100.00
comp08	37*	11	29.73	**37**	100.00	**37**	100.00	0	0.00	2	5.41	**37**	100.00	**37**	100.00	**37**	100.00	**37**	100.00
comp09	96*	21	21.88	68	70.83	66	68.75	0	0.00	0	0.00	46	47.92	72	75.00	**92**	95.83	35	36.46
comp10	4*	2	50.00	3	75.00	**4**	100.00	0	0.00	0	0.00	**4**	100.00	**4**	100.00	2	50.00	**4**	100.00
comp11	0*	**0**	100.00	**0**	100.00	**0**	100.00	**0**	100.00	**0**	100.00	**0**	100.00	**0**	100.00	**0**	100.00	**0**	100.00
comp12	298	39	13.09	101	33.89	95	31.88	0	0.00	5	1.68	53	17.79	**109**	36.58	100	33.56	99	33.22
comp13	59*	14	23.73	52	88.14	54	91.53	3	5.08	0	0.00	41	69.49	**59**	100.00	57	96.61	**59**	100.00
comp14	51*	40	78.43	41	80.39	42	82.35	0	0.00	0	0.00	46	90.20	**51**	100.00	48	94.12	**51**	100.00
Avg1-14		1	39.03	4	62.67	6	73.55	1	15.61	3	27.97	4	67.53	9	81.92	8	79.80	9	71.21

Table 1 continued

Inst	WBest	BMPR10						BMPR12				LL12		HB11		CCRT13		AN14	
		LB0	%	LB1	%	LB2	%	LB0	%	LB1	%	LB	%	LB	%	LB	%	LB	%
comp15	64													38	59.38	**52**	81.25	28	43.75
comp16	18*													16	88.89	13	72.22	**18**	100.00
comp17	56*													48	85.71	48	85.71	**56**	100.00
comp18	61*													24	39.34	**52**	85.25	27	44.26
comp19	57*													**56**	98.25	48	84.21	46	80.70
comp20	4*													2	50.00	**4**	100.00	**4**	100.00
comp21	74*													61	82.43	**68**	91.89	42	56.76
Avg1–21														10	78.61	12	81.80	12	72.50

BMPR10 lower bounds obtained by Burke et al. (2010a) with Monolithic formulation (LB0), Surface1 (LB1) and Surface2 (LB2)—Intel Pentium 4 3.20 GHz—Cplex 11−tl = 31200" (Monolithic and Surface2), tl = 780" (Surface1);
BMPR12 lower bounds obtained by Burke et al. (2012) by branch-and-cut with both Type 1 and 2 cuts (LB0) and by Surface1 plus IB Cuts (LB1)—Intel Pentium 4 3.20 GHz—Cplex 10 and Cplex 12.1−tl = 7200" (Branch-and-Cut), tl = 1800" (Surface1);
LL12 first stage of Lach and Lübbecke (2012)—Linux PC 3.4 GHz—Cplex 11.0.1−tl = 13000";
HB11 divide-and-conquer of Hao and Benlic (2011)—Xeon E5440 2.83 GHz—COIN-OR 2.2.2−tl = 25200" per subproblem;
CCRT13 column generation of Cacchiani et al. (2013)—Intel Xeon E5310 Dual Core 1.6 GHz—Cplex 11.2−tl = 22800";
AN14 Partial MaxSAT of Asín Achá and Nieuwenhuis (2014) using Barcelogic Partial MaxSAT solver—2100 MHz AMD-Opteron—tl = 100,000"

features, especially arising in real-world university timetabling problems, and to the many related constraints and objectives, it is not possible to resort to a single approach, but rather to a set of alternative approaches.

4 Heuristic algorithms

In this section, we describe the most effective heuristic algorithms proposed for CB-CTT. In particular, we illustrate the algorithms classified in the first five places according to the ranking of the ITC-2007 competition, the seminal works and the most recent effective methods for CB-CTT. We classify them in ILP-based heuristic algorithms (Sect. 4.1) and metaheuristic algorithms (Sect. 4.2). We further distinguish the metaheuristic algorithms in tabu search methods, simulated annealing methods and hybrid methods. Clearly, this classification is not strict: most of the methods are complex ones and incorporate many ingredients coming from different classes of algorithms. In Sect. 4.3, we report the results of the described methods on ITC-2007 instances.

As it will be shown in Sect. 4.3, metaheuristic algorithms often perform better than ILP-based ones. In addition, hybridized methods are often very effective, since they incorporate good features from different types of algorithms. Methods based on constraint satisfaction are also very effective, even if sometimes they require longer computing times.

4.1 ILP-based heuristic algorithms

These methods are based on the mathematical models described in Sect. 3.

In Burke et al. (2010a), the authors propose the relaxed mathematical formulations *Surface1* and *Surface2*, described in Sect. 3.1. Based on these formulations, a heuristic algorithm is proposed. The idea is to consider, in a first phase, only one computationally difficult component of the full problem and the associated subset of objectives, formulated as an ILP model (i.e., *Surface1* and *Surface2*), and derive feasible solutions for it. These solutions define the neighborhoods to dive into in a second phase. More precisely, in the first phase an objective-restricted neighborhood generator is used to find an assignment of events to periods, such that at most a given number of events take place at any given period, but disregarding other issues of the assignment of rooms, such as room capacities. In the diving phase, two value restricted submodels are defined: *PeriodFixed dives*, in which the periods of all the courses are fixed to be those obtained from the surface solution, and *DayFixed dives*, in which the days of all the courses are fixed (but not the specific period of the day). The authors consider three configurations of their algorithm, according to the computing time available: the first one (1 CPU time unit) consists of solving *Surface1* and then diving according to the *PeriodFixed dive*, the second one (10 CPU time units) and the third one (40 CPU time units) solve *Surface2* and then *PeriodFixed dive* followed by *DayFixed dive* for different time limits. The algorithm is tested on the instances of ITC-2007 (`comp01-comp14`). The obtained results are presented in Table 2 (see Sect. 4.3).

In Lach and Lübbecke (2012), the two-stage mathematical model described in Sect. 3.3 is proposed. This model is based on the decomposition of the problem in two stages.

Table 2 Comparison of the best solution values found by the heuristic methods (UD2)

Inst	WBest	Best	M09	BMPR10	LH10	LHG11	LL12	AT12	BDS12	BSTIS13	AN14	K14
comp01	5*	5	**5**	9	**5**	**5**	12	**5**	**5**	**5**	**5**	**5**
comp02	24	24	43	63	34	40	46	26	41	125	**24**	34
comp03	64	66	72	123	70	71	**66**	70	**66**	196	111	67
comp04	35*	35	**35**	36	38	39	38	**35**	**35**	36	**35**	**35**
comp05	284	295	298	629	298	298	368	**295**	301	947	1343	299
comp06	27*	27	41	46	47	47	51	30	43	155	**27**	40
comp07	6*	6	14	45	19	21	25	7	18	79	**6**	8
comp08	37*	37	39	41	43	43	44	**37**	39	39	**37**	39
comp09	96*	96	103	105	99	101	99	102	**96**	264	171	98
comp10	4*	4	9	23	16	18	16	5	15	**4**	**4**	9
comp11	0*	0	**0**	12	**0**	**0**	7	**0**	**0**	**0**	**0**	**0**
comp12	298	306	331	785	320	320	548	315	320	1114	977	**306**
comp13	59*	59	66	67	65	65	66	**59**	64	112	**59**	**59**
comp14	51*	51	53	55	52	55	53	61	53	52	**51**	53
No. of best			3	0	2	2	1	6	5	3	**10**	5
No. of best only			0	0	0	0	0	1	1	0	**4**	1
Avg gap 14			16.7 %	44.1 %	18.7 %	20.7 %	36.4 %	**5.9 %**	17.3 %	41.7 %	16.5 %	11.2 %

Table 2 continued

Inst	WBest	Best	M09	BMPR10	LH10	LHG11	LL12	AT12	BDS12	BSTIS13	AN14	K14
comp15	64	66			69			69	**66**	196	111	**66**
comp16	18*	18			38			**18**	28	28	**18**	28
comp17	56*	56			80			60	71	171	**56**	67
comp18	61*	65			67			69	69	184	83	**65**
comp19	57*	57			59			**57**	60	91	**57**	60
comp20	4*	4			35			7	29	80	**4**	17
comp21	74*	86			105			**86**	89	232	**86**	87
No. of best			-	-	2	-	-	9	6	3	**15**	7
No. of best only			-	-	0	-	-	1	1	0	**6**	2
Avg gap 21			-	-	22.0 %	-	-	**6.8 %**	19.0 %	48.3 %	14.0 %	13.9 %

M09 Müller (2009)—best of 100 runs, 1 CPU time unit;
BMPR10 Burke et al. (2010a)—40 CPU time units (1 CPU time unit = 780 s)—Intel Pentium 4 3.2 GHz;
LH10 Lü and Hao (2010)—best over 100 runs, 1 CPU time unit (=390 s)—3.4 GHz PC;
LHG11 Lü et al. (2011)—best of 10 runs, 1 CPU time unit—3.4 GHz computer;
LL12 Lach and Lübbecke (2012)—40 CPU time units (1 CPU time unit = 400 s)—Linux PC 3.4 GHz;
AT12 Abdullah and Turabieh (2012)—best of 10 runs, 1 CPU time unit (=600 s)—Intel Pentium 4 2.33 GHz;
BDS12 Bellio et al. (2012)—1 CPU time unit, best solutions found during the experimentation—Intel Quad core;
BSTIS13 Banbara et al. (2013)—3 h—2.66 GHz Intel Xeon;
AN14 Asín Achá and Nieuwenhuis (2014)—100,000 s—2.1 GHz AMD-Opteron;
K14 Kiefer et al. (2014)—best over 10 runs, 1 CPU time unit—Core i5-3550 3.30 GHz

The soft constraint of room stability is taken into account only in the second stage and the model gives therefore heuristic solutions for UD2. The model is tested on the instances of ITC-2007 (comp01–comp14) and on additional instances reflecting the timetabling situation at the Technical University of Berlin (DDS1–DDS7). The obtained results are presented in Table 2 (see Sect. 4.3).

4.2 Metaheuristic algorithms

In this section, we describe effective metaheuristic algorithms for CB-CTT. They are usually characterized by the definition and exploration of several different neighborhoods, obtained, for example, by changing the time period assigned to a lecture, or by changing the room assignment for a lecture or by interchanging Kempe chains (i.e., by moving a subset of courses in the timetable to any other time period, while always keeping the feasibility of the solution). More complex neighborhoods are also used and in some cases it is allowed to accept moves to infeasible solutions. The metaheuristic algorithms are often divided in phases: the first phase is used for constructing an initial solution and the subsequent phases are used to explore the neighborhoods to determine an improved solution. Another common feature of these algorithms is to introduce a way to avoid being trapped in local optima. To this aim tabu search, simulated annealing or great deluge are used. Most of the methods combine more components.

In the following sections, we give an overview of these metaheuristic algorithms.

4.2.1 Tabu search algorithms

As described in the seminal paper by Di Gaspero and Schaerf (2003), one of the most critical choices in local search is the definition of the neighborhood structure. In their paper, the authors investigate and compare three different structures: (i) *neighborhood union*: in this case the union of many neighborhoods is considered and, at each iteration, the local search algorithm selects a move belonging to any of the components; (ii) *neighborhood composition*: in this case chains of moves belonging to different neighborhoods are considered; (iii) *token-ring search*: given an initial state and a set of algorithms based on different neighborhood functions, the token-ring search makes circularly a run of each algorithm, always starting from the best solution found by the previous one. The search space is composed of the assignments for which all the lectures are scheduled and assigned to distinct periods, and the availability constraints are respected. The violations of the other hard constraint are penalized in the cost function, together with the soft constraints. Two types of neighborhoods are considered: the first one is defined by changing the time period assigned to a lecture of a given course to a new one which satisfies the availability constraints; the second one is defined by changing the room assigned to a lecture in a given time period. Hill climbing and tabu search algorithms are compared on these two types of neighborhoods, on their union, on their composition, and combined in a token-ring search. The algorithms are tested on four instances of the Udine University (test1–test4).

Atsuta et al. (2008) propose to formulate the timetabling instances of CB-CTT as instances of the *constraint satisfaction problem* (CSP), and then to apply a general purpose CSP solver, which adopts tabu search and iterated local search procedures, to find their solutions. The proposed approach is tested on the ITC-2007 instances, and classified in the third position at the ITC-2007 competition.

In Clark et al. (2009), the authors present a solver called *QuickFix*. It constructs an initial assignment of teacher, period and room to each lecture of each course, by allowing the violation of constraints. Then, it picks a violated constraint, selected randomly out of the set of violated constraints, with the selection weighted by constraint weights (hard constraints have a very high weight while soft constraints have the weight as in the objective function). A move is applied to repair the constraint: in particular, the solution assignments are updated by performing swap moves (teacher–period swap or room–period swap). To avoid being trapped in local minima, QuikFix employs two tabu lists that store the recent and the bad moves, and adopts the strategic oscillation technique, a mechanism for escaping from local minima by modifying the weights of the constraints. In addition, if the algorithm is not able to find a feasible solution better than the initial one within a given number of moves, it re-starts the search from the best solution found so far. The algorithm is tested on the instances of ITC-2007 (`comp01-comp14`) and classified in the fifth position at the ITC-2007 competition.

An *adaptive tabu search* is proposed in Lü and Hao (2010). The algorithm consists of three phases: initialization, intensification and diversification. In the initialization phase, a sequential greedy algorithm constructs a feasible timetable. The algorithm starts from an empty timetable and iteratively selects a lecture of a course (courses with a small number of available periods and a large number of unassigned lectures have priority) and assigns it a period and a room. As soon as a feasible initial assignment is reached, the adaptively combined intensification and diversification phases are used to reduce the number of soft constraint violations. The intensification phase consists of a tabu search algorithm that exploits two neighborhoods: one consists of exchanging the periods and rooms assigned to two lectures of different courses; the other one combines moves defined by interchanging two *Kempe chains* (see Lü and Hao 2010 for more details). When the search ends at a local optimum, the tabu search is restarted from this local optimum, but using the other neighborhood. This process is repeated until no improvement is possible. Iterated local search provides the diversification mechanism to guide the search to escape from the current local optimum. To destruct the reached local optimum solution, the algorithm uses a *penalty-guided perturbation operator* (see Lü and Hao 2009) based on the identification of a set of highly-penalized lectures and a random selection of a given number of neighborhood moves. The algorithm is tested on the instances of ITC-2007 (`comp01-comp21`) and classified in the second position at the ITC-2007 competition. The obtained results are presented in Tables 2 and 3 (see Sect. 4.3).

In Lü et al. (2011), an experimental analysis of neighborhood relations for local search algorithms is presented. The authors examine three neighborhoods from the literature and introduce a new one, called *Kempe swap*, which consists of interchanging the lectures of two distinct Kempe chains. They also consider neighborhood union (at each iteration the neighborhood structure includes all the moves of two different

Table 3 Comparison of the average solution values found by the heuristic methods (UD2)

Inst	WBest	Best	M09	LH10	AT12	TAA13	BCDSU14	K14
comp01	5*	5	**5.0**	**5.0**	**5.0**	**5.0**	5.3	**5.0**
comp02	24	24	61.3	60.6	36.4	42.2	54.7	**41.9**
comp03	64	66	94.8	86.6	74.4	81.3	80.7	**72.8**
comp04	35*	35	42.8	47.9	38.5	50.7	40.4	**35.2**
comp05	284	295	343.5	328.5	314.5	**301.3**	339.0	306.3
comp06	27*	27	56.8	69.9	**45.3**	56.7	53.7	48.1
comp07	6*	6	33.9	28.2	**12.0**	21.1	26.8	15.3
comp08	37*	37	46.5	51.4	40.8	52.4	44.5	**40.6**
comp09	96*	96	113.1	113.2	108.4	106.3	107.9	**102.4**
comp10	4*	4	21.3	38.0	**8.4**	11.1	22.9	13.3
comp11	0*	0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
comp12	298	306	351.6	365.0	320.3	**314.1**	340.1	323.9
comp13	59*	59	73.9	76.2	64.3	76.3	75.7	**63.8**
comp14	51*	51	61.8	62.9	64.4	64.6	58.6	**56.1**
comp15	64	66	94.8	87.8	**72.7**	75.5	80.0	73.8
comp16	18*	18	41.2	53.7	**23.7**	42.5	39.3	34.8
comp17	56*	56	86.6	100.5	76.4	**71.3**	79.9	73.0
comp18	61*	65	91.7	82.6	75.6	79.9	83.7	**66.5**
comp19	57*	57	68.8	75.0	66.8	70.7	67.1	**64.6**
comp20	4*	4	34.3	58.2	**13.5**	17.8	47.5	24.0
comp21	74*	86	108.0	125.3	100.7	**87.8**	104.7	95.3
No. of best			2	2	8	6	1	**11**
Avg gap			33.46 %	36.05 %	**20.51 %**	27.50 %	30.37 %	21.78 %

M09 Müller (2009)—average out of 10 runs, 1 CPU time unit (results taken from the competition website http://www.cs.qub.ac.uk/itc2007/);
LH10 Lü and Hao (2010)—average out of 100 runs, 1 CPU time unit—3.4 GHz PC;
AT12 Abdullah and Turabieh (2012)—average out of 10 runs, 1 CPU time unit (=600 s)—Intel Pentium 4 2.33 GHz;
TAA13 Tarawneh et al. (2013)—average out of 30 runs, 1 CPU time unit—2.1 GHz PC;
BCDSU14 Bellio et al. (2014)—average out of 10 runs, 1 CPU time unit—Intel Xeon E-2660 2.2 GHz;
K14 Kiefer et al. (2014)—average over 10 runs, 1 CPU time unit—Core i5-3550 3.30 GHz

neighborhoods) and token-ring search (different neighborhoods are consecutively used on the local optimum of the previous neighborhood until no improvement is possible). To evaluate the behaviors of the different neighborhoods and their combinations, a steepest descent algorithm and three metaheuristic algorithms, i.e., tabu search, iterated local search and adaptive tabu search, are used. Three evaluation criteria are introduced to characterize the search capability of a neighborhood: percentage of improving neighbors, improvement strength and search steps. The analysis shows that the Kempe swap neighborhood is more powerful than the other neighborhoods, and expresses the superiority of the token-ring search for a combination of neighborhoods. Results on the instances of ITC-2007 are reported (comp01–comp14). The obtained results are presented in Table 2 (see Sect. 4.3).

4.2.2 Simulated annealing algorithms

Geiger (2012) proposes a local search procedure based on *threshold accepting*, a simplified deterministic variant of simulated annealing. This technique is used to overcome local optima by a deterministic acceptance of inferior solutions up to a given threshold. A constructive phase tries to determine a feasible assignment of lectures to time periods: at each iteration, the most critical event, i.e., the lecture with the smallest number of time periods to which it may be assigned, is selected. Then, the assignment is performed taking into account how well the number of students of the lecture matches the capacity of the available rooms. A reactive procedure is applied that gives priority to the most difficult events found in the constructive phase. An iterative procedure is then applied: at each step, a number of randomly chosen events is unassigned; a reassignment randomized phase follows, giving to each event the same probability. Two criteria are considered for evaluating the timetables, i.e., the distance from feasibility and the total penalty due to the soft constraints violation. In case of identical distance from feasibility, solutions with a higher total penalty are accepted up to a given threshold. The algorithm is tested on the instances of ITC-2007 (`comp01-comp21`) and classified in the fourth position at the ITC-2007 competition.

In Bellio et al. (2014) (see also Bellio et al. 2013), a simulated annealing algorithm is proposed. Two neighborhoods are considered, i.e., the move of a lecture from a time period/room to another time period and/or another room (possibly to one that is empty in that time period), and the swap of time periods and rooms of two lectures of distinct courses. The algorithm uses a *swap rate parameter* to control how often the second neighborhood is selected with respect to the first one. The algorithm consists of a single-stage simulated annealing, which is enhanced by two features: a cutoff-based temperature cooling scheme and a stopping condition based on the maximum number of allowed iterations. The authors develop a deep statistical analysis and are able to determine a linear regression model between the instance features and the search method parameters, that allows to set the parameters for new instances on the basis of a simple inspection of the characteristics of the instance. The algorithm is trained on a large set of instances generated by the generator of Lopes and Smith-Miles (2010) and is validated on the instances of ITC-2007 (`comp01-comp21`). Finally, the algorithm is tested on new instances (`DDS1-DDS7`, `erlangen*`, `Udine*`, `EA*`). The obtained results are presented in Table 3 (see Sect. 4.3).

Tarawneh et al. (2013) present a hybrid simulated annealing algorithm, enhanced by storing unaccepted solutions and using them when trapped in a local optimum. An initial feasible solution is build with the greedy algorithm by Lü and Hao (2010). Then simulated annealing is applied. Three neighborhoods are considered: (i) moving one lecture from a time period to a free time period that does not cause any conflict, (ii) swapping two lectures without violating the hard constraints, and (iii) swapping the time period with the highest penalty with a randomly selected time period. The algorithm is tested on the instances of ITC-2007 (`comp01-comp21`) and on `test1-test4`, `DDS1-DDS7`. The obtained results are presented in Table 3 (see Sect. 4.3).

4.2.3 Hybrid algorithms

In Müller (2009), a hybrid approach consisting of three phases is presented. In the construction phase, a complete feasible solution is found using an *iterative forward search* that iteratively selects a lecture of a course and assigns it a room and a time period. If this causes any violations of hard constraints with existing assignments, the conflicting assignments are unassigned. The search ends when all the lectures are assigned a room and a time period. Once a complete solution is found, a *hill climbing* phase is applied to find a local optimum. Several neighborhoods are considered, corresponding to time/room/lecture moves as well as moves related to the soft constraints. Hill climbing is stopped after a given number of iterations during which a solution has not been improved. Once a solution can no longer be improved using this method, a *great deluge* phase is used, in which a bound is imposed on the value of the current solution, and the bound is iteratively decreased during the search according to a cooling rate. Optionally, simulated annealing can also be used. The search ends after a predetermined time limit has been reached. The algorithm was applied to the instances of the three tracks of ITC-2007 and was the winner of two tracks, including CB-CTT. The algorithm is tested on the ITC-2007 instances (`comp01–comp21`). The obtained results are presented in Sect. 4.3.

In Abdullah and Turabieh (2012), a tabu-based memetic algorithm that hybridises a genetic algorithm with a tabu search algorithm is proposed. An initial population is generated in a heuristic way. In this population, a chromosome represents a feasible solution. Two solutions are selected from this population, according to the roulette wheel selection procedure. Crossover and mutation are then employed: the crossover operation represents a period-exchange crossover that ensures that the feasibility of the offspring is maintained, and the mutation is used to allow diversification by moving/swapping some events from one time period to another one. A set of involved neighborhoods are then employed in an improvement procedure. A tabu list is embedded to control the selection of neighborhoods. The best solution found is added to the population and the process is iterated until a time limit is reached. Different ways of selecting the sequence of neighborhood explorations are investigated and the algorithm is tested on the instances of ITC-2007 (`comp01–comp21`). The obtained results are presented in Sect. 4.3. The algorithm is also tested on the examination timetabling problem.

In Bellio et al. (2012), a hybrid local search algorithm is presented, which combines simulated annealing with dynamic tabu search. The search space is the same as in Di Gaspero and Schaerf (2003). It takes into account the same neighborhoods used in Bellio et al. (2013). In the tabu search algorithm, the size of the tabu list is variable and all non-tabu neighbors are evaluated to select the best one. The tabu search algorithm is dynamic, i.e., it changes continuously the shape of the cost function in an adaptive way, by changing the weights of the constraints penalized in the cost function. The simulated annealing phase and the tabu search phase are combined in a token-ring search, that makes circularly a run of each algorithm, always starting from the best solution found by the previous one. A complex statistical analysis is developed. The hybrid algorithm is tested on the instances of ITC-2007 (`comp01–comp21`). The obtained results are presented in Sect. 4.3.

In Shaker et al. (2013), a combination of two metaheuristic algorithms is presented, namely great deluge and tabu search. A feasible timetable is constructed by means of a *largest degree heuristic*. An improvement algorithm is then applied, which combines *great deluge* and *tabu search*. Two types of neighborhoods are considered: the first one consists of randomly choosing a lecture and moving it to the feasible time period that gives the smallest penalty cost; the second one consists of randomly selecting two lectures in the same room and swapping their time periods. In the great deluge phase, if a better solution is found, it is updated. Otherwise, the solution cost is compared to a threshold level and the new solution is accepted only if its cost is below this level. In the tabu search phase, a tabu list is used to avoid repeating moves for a given number of iterations. The process is repeated until a time limit is reached. The algorithm is tested on the instances of ITC-2007 (`comp01-comp21`) and on `DDS1-DDS7`. A variant of the algorithm is tested on the post-enrolment course timetabling problem.

In Kiefer et al. (2014), an *adaptive large neighborhood search* algorithm is proposed. Starting from an initial feasible solution, at each iteration parts of the incumbent solution are destroyed and subsequently repaired to improve the solution. We refer the reader to Kiefer et al. (2014) for full details on the many different destroy and repair operators employed in the algorithm. Each operator is assigned a weight: initially, each operator has the same selection probability. At each iteration, the selection of destroy and repair operators is based on a roulette wheel mechanism. Each time the algorithm has performed a given number of iterations, the weights are recomputed, taking into account the old weight and the score, that depends on the solution quality achieved during these iterations (the scores are normalized by the computational effort). The destroy operators are responsible for releasing parts of the search space that are subsequently explored by a repair heuristic. In particular, at each iteration, a given number of lectures is removed from the current timetable and then reinserted through a repair heuristic, while the other lectures are kept fixed in the schedule. The number of lectures to be removed is randomly chosen in a range where the maximum value is decreased during the iterative process. The acceptance scheme for the obtained solutions is based on simulated annealing. In addition, the algorithm does not prohibit infeasible solutions, with the advantage of allowing for making shortcuts by traversing infeasible regions of the search space and reducing the computing time of the repair heuristic algorithm. A *large neighborhood search* algorithm is also presented, which is based on the information on the selection rates of the operators used during the adaptive large neighborhood search. In particular, the average selection rates, computed over ten runs for each instance, are used as input for the large neighborhood search. The latter has the advantage to use predefined operator selection probabilities from the very beginning. The algorithm is tested on the instances of ITC-2007 (`comp01-comp21`) and on `DDS1-DDS7`, `Udine*` and `EA*`. The results obtained by the large neighborhood search algorithm are presented in Sect. 4.3.

4.3 Computational results

In this section, we report the computational results obtained by the most effective heuristic algorithms described in Sect. 4. We consider `UD2` and the instances

comp01–comp21 used for the ITC-2007 competition. We present the results in two tables: in Table 2 we report, for each instance, the best solution value obtained by each algorithm, while in Table 3 we report the average solution values obtained over multiple runs of the algorithms, similarly to what was done for the ITC-2007 competition. If only the best solution values (the average solution values, resp.) are known for an algorithm, then it will appear only in Table 2 (Table 3, resp.).

In each table, we indicate in the first column the instance name (Inst), in the second column the best known solution (WBest) as taken from the website (http://satt.diegm.uniud.it/ctt),[3] and in the third column the best solution (Best) obtained by one of the considered heuristic algorithms. Each of the remaining columns corresponds to one of the algorithms (indicated in chronological order). In particular, we compare the following algorithms:

- M09: Hybrid algorithm (see Sect. 4.2.3) proposed by Müller (2009);
- BMPR10: ILP-based algorithm (see Sect. 4.1) proposed by Burke et al. (2010a);
- LH10: Tabu search algorithm (see Sect. 4.2.1) proposed by Lü and Hao (2010);
- LHG11: Tabu search algorithm (see Sect. 4.2.1) proposed by Lü et al. (2011);
- LL12: ILP-based algorithm (see Sect. 4.1) proposed by Lach and Lübbecke (2012);
- AT12: Hybrid algorithm (see Sect. 4.2.3) proposed by Abdullah and Turabieh (2012);
- BDS12: Hybrid algorithm (see Sect. 4.2.3) proposed by Bellio et al. (2012);
- BSTIS13: Constraint satisfaction algorithm (see Sect. 3.7) proposed by Banbara et al. (2013);
- AN14: Constraint satisfaction algorithm (see Sect. 3.6) proposed by Asín Achá and Nieuwenhuis (2014);
- K14: Hybrid algorithm (see Sect. 4.2.3) proposed by Kiefer et al. (2014);
- TAA13: Hybrid simulated annealing algorithm (see Sect. 4.2.2) proposed by Tarawneh et al. (2013);
- BCDSU14: Simulated annealing algorithm (see Sect. 4.2.2) proposed by Bellio et al. (2014).

At the bottom of each table, we indicate a legend with the reference corresponding to each algorithm, the time limit used by it (expressed in CPU time units when available) and the computer used. We identify with an '*' the optimal solution values and in bold the best values. In Tables 2 and 3 we report, for each method, the number of best solution values obtained (out of 14 or 21 instances) and the average percentage gaps from the best solution values obtained by one of the considered algorithms (computed on 14 or 21 instances). In addition, in Table 2 we report, for each method, the number of instances for which the considered algorithm is the only one able to determine the best solution value (out of 14 or 21 instances).

We wish to mention that it is difficult to establish an overall "best" algorithm, since not all the algorithms are run for the same computing time, and also because more than one algorithm turn out to be effective. We can observe that metaheuristic algorithms and algorithms based on constraint satisfaction usually have a better performance than those based on mathematical models. In particular, the method proposed by

[3] Visualized on 9 October 2014.

Asín Achá and Nieuwenhuis (2014) obtains many of the best solution values and for many instances is the only algorithm able to determine the best solution value, while that proposed by Abdullah and Turabieh (2012) is the one reaching the best average percentage gap. We can also see that there is still room for improvement of the results.

5 Extensions of the problem

As mentioned in Sect. 2, some simplifications have been introduced to keep a certain level of generality in the definition of CB-CTT. In this section, we present some variants and extensions of CB-CTT. First of all, we describe the extensions proposed in Bonutti et al. (2012). In addition or in replacement of (some of) the soft constraints described in Sect. 2, other cost components can be taken into account:

- *windows*: this is an alternative way to take into account curriculum compactness: lectures of a curriculum should not have time windows without teaching between them; for a given curriculum, a violation is considered every time there is an empty window between two lectures of the curriculum in the same day;
- *student min/max load*: for a given curriculum, the number of daily lectures that a student must attend should be within a given range;
- *travel distance*: students should have the time to move from one building to another one between two lectures;
- *room suitability*: some rooms may be not suitable for a given course because of the absence of necessary equipment (this is modeled as a hard constraint in `UD4`);
- *double lectures*: some courses require that lectures in the same day are grouped together, i.e., they should be scheduled in adjacent time periods and in the same room.

These additional features appear in the variants of CB-CTT defined as `UD3`, `UD4` and `UD5`. Such alternative variants of CB-CTT have received in the literature (Bellio et al. 2012; Lach and Lübbecke 2012; Banbara et al. 2013; Phillips et al. 2015) less attention than UD2 even if they deal with important aspects of the problem.

In Bonutti et al. (2012), the authors underline that the real-world problems can contain many other features, such as *preassignment* of a lecture to a specific room or time period, *lunch break*, i.e., leaving an empty time period around lunch time, *teacher min/max load*, i.e., imposing bounds on the daily teaching activity for each teacher, *day patterns* that specify specific patterns for some course (e.g. lectures in the morning and laboratory exercises in the afternoon), etc. We refer the reader to Bonutti et al. (2012) for a complete overview of the proposed variants.

As shown for example in Avella and Vasil'ev (2005), Daskalaki and Birbas (2005), Schimmelpfeng and Helber (2007) and Lach and Lübbecke (2012), preferences can be given by teachers to time periods (or even more complex types of preferences as in Schimmelpfeng and Helber (2007)) and this can become one of the objectives or even the main goal to be optimized.

Multi-objective optimization also plays a role in this context (see e.g. Landa-Silva et al. 2004). Often conflicting objectives appear in university timetabling, taking into account the needs of the students as well as those of the faculties. A common approach is to introduce soft constraints and penalize their violations in a weighted cost objective

function to be minimized. In Geiger (2009), a *weighted sum* method and a *reference point*-based approach are compared. An interesting alternative is to tackle the problem as a multi-objective one and determine the entire Pareto front (see Phillips et al. 2015).

In Mühlenthaler and Wanka (2014), fairness is taken into account in the determination of university course timetables. In this context, fairness requires that the penalties due to the violation of the soft constraints assigned to a timetable are distributed in a fair way among the different curricula. The authors consider two types of fairness. The first one is *Max–min* fairness that aims at producing an optimal outcome for the worst-off stakeholder and, under this condition, aims at producing an optimal outcome for the second worst-off stakeholder, and so on for all the remaining stakeholders, considered in lexicographic order. The second one is *Jain's fairness index* (Jain et al. 1984) that corresponds to a bi-criteria optimization problem, taking into account fairness and efficiency.

In Beliën and Mercy (2013), the student flows minimization related to the determination of university timetables is considered. Queues and congestion problems in stair halls and elevators are often caused by students that move from a room to another one between two consecutive lectures. The authors propose to take into account student flows when constructing the course timetables, e.g. timetables in which consecutive lectures are scheduled in the same room or in rooms situated on the same floor are to be preferred.

6 Conclusions and open perspectives

We have presented a review on a very active research topic, i.e., the curriculum-based course timetabling problem. As it is evident from the review, an important contribution to this area has been the organization of the International Timetabling Competitions, which have been a successful way to attract more researchers to work on the topic, and have provided benchmark instances coming from real-world problems. One of the aims of this paper is to increase the interest on this topic and open new perspectives of further developments.

Besides the interesting approaches described in this paper, we have also presented some extensions and variants of the problem that could be investigated as future research, some of which have recently been studied. In particular, the aspect of fairness between different curricula, but even between different teachers, seems to be very relevant in this context. It could be preferable to have a solution with a higher global penalty but which spreads the penalties over all the curricula instead of having a few curricula with a "bad" schedule.

Some of the soft constraints, such as room capacity or travel distance, could be dealt with as hard ones. Often room capacity cannot be considered as a hard constraint because this would lead to infeasible solutions. However, some universities, especially the new ones, should have enough seats for the students and in this case, it would be natural to have room capacity as a hard constraint. Travel distance might also become a hard constraint: some universities are characterized by several buildings, which can

be placed far away from each other and it is therefore necessary to give the students and/or the teachers enough time to move from one place to another one.

If we look at the methodologies proposed for CB-CTT, we can see that there are many complex algorithms and models that produce good quality results. We can also observe that the problem is still hard to be solved to optimality and only few methods have been proposed for determining its optimal solution. The recent availability of affordable general purpose GPUs has increased the development of parallel processing methods (see e.g. Bożejko et al. 2014; Kolonias et al. 2014 in the educational timetabling context). This is an interesting research area, that can improve the effectiveness of exact methods. In addition, new benchmark instances (`erlangen*`, `Udine*` and `EA*` described in Sect. 2.1) have recently been uploaded on the website http://satt.diegm.uniud.it/ctt. This should also stimulate the research in solving these challenging problems.

Another challenge is to keep the studied problems as close to real-world practical problems as possible. Indeed, standardization of timetabling benchmarks might lead to problems that are far from practical applications. Therefore, there must be a direct connection between academic problems/methods and practical needs. Furthermore, in a real-world environment, it is not only important to develop an effective method that obtains good or even optimal timetables, but also to have a whole automated framework that takes into account data integrity, graphical user interface, security, access level, etc.

We have noticed that we did not find a "winner" method able to outperform all the other ones in all the benchmark instances. Therefore, it is hard to say which methods should be investigated or could be more effective in different situations. Certainly, it would be interesting to determine why a method fails or succeeds in different sets of instances with specific features. This would also be useful to practitioners, who can then appropriately choose the method to be applied. We leave this as a future challenge to be investigated.

Acknowledgments We would like to thank the ITC-2002 and ITC-2007 organizers for providing the benchmark instances and the formal description of the CB-CTT, as well as Alex Bonutti, Luca Di Gaspero and Andrea Schaerf who maintain the website. We would like to thank Miguel A. Goberna, the editor of TOP, for the invitation to write this paper. We would also like to thank Roberto Asín, Edmund K. Burke, John H. Drake, Marco Lübbecke, Barry McCollum, Ender Özcan and Andrea Schaerf for their insightful comments on the first version of the paper.

References

Abdullah S, Turabieh H (2012) On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. Inf Sci 191:146–168

Abdullah S, Burke EK, McCollum B (2007) Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In: Doerner K, Gendreau M, Greistorfer P, Gutjahr W, Hartl R, Reimann M (eds) Metaheuristics, operations research/computer science interfaces series, vol 39. Springer, US, pp 153–169

Al-Yakoob S, Sherali H (2007) A mixed-integer programming approach to a class timetabling problem: a case study with gender policies and traffic considerations. Eur J Oper Res 180(3):1028–1044

Asín Achá R, Nieuwenhuis R (2014) Curriculum-based course timetabling with SAT and MaxSAT. Ann Oper Res 218(1):71–91

Atsuta M, Nonobe K, Ibaraki T (2008) Itc 2007 track 2, an approach using general csp solver. Technical report, www.cs.qub.ac.uk/itc2007
Avella P, Vasilev I (2005) A computational study of a cutting plane algorithm for university course timetabling. J Sched 8(6):497–514
Babaei H, Karimpour J, Hadidi A (2014) A survey of approaches for university course timetabling problem. Comput Ind Eng. doi:10.1016/j.cie.2014.11.010
Banbara M, Soh T, Tamura N, Inoue K, Schaub T (2013) Answer set programming as a modeling language for course timetabling. Theory Pract Log Program 13(4–5):783–798
Beliën J, Mercy A (2013) Building university course timetables with minimized resulting student flows. In: Proceedings of the 6th multidisciplinary international conference on scheduling: theory and applications (MISTA 2013), Belgium, pp 737–740
Bellio R, Di Gaspero L, Schaerf A (2012) Design and statistical analysis of a hybrid local search algorithm for course timetabling. J Sched 15(1):49–61
Bellio R, Ceschia S, Di Gaspero L, Schaerf A, Urli T (2013) A simulated annealing approach to the curriculum-based course timetabling problem. In: Proceedings of the 6th multidisciplinary international conference on scheduling: theory and applications (MISTA 2013), Belgium, pp 314–317
Bellio R, Ceschia S, Di Gaspero L, Schaerf A, Urli T (2014) Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. arXiv:1409.7186
Bonutti A, De Cesco F, Di Gaspero L, Schaerf A (2012) Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. Ann Oper Res 194(1):59–70
Bożejko W, Gniewkowski Ł, Wodecki M (2014) Solving timetabling problems on gpu. In: Artificial intelligence and soft computing, Springer, pp 445–455
Burke E, Petrovic S (2002) Recent research directions in automated timetabling. Eur J Oper Res 140(2):266–280
Burke E, Jackson K, Kingston JH, Weare R (1997) Automated university timetabling: the state of the art. Comput J 40(9):565–571
Burke E, Mareček J, Parkes A, Rudová H (2008) Penalising patterns in timetables: novel integer programming formulations. Oper Res Proc 2007:409–414
Burke E, Mareček J, Parkes A, Rudová H (2010a) Decomposition, reformulation, and diving in university course timetabling. Comput Oper Res 37(3):582–597
Burke E, Mareček J, Parkes A, Rudová H (2010b) A supernodal formulation of vertex colouring with applications in course timetabling. Ann Oper Res 179(1):105–130
Burke E, Mareček J, Parkes A, Rudová H (2012) A branch-and-cut procedure for the Udine course timetabling problem. Ann Oper Res 194(1):71–87
Cacchiani V, Caprara A, Roberti R, Toth P (2013) A new lower bound for curriculum-based course timetabling. Comput Oper Res 40(10):2466–2477
Carter M (2001) A comprehensive course timetabling and student scheduling system at the university of waterloo. In: Burke E, Erben W (eds) Practice and theory of automated timetabling III, vol 2079., Lecture notes in computer scienceSpringer, Berlin, pp 64–82
Carter M (2013) Timetabling. In: Gass S, Fu M (eds) Encyclopedia of operations research and management science. Springer, US, pp 1552–1556
Chiarandini M, Birattari M, Socha K, Rossi-Doria O (2006) An effective hybrid algorithm for university course timetabling. J Sched 9(5):403–432
Clark M, Henz M, Love B (2009) Quikfix a repair-based timetable solver. In: Proceedings of the 7th international conference on the practice and theory of automated timetabling (PATAT-2008)
Daskalaki S, Birbas T (2005) Efficient solutions for a university timetabling problem through integer programming. Eur J Oper Res 160(1):106–120
Daskalaki S, Birbas T, Housos E (2004) An integer programming formulation for a case study in university timetabling. Eur J Oper Res 153(1):117–135
Di Gaspero L, Schaerf A (2003) Multi-neighbourhood local search with application to course timetabling. In: Burke E, De Causmaecker P (eds) Practice and theory of automated timetabling IV, vol 2740., Lecture notes in computer scienceSpringer, Berlin, pp 262–275
Di Gaspero L, Schaerf A (2006) Neighborhood portfolio approach for local search applied to timetabling problems. J Math Model Algorithms 5(1):65–89
Di Gaspero L, McCollum B, Schaerf A (2007) The second international timetabling competition (itc-2007): curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical

Engineering and Computer Science, Queens University, Belfast (UK), ITC-2007. site: http://www.cs.qub.ac.uk/itc2007/

Geiger MJ (2009) Multi-criteria curriculum-based course timetablinga comparison of a weighted sum and a reference point based approach. In: Ehrgott M, Fonseca CM, Gandibleux X, Hao JK, Sevaux M (eds) In: Proceedings of the 5th international conference on evolutionary multi-criterion optimization, EMO 2009, Springer, Lecture notes in computer science, vol 5467, pp 290–304

Geiger MJ (2012) Applying the threshold accepting metaheuristic to curriculum based course timetabling. Ann Oper Res 194(1):189–202

Hansen P, Hertz A, Kuplinsky J (1993) Bounded vertex colorings of graphs. Discret Math 111(13):305–312

Hao JK, Benlic U (2011) Lower bounds for the ITC-2007 curriculum-based course timetabling problem. Eur J Oper Res 212(3):464–472

Jain R, Chiu DM, Hawe WR (1984) A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Technical report DEC-TR-301, Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA

Kiefer A, Hartl R, Schnell A (2014) Adaptive large neighborhood search for the curriculum-based course timetabling problem. Technical report UNIVIE-PLIS-2014-001, University of Vienna

Kingston JH (2013) Educational timetabling. In: Uyar AS, Ozcan E, Urquhart N (eds) Automated scheduling and planning, studies in computational intelligence, vol 505. Springer, Berlin, pp 91–108

Kolonias V, Goulas G, Gogos C, Alefragis P, Housos E (2014) Solving the examination timetabling problem in gpus. Algorithms 7(3):295–327

Kostuch P (2005) The university course timetabling problem with a three-phase approach. In: Burke E, Trick M (eds) Practice and theory of automated timetabling V, vol 3616., Lecture notes in computer scienceSpringer, Berlin, pp 109–125

Kristiansen S, Stidsen T (2013) A comprehensive study of educational timetabling—a survey. Technical report, DTU Management Engineering

Lach G, Lübbecke M (2008) Optimal university course timetables and the partial transversal polytope. In: McGeoch C (ed) Experimental algorithms, vol 5038., Lecture notes in computer scienceSpringer, Berlin, pp 235–248

Lach G, Lübbecke M (2012) Curriculum based course timetabling: new solutions to Udine benchmark instances. Ann Oper Res 194(1):255–272

Landa-Silva D, Obit JH (2008) Great deluge with non-linear decay rate for solving course timetabling problems. In: Intelligent systems, 2008. IS'08. In: 4th international IEEE conference, IEEE, vol 1, pp 8–11

Landa-Silva J, Burke E, Petrovic S (2004) An introduction to multiobjective metaheuristics for scheduling and timetabling. In: Metaheuristics for multiobjective optimisation. Springer, pp 91–129

Lewis R (2008) A survey of metaheuristic-based techniques for university timetabling problems. OR Spectr 30(1):167–190

Lewis R, Paechter B, Rossi-Doria O (2007) Metaheuristics for university course timetabling. Stud Comput Intell 49(49):237–272

Lopes L, Smith-Miles K (2010) Pitfalls in instance generation for udine timetabling. In: Blum C, Battiti R (eds) Learning and intelligent optimization, vol 6073., Lecture notes in computer scienceSpringer, Berlin, pp 299–302

Lopes L, Smith-Miles K (2013) Generating applicable synthetic instances for branch problems. Oper Res 61(3):563–577

Lü Z, Hao JK (2009) A critical element-guided perturbation strategy for iterated local search. In: Cotta C, Cowling P (eds) Evolutionary computation in combinatorial optimization, vol 5482., Lecture notes in computer scienceSpringer, Berlin, pp 1–12

Lü Z, Hao JK (2010) Adaptive tabu search for course timetabling. Eur J Oper Res 200(1):235–244

Lü Z, Hao JK, Glover F (2011) Neighborhood analysis: a case study on curriculum-based course timetabling. J Heuristics 17(2):97–118

McCollum B (2007) A perspective on bridging the gap between theory and practice in university timetabling. In: Burke E, Rudov H (eds) Practice and theory of automated timetabling VI, vol 3867., Lecture notes in computer scienceSpringer, Berlin, pp 3–23

McCollum B, Ireland N (2006) University timetabling: bridging the gap between research and practice. In: Proceedings of the 5th international conference on the practice and theory of automated timetabling, pp 15–35

McCollum B, Schaerf A, Paechter B, McMullan P, Lewis R, Parkes A, Di Gaspero L, Qu R, Burke E (2010) Setting the research agenda in automated timetabling: the second international timetabling competition. INFORMS J Comput 22(1):120–130

Miranda J (2010) eClasSkeduler: a course scheduling system for the executive education unit at the Universidad de Chile. Interfaces 40(3):196–207

MirHassani S (2006) A computational approach to enhancing course timetabling with integer programming. Appl Math Comput 175(1):814–822

MirHassani S, Habibi F (2013) Solution approaches to the course timetabling problem. Artif Intel Rev 39(2):133–149

Mühlenthaler M, Wanka R (2014) Fairness in academic course timetabling. Ann Oper Res 1–18

Müller T (2009) Itc 2007 solver description: a hybrid approach. Ann Oper Res 172(1):429–446

Müller T, Murray K (2010) Comprehensive approach to student sectioning. Ann Oper Res 181(1):249–269

Petrovic S, Burke E (2004) University timetabling. In: Leung JYT (ed) Handbook of scheduling: algorithms, models, and performance analysis, CRC Press, Boca Raton

Phillips AE, Waterer H, Ehrgott M, Ryan DM (2015) Integer programming methods for large-scale practical classroom assignment problems. Comput Oper Res 53:42–53

Pillay N (2014) A review of hyper-heuristics for educational timetabling. Ann Oper Res 1–36. doi:10.1007/s10479-014-1688-1

Qualizza A, Serafini P (2005) A column generation scheme for faculty timetabling. In: Burke E, Trick M (eds) Practice and theory of automated timetabling V, vol 3616., Lecture notes in computer scienceSpringer, Berlin, pp 161–173

Schaerf A (1999) A survey of automated timetabling. Artif Intel Rev 13(2):87–127

Schimmelpfeng A, Helber S (2007) Application of a real-world university-course timetabling model solved by integer programming. OR Spectr 29(4):783–803

Shaker K, Abdullah S, Alqudsi A, Jalab H (2013) Hybridizing meta-heuristics approaches for solving university course timetabling problems. In: Lingras P, Wolski M, Cornelis C, Mitra S, Wasilewski P (eds) Rough sets and knowledge technology, vol 8171., Lecture notes in computer scienceSpringer, Berlin, pp 374–384

Tarawneh HY, Ayob M, Ahmad Z (2013) A hybrid simulated annealing with solutions memory for curriculum-based course timetabling problem. J Appl Sci 13:262–269

Van Den Broek J, Hurkens C, Woeginger G (2009) Timetabling problems at the TU Eindhoven. Eur J Oper Res 196(3):877–885

Wren A (1996) Scheduling, timetabling and rostering a special relationship? In: Burke E, Ross P (eds) Practice and theory of automated timetabling, vol 1153., Lecture notes in computer scienceSpringer, Berlin, pp 46–75