Lower Bounds for the ITC-2007 Curriculum-Based Course Timetabling Problem

Jin-Kao Hao and Una Benlic

LERIA, Université d'Angers
2 Boulevard Lavoisier, 49045 Angers Cedex 01, France

To appear in *European Journal of Operational Research*
Doi:10.1016/j.ejor.2011.02.019

Abstract

This paper describes an approach for generating lower bound s for the Curriculum-based course timetabling problem, which was presented at th e International Timetabling Competition (ITC-2007, Track 3). So far, several methods bas ed on integer linear programming have been proposed for computing lower bounds o f this minimization problem. We present a new partition-based approach that is ba sed on the "divide and conquer" principle. The proposed approach uses Iterati ve Tabu Search to partition the initial problem into sub-problems which are solve d with an ILP solver. Computational outcomes show that this approach is able to im prove on the current best lower bounds for 12 out of the 21 benchmark instances, an d to prove optimality for 6 of them. These new lower bounds are useful to estimat e the quality of the upper bounds obtained with various heuristic approaches.

Keywords: bounds, partitioning, tabu search, timetabling

1 Introduction

The Curriculum-based course timetabling (CBCT) problem is a multiple constraint scheduling problem which arises regularly at every university. It consists of assigning a number of tuples, each comprising a class of students, a

Email addresses: hao@info.univ-angers.fr (Jin-Kao Hao), benlic@info.univ-angers.fr (Una Benlic).

teacher, a course and a room, to a limited number of periods. These tuples may be scheduled provided that a set of constraints is satisfied. Constraints are generally classified as hard or soft. Hard constraints cannot be violated under any condition. If at least one of these constraints is violated, the solution is infeasible. Soft constraints are used to define the objective function to be minimized. Examples of recent studies on university timetabling can be found in [1,2,4,5,6,10,12,14,16,18].

The CBCT problem considered in this paper constitutes the topic of Track 3 [7] of the International Timetabling Competition (ITC-2007) [15]. The ITC-2007 has led to the development of many heuristic solution approaches. The five top performing approaches (finalists) of the ITC-2007 include a hybrid approach [16], a tabu search [14], a general CSP solver [2,17], a threshold accepting approach [10], and a repair-based heuristic approach [6]. These heuristic approaches produce only upper bounds to this minimization problem. Lower bounds are needed to assess the quality of these solutions.

Until now, only a few studies have been devoted to determining lower bounds for the CBCT problem. In [12], Lach and Lübbecke proposed an integer programming formulation that gives satisfactory lower and upper bounds in reasonable computational time without particular tuning. Solving their model using CPLEX11 on a 3.4GHz Linux PC with 1GB memory within a time limit of about 4 hours, they obtain lower bounds with an average gap to the best existing upper bounds of 32.63% over the first 14 benchmark instances. In [4], Burke *et al.* devised a problem specific branch-and-cut procedure using CPLEX11 that reaches globally better bounds (average gap of 26.65%) within a time limit of 8 hours on an Intel Pentium 4 processor clocked at 3.20GHz with 4GB of RAM.

In this paper, we develop a partition-based approach, inspired by the work of Vasquez and Hao [19], which follows the well-known "divide and conquer" principle. We divide the initial (large) timetabling problem into a number of smaller sub-problems. Then, we formulate these sub-problems as integer linear programs, and solve them via an ILP solver (in our case, the non-commercial COIN-OR Cbc–2.2.2 solver is used). The sum of lower bounds of the sub-problems corresponds then to a lower bound of the initial problem. For each sub-problem, we adopt the Lach and Lübbecke's IP formulation [12]. To be effective, this partition-based approach needs to optimize the way the initial problem is divided. For this purpose, we devise an effective Iterated Tabu Search procedure which seeks the best partition possible.

The proposed approach is assessed on the set of benchmark instances from the ITC-2007. The computational outcomes show that this approach is able to improve on the current best lower bounds for 12 out of 21 benchmark instances and to prove optimality for 6 of them (with new optimality for 1

instance), reducing the average gap to the best-known upper bounds from 35.03% to 22.37%.

The paper is organized as follows. The CBCT problem description is introduced in Section 2, together with the ITC–2007 benchmark instances and the best known bounds. In Section 3, we briefly introduce the IP model presented in [12], which we used in our partition-based approach. The partition-based approach is described in Section 4. Finally in Section 5, we present the improved lower bounds obtained with the proposed method.

2 Problem Description of the ITC-2007 Track 3

2.1 Constraints

The CBCT problem from the ITC–2007 [7] consists in weekly scheduling a set of m courses $C = \{c_1, c_2, ..., c_m\}$, within a set of n rooms $R = \{r_1, r_2, ..., r_n\}$ and a set of h periods $P = \{p_1, p_2, ..., p_h\}$, in accordance with a given set of constraints. Each course c consists of a given number of lectures l_i and is associated to a teacher. A period p is a pair composed of a day and a timeslot. Thus, the total number of scheduling periods is the product of the number of days and the number of timeslots per day. In addition, there is a set of q curricula $CU = \{cu_1, cu_2, ..., cu_q\}$ where each curriculum cu_i corresponds to a group of courses such that any pair of courses in the group have students in common.

A timetabling solution is feasible if all lectures are scheduled such that no *hard constraint* of types $H_1 - H_4$, given below, is violated.

H_1 **Lectures:** All lectures of a course must be scheduled in distinct periods.

H_2 **Room Occupancy:** Two lectures cannot be assigned in the same room at the same time.

H_3 **Conflict:** All lectures belonging to the same curriculum or taught by the same teacher must be scheduled in distinct periods.

H_4 **Availability:** If a teacher is not available to give a lecture at a certain time, then the lecture has to be scheduled in another period.

The quality of a feasible solution depends on the satisfaction of four types of *soft constraints*. If a soft constraint is violated by a solution, a penalty β (a positive integer) is induced. These soft constraints $S_1 - S_4$, as well as the

associated penalties for violating these constraints are as follows:

S_1 **Room capacity:** For every lecture, the number of students attending the lecture should be less than or equal to the number of classroom seats. Then, the penalty β_1 for violating constraint S_1 is the number of students left without a seat at a lecture l_i, summed across every lecture l_i for each $c_i \in C$ when this value is positive.

S_2 **Minimum Working Days**: Lectures of a course should be spread over a specified number of distinct working days. The penalty β_2 for violating constraint S_2 sums the value of the specified number of distinct working days d_{ci}, minus the actual number of distinct working days across each $c_i \in C$ where this value is positive.

S_3 **Curriculum Compactness:** Lectures belonging to the same curriculum should be scheduled in consecutive periods. The penalty β_3 for violating constraint S_3 sums the number of isolated lectures l_i in each curriculum cu_i in CU.

S_4 **Room Stability:** All lectures of a course should take place in the same classroom. The penalty β_4 for violating constraint S_4 sums over each $c_i \in C$ the number of distinct course-room allocations above a single class-room allocation per course.

2.2 Objective

The optimization objective of the CBCT problem is to find a feasible timetable satisfying constraints $H_1 - H_4$ while minimizing a weighted sum of all the penalties of the unsatisfied soft constraints $S_1 - S_4$.

More formally, let Ω be the set of all feasible solutions (timetables). For each $x \in \Omega$, its cost is defined by:

$$f(x) = \sum_{i=1}^{4} \alpha_i \cdot \beta_i(x)$$

where $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (1, 5, 2, 1)$ identifies the weights corresponding to soft constraints $S_1 - S4$. These α weights are part of the problem definition, and imposed by the ITC-2007 competition organizers. They correspond to the decision-maker's preference.

Then the *goal* of the ITC-2007 CBCT problem is to find a feasible solution $x^* \in \Omega$ such that for all $x \in \Omega$, $f(x^*) \leq f(x)$.

Table 1
Main characteristics of the 21 ITC–2007 CBCT problem instances together with the best known bounds from [3].

Instance	#Rooms	#Periods	#Courses	#Events	#Curricula	Best UB	Best LB
comp01	6	30	30	160	14	**5**	5*
comp02	16	25	82	283	70	24	10
comp03	16	25	72	251	68	66	38
comp04	18	25	79	286	57	**35**	**35**
comp05	9	36	54	152	139	291	114*
comp06	18	25	108	361	70	27	16*
comp07	20	25	131	434	77	**6**	**6**
comp08	18	25	86	324	61	**37**	**37**
comp09	18	25	76	279	75	96	66*
comp10	18	25	115	370	67	**4**	**4**
comp11	5	46	30	162	13	**0**	**0**
comp12	11	36	88	218	150	300	53
comp13	19	25	82	308	66	59	48
comp14	17	25	85	275	60	**51**	**51**
comp15	16	25	72	251	68	66	41
comp16	20	25	108	366	71	18	13
comp17	17	25	99	339	70	56	44
comp18	9	36	47	138	52	62	0
comp19	16	25	74	277	66	57	49
comp20	19	25	121	390	78	4	0
comp21	18	25	94	327	78	83	0

2.3 The Udine Benchmark Instances

The ITC–2007 presented a total of 21 benchmark instances. Seven of them (comp01-07) were provided at the outset of the competition, seven more instances (comp08-14) were made known just before the submission deadline for participating in the competition. The last seven (comp15-21), also called "hidden instances", were known only by the competition organizers and were used as additional instances to benchmark the competing algorithms.

Table 1 summarizes the main characteristics of these 21 benchmark instances. The last two columns list the best known upper and lower bounds available at the CBCT website [3], which is maintained by the ITC-2007 competition organizers. The bounds from Table 1 are retrieved on November 2010 from this website. In addition, the last column integrates 4 recently improved lower bounds from [4] next to which we include a (*) symbol. The optimal solutions (for a total of 7) are given in bold.

3 ITC-2007 Lower Bounds: State of the Art

To the best of our knowledge, only two recently published studies are devoted to the calculation of lower bounds for the CBCT problem ([4],[12]). These studies are based on different integer programming formulations. In [4], the authors additionally propose, together with an IP formulation, a new problem specific branch and cut procedure, where an exponential number of cuts has to be considered to ensure optimality, but the separation can be achieved in reasonable time. This approach has obtained very good lower bounds. We do not provide details regarding this procedure since our partition-based approach is not based on it.

In our work, instead of developing a new model, we decided to use the well tested Lach & Lübbecke IP formulation [12]. This model is relatively simple and gives satisfactory upper and lower bounds in reasonable time without special tuning. In the next subsection, we briefly describe the underlying IP formulation.

3.1 Lach and Lübbecke's IP Model

In [12], Lach and Lübbecke reduce the problem in three dimensions (i.e. course, period, room) to a problem in two dimensions. Instead of directly solving the natural formulation with binary variables that indicate whether course c is scheduled for period p in room r, they split the problem into two stages. In the first stage they only match courses with periods. In the second stage rooms are feasibly assigned to these course/period pairs. All constraints, except for Room Stability (S_4), are taken care of during the first stage of the algorithm.

Unfortunately, it is impossible to integrate the S_4 constraints into the first stage with the proposed formulation. Since the S_4 constraints need to go in the second stage, we might not obtain a globally optimal solution even when both stages are solved to optimality. Therefore, in our approach, we only use the formulation of the first stage to compute the lower bounds. Nevertheless, the solution quality did not substantially decrease since the S_4 constraints are the least important soft constraints according to the the weight given to these constraints.

We give below a brief explanation of the Lach and Lübbecke's model. More details can be found in [12].

3.1.1 IP Formulation of the First Stage [12]

The complete input of this formulation is captured by the following variables and their mappings:

- C, CU, R, S, D, P, T are respectively the set of courses, curricula, rooms, distinct room capacities, days, periods, and teachers.
- $P(c)$ is the subset of periods P that are eligible for course c.
- $C(t)$ is the subset of courses C taught by teacher t.
- $C_{\geq s}$ is the subset of courses C that require more than s seats.
- $R_{\geq s}$ is the subset of rooms R with capacity of more than s seats.
- $l(c)$ is the number of lectures of course c.
- $mnd(c)$ is the prescribed minimum number of different days during which course c should be taught.
- x are binary decision variables indexed with courses and periods. For each pair course/period, we include a variable x only if teacher is available to give a lecture of course c at period p. In this way, we ensure that no Availability (H_4) constraint is violated. A course c should be taught at p only if $x_{c,p}$ is set to one.

Apart from the "core" decision variables x, there is also a group of five dependent variables y, z, w, r and v, whose values are derived from the value of x in the solution process, which seeks to minimize the following linear combination of penalty terms:

$$\sum_{p\in P, s\in S, c\in C_{\geq s}} obj_{s,c,p} \cdot y_{s,c,p} + \sum_{c\in C} 5 \cdot w_c + \sum_{cu\in CU, p\in P} 2 \cdot v_{cu,p}$$

The first term of the objective function (equivalent to penalty β_1 of S_1 defined in Section 2.2) sums, over every lecture, the number of students left without a seat. If the number of students attending a lecture of course c at period p is greater than the capacity of the allocated room, the value $obj_{s,c,p}$ is the difference between the number of students attending the lecture and the capacity of the allocated room.

The second term $5 \cdot w_c$ (equivalent to penalty β_2 of S_2 defined in Section 2.2) sums over all courses the difference between the prescribed number of distinct days of instruction and the actual number of distinct days of instruction if this difference is positive. Coefficient 5 is the weight α_2 (see Section 2.2) for violating a minimum working day constraint.

The third term of the objective function (equivalent to penalty β_3 of S_3 defined in Section 2.2) sums the number of isolated lectures of each curriculum, where 2 is the weight α_3 (see Section 2.2) for violating a curriculum compactness constraint.

$$min \sum_{p\in P, s\in S, c\in C_{\geq s}} obj_{s,c,p} \cdot y_{s,c,p} + \sum_{c\in C} 5 \cdot w_c + \sum_{cu\in CU, p\in P} 2 \cdot v_{cu,p}$$

subject to

$$\sum_{p\in P} x_{c,p} = l(c) \qquad \forall c \in C \tag{1}$$

$$\sum_{c\in C} x_{c,p} \leq \mid R \mid \qquad \forall p \in P \tag{2}$$

$$x_{c,p} - y_{s,c,p} \geq 0 \qquad \forall s \in S, c \in C_{\geq s}, p \in P \tag{3}$$

$$\sum_{c\in C_{\geq s}} x_{c,p} - y_{s,c,p} \leq \mid R_{\geq s} \mid \qquad \forall s \in S, p \in P \tag{4}$$

$$\sum_{p\in d} x_{c,p} - z_{c,d} \geq 0 \qquad \forall c \in C, d \in D \tag{5}$$

$$\sum_{d\in D} z_{c,d} + w_c \geq mnd(c) \qquad \forall c \in C \tag{6}$$

$$\sum_{c\in cu} x_{c,p} - r_{cu,p} = 0 \qquad \forall cu \in CU, p \in P \tag{7}$$

$$- r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} \leq 0 \qquad \forall cu \in CU, p \in P \tag{8}$$

$$\sum_{c\in C(t)} x_{c,p} \leq 1 \qquad \forall t \in T, p \in P \tag{9}$$

$$x_{cp} \in \{0,1\}$$
$$y_{s,c,p} \in \{0,1\}$$
$$z_{c,d} \in \{0,1\}$$
$$w_c \in \mathbb{Z}_+$$
$$r_{cu,p} \in \{0,1\}$$
$$v_{cu,p} \in \{0,1\}$$

Fig. 1. IP formulation of the First Stage [12]

Figure 1 gives the complete formulation of the first stage. The equation (1) satisfies the Lectures (H_1) constraints, which require that the number of matched course/period pairs of course c must be equal to the number of lectures of c.

The following three constraints (2–4) are related to the Room Capacity constraint. In addition, these constraints ensure that no Room Occupancy (H_2) constraint is ever violated. The first of these requires the number of courses that can take place at p to be at most the number of available rooms. Constraints (3) and (4) take into consideration the different room capacities and

number of seats needed for each course. For each $s \in S$, except the smallest, and for all $c \in C_{\geq s}$ there is a binary variable $y_{s,c,p}$ which is set equal to 1 if course c takes place in a room with less than s seats. Constraint (4) ensures that this does not happen for more courses than the number of rooms that have suitable capacity.

Constraints (5) and (6) are the Minimum Working Day (S_2) constraints. For every course and every feasible day of that course, we add a binary variable $z_{c,d}$ which is set equal to 1 if course c takes place during some period of day d. In addition, we add another integer variable w_c which is set equal to 0 only if course c is spread across more than *mnd(c)–1* days.

The following constraints (7) and (8) are the Curriculum Compactness (S_3) constraints. Additionally, equation (7) directly implies the satisfaction of each Conflict H_3 constraint. For every period and every curriculum, we introduce a binary variable $r_{cu,p}$ which takes on the value 1 if some course of curriculum cu is scheduled for period p. In constraint (8), we omit the term $r_{cu,p+1}$ if period p is the last period of the day. In the same way, we omit $r_{cu,p-1}$ if period p is the first period of the day. A binary variable $v_{cu,p}$ is set equal to 1 only if curriculum cu has an isolated lecture at period p.

Finally, the last constraint requires that the number of courses, which are taught by teacher t at period p, must be at most 1. This implies the satisfaction of the hard constraints H_3.

3.2 Lower Bounds using Lach and Lübbecke's Formulation

Table 2 lists the lower bounds obtained with Lach and Lübbecke's first IP stage for the first 14 instances. The bounds for the seven last instances (comp15–comp21) are omitted since they are not reported in [12]. The upper bounds are also included for information.

The results in Table 2 are generated with the commercial CPLEX11 solver, as well as with the non-commercial COIN-OR Cbc–2.2.2 solver, within a time limit set to 40 CPU units. One CPU unit, which depends on machine speed, corresponds to the time allowed for one run at the ITC-2007 competition. The bounds computed with CPLEX11, which are reported in Lach and Lübbecke's paper [12], are obtained within a time limit of 15,200 seconJΩ-336.555-14.444312359(h)-454.2377.91

Table 2
Bounds using the first IP stage of Lach and Lübbecke's IP model, solved by CPLEX11 with COIN-OR Cbc-2.2.2, within a time limit set to 40 CPU units. The current best upper and lower bounds are shown in Table 1.

	CPLEX11			COIN-OR Cbc-2.2.2		
Instance	UB	LB	gap%	UB	LB	gap%
comp01	4	4	0.00	4	4	0.00
comp02	45	11	77.04	67	6	91.04
comp03	66	25	62.12	94	25	73.62
comp04	35	28	21.61	39	27	30.89
comp05	365	108	70.43	614	58	90.66
comp06	37	10	72.97	74	12	83.78
comp07	6	6	0.00	6	4	47.23
comp08	37	37	0.00	39	24	39.80
comp09	99	46	53.65	127	47	63.66
comp10	4	4	0.00	6	4	33.33
comp11	0	0	0.00	0	0	0.00
comp12	546	53	90.34	723	57	92.12
comp13	61	41	33.72	75	34	54.98
comp14	51	46	9.92	72	40	44.44
Average gap			35.1			53.25

bounds than COIN-OR Cbc–2.2.2. The difference between the average gaps of the results produced by the two solvers confirms that CPLEX11 is more efficient than the non-commercial COIN-OR Cbc–2.2.2.

In the following sections, we show that better lower bounds can be obtained by our partition-based method even with COIN-OR Cbc–2.2.2.

4 A Partition-based Approach for Improved Lower Bounds

4.1 Rationale

Our partition-based approach for calculating CBCT lower bounds is inspired by the work of [19], which is based on the well-known "divide and conquer" principle. The general idea of this approach is as follows.

Exact solvers may fail to solve some large size instances but are often able to handle instances of reasonable size. Therefore, we can divide a large problem into several smaller sub-problems which are solved separately. By doing this, we ignore the constraints that link the sub-problems, and the set of the sub-problems constitutes a relaxation of the initial problem. Finally, the sum of the optimal values of the sub-problems or their lower bounds gives a lower bound of the initial problem.

The rationale of our partition-based approach for calculating CBCT lower bounds can be formally making use of the following notation and definitions.

- C denotes the set of all courses of a CBCT instance P.
- $\{X_p\}_{1\leq p\leq k}$ denotes a partition of C composed of p *classes* such that $\cup_{p=1}^{k} X_p = X$ and $\forall$, $i, j \in \{1, \ldots, k\}$, $i \neq j$, $X_i \cap X_j = \emptyset$.
- $\{P(X_p)\}_{1\leq p\leq k}$ represents the set of k sub-problems induced by $\{X_p\}_{1\leq p\leq k}$, such that each $P(X_p)$ is a sub-problem of P that is limited to the sole variables of X_p and the constraints involving only these variables.

  The set of sub-problems $\{P(X_p)\}_{1\leq p\leq k}$ is a relaxation of the initial problem P, where constraints linking the sub-problems are ignored.
- $\{LB_p\}_{1\leq p\leq k}$ identifies the set of lower bounds of the k sub-problems $\{P(X_p)\}_{1\leq p\leq k}$.

Then the following relation holds:

$$LB = \sum_{p=1}^{k} LB_p \leq f^*,$$

i.e. LB gives us a lower bound of the initial problem P.

Now the general procedure of our partition-based approach can be summarized as follows:

(1) Generate a partition $\{X_p\}_{1\leq p\leq k}$ of a course set C of an instance P.

(2) Solve each of the k sub-problems $P(X_p)$, $1 \leq p \leq k$, to obtain a set of optimal values or lower bounds $\{LB_p\}_{1\leq p\leq k}$ of these sub-problems.

(3) Sum up the values of the set $\{LB_p\}_{1\leq p\leq k}$ to obtain the desired lower bound LB of the instance P.

For step (1), we use an iterated Tabu Search algorithm to obtain optimized partitioning (Sect. 4.4). For step (2), we formulate the k sub-problems using the IP formulation given in Section 3.1 and solve them with the IP solver

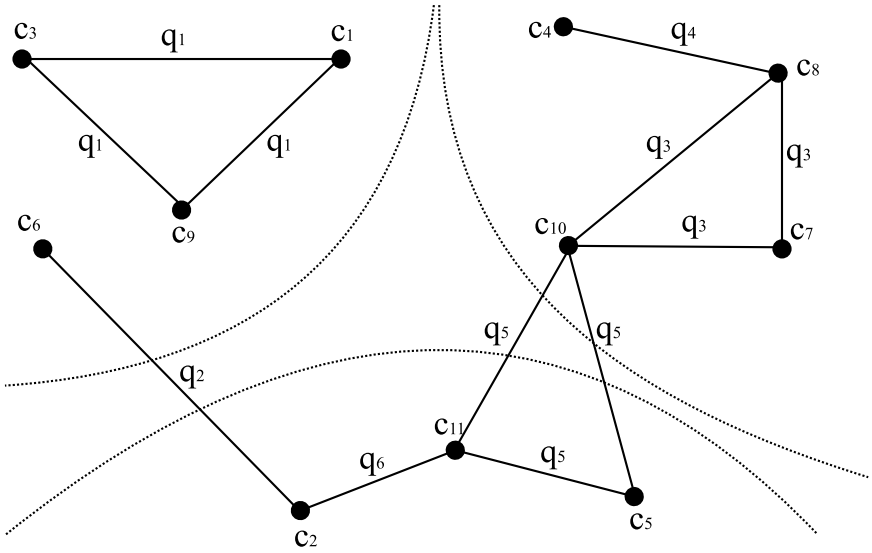

Fig. 2. An example of a partition composed of three classes.

COIN-OR Cbc–2.2.2.

4.2 Partition and Relaxed Constraints: an Illustrative Example

When the initial problem is partitioned into k sub-problems, the constraints linking the sub-problems are relaxed. Concretely, constraint relaxation concerns the following constraints: Curriculum Compactness (S_3), Room Occupancy (H_2) and Conflict (H_3). To visually illustrate the idea, we focus on the Curriculum Compactness constraints S_3, and for this purpose introduce a graph representation of S_3.

Definition 1 (Graph of Constraints): Let $C = \{c_1, c_2, ..., c_m\}$ be the set of m courses, and $CU = \{cu_1, cu_2, ..., cu_q\}$ be the set of q curricula. Then, graph $G = \{V, E\}$ consists of a set V of m nodes and a set of edges E between these nodes. Each node $v \in V$ corresponds to a course from the set C. For each pair of courses u and v in C, $\{u, v\} \in E$ if there exists at least one curriculum $cu_i \in CU$ such that $u \in cu_i$ and $v \in cu_i$.

According to this definition, the courses of each curriculum $cu_i \in CU$ are pairwise connected and form a clique. A k-partition of the graph leads to k distinct classes. If an edge joins two classes of the partition, this cutting edge as well as the other edges of the underlying clique are removed (relaxed). Therefore, a curriculum $cu_i \in CU$ is either kept or removed completely, since a reduced curriculum might lead to a higher compactness cost than the complete curriculum.

In Figure 2, we show an example of a problem with 11 courses and 6 cur-

Table 3
Trivial partitions of the problem instance comp12.

k	2	3	4	5	6
#RC	101	139	141	142	146
rLB	54	25	25	25	12

ricula, which is partitioned into three classes: $\{c_1, c_3, c_6, c_9\}$, $\{c_2, c_5, c_{11}\}$ and $\{c_4, c_7, c_8, c_{10}\}$. This partition leads to the removal of constraint $\{c_2 - c_6\}$ of curriculum q_2. Since the courses c_5, c_{10} and c_{11} of curriculum q_5 form a clique, we remove not only the two cutting edges $\{c_{10} - c_5, c_{10} - c_{11}\}$, but also the other edge $\{c_5 - c_{11}\}$ of the same clique.

Room Occupancy and Conflict constraints (H_2 and H_3) linking sub-problems of a partition are relaxed in a similar way.

We insist that within each sub-problem, hard constraints H_1 to H_4 are always maintained, as well as soft constraints $S_1 - S_3$. Since the Room Stability constraints (S_4) are excluded from the Lach and Lübbecke's formulation of the first stage (see Section 3.1), our partition-based approach completely neglects S_4.

4.3 Influence of the Partition and Criterion for Optimized k-Partitioning

In order to see the influence of the partition on lower bounds, let us first take a look at a trivial partition. Given a fixed number k, we divide the problem with m courses into k classes of equal size m/k. The first m/k courses go to class X_1, the second m/k courses go to class X_2, and so on.

Table 3 shows the results of this trivial partitioning on the problem instance "comp12" when k is varied from 2 to 6. The second row #RC indicates the number of relaxed S_3 constraints between the sub-problems. a ae

We focus on optimizing the k-partition for a fixed k. To determine the best partition, we vary k from 2 to k_{max} (in this work, $k_{max} = 6$).

4.4 An Iterative Tabu Search Algorithm for k-Partitioning

For the partitioning task, we devise an Iterative Tabu Search procedure that combines a TS algorithm [11] with a critical element-guided perturbation (CEGP) strategy [13].

Given a set of courses C and a fixed number k, the TS algorithm determines the partition $\{X_p\}_{1 \leq p \leq k}$ of the set C into k classes by minimizing the number of relaxed different curriculum compactness constraints (see Section 4.2) that link any two courses in different classes of the partition.

When this TS phase reaches its best local optimum, a *perturbation* phase using the CEGP strategy is applied to the best partition found to generate a perturbed new solution, from which a new round of Tabu Search starts. This TS↔Perturbation process repeats until a predetermined stop condition is verified. We next illustrate the TS procedure and the CEGP operator.

4.4.1 Tabu Search

To describe the TS components, we first define some terms used for the purpose of this discussion.

Definition 2. Unsatisfied curriculum (cu^{ns}): A curriculum is unsatisfied by a partition if the courses of the curriculum belong to more than one class of the partition. The unsatisfied curricula cu^{ns} form a subset CU^{ns} of CU, where CU is the set of all the curricula.

Definition 3. Unsatisfied course (c^{ns}): Given a subset of curricula $CU(c)$ containing course c, we say that c is unsatisfied by a partition if there is at least one curriculum of the subset $CU(c)$ which is left unsatisfied. The unsatisfied courses c^{ns} form a subset C^{ns} of C, where C is the set of all the courses.

Definition 4. Most unsatisfied course (c^{mns}): The most unsatisfied course c^{mns} is a course that belongs to the largest number of unsatisfied curricula.

Configuration: Given a set of courses C and a fixed number k, a configuration is any partition of C into k classes. Therefore, the search space consists of all such partitions.

Evaluation function: Given two k-partitions, the evaluation function prefers

the partition that leads to a less relaxed set of Curriculum Compactness constraints (S_3), i.e. unsatisfied curricula. Ties are broken by the number of courses (a smaller value is better) implied in the unsatisfied (relaxed) curricula.

Neighborhood: Our TS procedure is based on four neighborhoods listed below. Given a partition formed of k classes, the basic idea of these neighborhoods is to generate a new partition by moving one or more courses from one class to another, or by swapping courses of two different classes.

$N1$: Select the most unsatisfied course c^{mns}, and select randomly another unsatisifed course $c_i^{ns} \in C^{ns}$ in a different class that does not contain c^{mns}. Then, swap the two selected courses c^{mns} and c_i^{ns}.

$N2$: Select the most unsatisfied course c^{mns}, and an unsatisfied curriculum $cu_i^{ns} \in CU^{ns}$ in which appears c^{mns}. Then, move c^{mns} to another class where there is at least one course belonging to the unsatisfied curriculum cu_i^{ns}.

$N3$: Select an unsatisfied curriculum $cu_i^{ns} \in CU^{ns}$ in one class of the partition, and an unsatisfied curriculum $cu_j^{ns} \in CU^{ns}$ in another class of the partition. Then, swap their courses. If the size of one curriculum is bigger than the size of another curriculum, move the rest of the courses of the bigger curriculum to the other class of the partition.

$N4$: Select an unsatisfied curriculum $cu_i^{ns} \in CU^{ns}$ and move all its courses to a class where there is at least one course belonging to curriculum cu_i^{ns}.

Only the first neighborhood relation is symmetric, meaning that the size of each class of the partition is kept unchanged. The other three neighborhood relations are not symmetric, and the size of the partition classes might vary during the search. With these neighborhoods, we allow a course to be moved from class Xs to class Xt only if $|Xt| \leq s_{max}$, where s_{max} is the maximum number of elements allowed in a class. In this study, s_{max} is set to $1.2*(|C|/k)$, where $|C|$ and k are the number of courses and classes respectively. This value for s_{max} was experimentally determined and proved to be suitable for almost all the CBCT instances. Notice that fine tuning s_{max} for each instance would lead to improved results, though we have not undertaken to do this in the present paper.

Our TS procedure uses neighborhood union of N1 and N2 (denoted by $N1 \cup N2$) and neighborhood union $N3 \cup N4$ in a token-ring search [8]. A neighborhood union includes all the moves of the neighborhood relations in the union. In our token-ring search, TS applies $N1 \cup N2$ to reach its best local minimum, switches to $N3 \cup N4$ to reach another local minimum, and then switches back

to $N1 \cup N2$. This process continues until no improvement is possible any more.

Tabu list management: Each time a move is performed, the reversing move cannot be carried out for a certain number of iterations (tabu tenure). The tabu tenure of a move is tuned adaptively according to the following formula proposed in [19]:

$$date(mv) + freq(mv) \leq iter$$

where $date(mv)$ is the iteration number when move mv was selected, $freq(mv)$ the number of times mv was performed, $iter$ the current number of iterations.

Aspiration criteria and stop condition: The tabu status of a move is disabled if it leads to a solution which is better than the current best solution. Our tabu search procedure ends when the best solution cannot be improved any more within a certain number of moves.

4.4.2 Critical Element-Guided Perturbation (CEGP)

To diversify the search, we use the CEGP strategy to escape local optima [13]. Contrary to a random perturbation, CEGP takes into account the specific problem structure and favours perturbation of some "critical" variables. Our CEGP strategy consists of the following three steps:

(1) Scoring – giving each course a ranking score;
(2) Selection – choosing a certain number of courses according to their ranking scores;
(3) Perturbing – randomly perturbing the chosen courses.

Scoring: The most important question that we need to answer for CEGP is how to score each element (course). For each course $c \in C$ we keep a list of curricula Q_c in which c occurs. We give a score S_{qc_i} to each curriculum qc_i from the curriculum list Q_c. This score is obtained by subtracting the total number of courses belonging to curriculum qc_i from the number of courses of curriculum qc_i that are in the same class of the partition as course c. If $S_{qc_i} = 0$ then the curriculum qc_i is satisfied, e.g. all the courses of qc_i are in the same partition. Otherwise, $S_{qc_i} < 0$ which means that qc_i is unsatisfied.

The score S_c for course c is then obtained by summing the scores of all the n curricula qc_i from the curriculum list Qc:

$$S_c = \sum_{i=1}^{n} S_{qc_i}$$

Selection: We randomly select a certain number γ (called perturbation strength) of courses with $S_c < 0$. Throughout this paper, we experimentally set γ to 5

which proves to be a robust value allowing the search to escape most local optima.

Perturbation: We apply the union of neighborhood relations $N1$ and $N2$ with the selected courses in order to obtain a perturbed solution from which starts a new round of Tabu Search.

5 New Lower Bounds for the ITC2007 Instances

To estimate the efficiency of the proposed partition-based approach, we conduct experiments on the whole set of benchmark instances from the ITC-2007 except on instance comp11. Indeed, the upper bound of comp11 is 0 (see Table 1), so there was no need to search for the lower bound for this instance.

Our algorithm is programmed in C++, and compiled with GNU gcc on a Xeon E5440 with 2.83 GHz and 8GB. As previously indicated, we use the non-commercial IP solver COIN-OR Cbc–2.2.2 in the sub-problem resolution phase.

5.1 Best Lower Bounds with the Partition-based Approach

The first comparison is with the best lower bounds reported at the Curriculum-Based Course Timetabling website [3]. The solution quality is measured by the gap between the best known upper bound bUB and the generated lower bound LB, computed as $(bUB - LB)/bUB$, where $bUB \neq 0$. It should be mentioned that the exact conditions used to obtain the best known bounds, such as running time, are not available from the website. To obtain our lower bounds, we set the time limit to 7 hours per sub-problem for COIN-OR Cbc-2.2.2, and vary the number of classes k from 2 to 6. The computational time needed in the partitioning phase with Tabu Search is at most 10 minutes, which is negligible compared to the time spent in the sub-problem resolution phase. Given the stochastic nature of our approach, we run the algorithm 5 times for each value of k, and report the best lower bound achieved.

Table 4 shows the current best upper and lower bounds reported at the CBCT website [3] (columns 2–3). Notice that the best lower bounds in column 3 also include 4 recently improved bounds from [4], indicated with an asterisk symbol. The fourth column indicates the gap between the best bounds ever obtained. The results of our partition-based approach are given in columns 5–8. Column 5 shows the value of k. As before, we give the number of relaxed constraints #RC. Finally, the last column shows the gap between the best-

Table 4
A comparison between the best partition-based lower bounds, and the best know bounds ever obtained from[3].

	Best known bounds			Best partition-based lower bounds			
Inst.	bUB	bLB	gap (%)	k	#RC	LB	gap (%)
comp01	**5**	5*	**0.00**	2	0	4	20.00
comp02	24	10	58.33	6	16	12	50.00
comp03	66	38	42.42	6	13	38	42.42
comp04	**35**	**35**	**0.00**	5	1	**35**	**0.00**
comp05	291	114*	60.82	4	70	183	37.11
comp06	27	16*	40.74	5	10	22	18.52
comp07	**6**	**6**	**0.00**	4	9	**6**	**0.00**
comp08	**37**	**37**	**0.00**	3	2	**37**	**0.00**
comp09	96	66*	31.25	6	7	72	25.00
comp10	**4**	**4**	**0.00**	6	16	**4**	**0.00**
comp11	**0**	**0**	**0.00**	4	0	**0**	**0.00**
comp12	300	53	82.33	5	44	109	63.67
comp13	**59**	48	18.64	4	1	**59**	**0.00**
comp14	**51**	**51**	**0.00**	3	6	**51**	**0.00**
comp15	66	41	37.88	5	8	38	42.42
comp16	18	13	27.78	4	8	16	11.11
comp17	56	44	21.43	5	14	48	20.00
comp18	62	0	100.00	4	19	24	63.08
comp19	57	49	14.03	3	2	56	0.02
comp20	4	0	100.00	5	12	2	50.00
comp21	83	0	100.00	5	17	61	26.51
Average gap			35.03				22.37

known bUB and our LB. Optimal solutions are given in bold.

Table 4 discloses that we were able to prove optimality for 6 instances. Among these 6 instances, the optimality of instance comp13 is proven for the first time. In addition, for all the instances except 2 cases (comp01 and comp15), we managed to improve on or reach the current best lower bound. As the last row of Table 4 shows, the partition-based approach reduced the average gap to the best upper bounds from 35.03% to 22.37%.

5.2 Comparisons with the Results of Lach&Lübbecke [12] and Burke et al.[4]

We now turn our attention to comparing our approach and the two approaches of [12] and [4]. Since the bounds for the last seven instances are not reported in these studies, we carry out this experiment using only the first 14 instances. For this experiment, a completely fair comparison is impossible since we have a non-deterministic approach (i.e. our approach) on the one hand and two

deterministic approaches (i.e. [12] and [4]) on the other hand. In addition, the use of two different solvers (i.e. commercial CPLEX11 and non-commercial COIN-OR Cbc-2.2.2) constitutes another source of difficulty for a fair comparison. This comparison is thus presented only for indicative purposes and should be interpreted with caution.

For this experiment, we fix $k = 5$ (except for the two small instances comp01 and comp11, $k = 2$) instead of varying it. To normalize the computing time used by different approaches, we follow the ITC–2007 competition rule and use a benchmarking program from the ITC-2007 website to determine the speed of our computer. One CPU unit which is the time permitted for one run at the ITC-2007 corresponds to 300 seconds on our machine. Similarly, one CPU time unit is equivalent to 380 and 780 seconds on the machines used in [4] and [12]. In these two studies, results are reported for 1 CPU, 10 CPU and 40 CPU units using CPLEX11. Though the same ILP model is used across all the CPU units in [12], different models and parameters are employed in [4] for the three CPU time units.

In this experiment, we apply our algorithm once to each problem instance using 1 CPU, 10 CPU, and 40 CPU units as the total time allowed to solve the k sub-problems of each instance. As in the first experiment, the time needed in the partitioning phase is at most 10 minutes. Results are presented in Table 5.

The outcome shown for each CBCT instance across each time unit makes it clear that, except in rare cases, the bounds of the partition-based approach are better than those reported in the first reference approach [12]. Comparing the other reference approach [4] and the partition-based approach, the results show that our approach remains competitive, leading to a slightly improved average gap across each time unit. However, the lower bounds generated with the proposed approach can be better appreciated if one takes into consideration the fact that the approach of [4] is among the current best ones, and its results for different CPU units are based on different models and parameters. In addition, we suspect that better lower bounds could be achieved if the more powerful CPLEX solver is used in solving of sub-problems within this partition-based approach.

6 Conclusion

We have demonstrated the efficacy of a partition-based approach for computing lower bounds for the Curriculum-Based Course Timetabling problem from the ITC-2007. Our approach, based on the "divide and conquer" principle, consists of three stages. In the first stage, we partition courses into a fixed

Table 5. Following Burke *et al.* [4], comparison of lower bounds on the first 14 instances reported in [4] and [12], with the partition-based bounds obtained in 1 CPU unit, 10 CPU units and 40 CPU units respectively.

	Lach and Lübbecke [12]						Burke *et al.* [4]						Our approach					
	1 CPU		10 CPU		40 CPU		1 CPU		10 CPU		40 CPU		1 CPU		10 CPU		40 CPU	
Instance	LB	gap%	LB	gap%	LB	gap%	LB	gap%	LB	gap%	LB	gap%	LB	gap%	LB	gap%	LB	gap %
comp01	4	**20.00**	4	**20.00**	4	20.00	0	100.00	4	**20.00**	5	**0.00**	4	**20.00**	4	**20.00**	4	20.00
comp02	0	100.00	8	66.66	11	54.17	0	100.00	0	100.00	1	95.83	10	**58.33**	12	**50.00**	12	**50.00**
comp03	0	100.00	0	100.00	25	62.12	25	62.12	33	50.00	33	**50.00**	26	**60.60**	34	**48.48**	36	**45.45**
comp04	22	37.14	28	20.00	28	20.00	35	**0.00**	35	**0.00**	35	**0.00**	35	**0.00**	35	**0.00**	35	**0.00**
comp05	92	68.38	103	64.60	108	62.89	119	**59.11**	111	**61.86**	114	**60.82**	19	93.47	69	76.29	80	72.51
comp06	7	74.07	10	62.96	10	62.96	13	**51.85**	15	**44.44**	16	**40.74**	12	55.55	12	55.55	16	**40.74**
comp07	0	100.00	2	66.66	6	**0.00**	6	**0.00**	6	**0.00**	6	**0.00**	5	16.67	6	**0.00**	6	**0.00**
comp08	30	18.92	34	8.12	37	**0.00**	37	**0.00**	37	**0.00**	37	**0.00**	37	0.00	37	**0.00**	37	**0.00**
comp09	37	61.46	41	57.29	46	52.08	68	**29.17**	65	32.29	66	31.25	39	59.38	67	**30.21**	67	**30.21**
comp10	2	50.00	4	**0.00**	4	**0.00**	3	25.00	4	**0.00**	4	**0.00**	4	**0.00**	4	**0.00**	4	**0.00**
comp11	0	**0.00**	0	**0.00**	0	**0.00**	0	**0.00**	0	**0.00**	0	**0.00**	0	**0.00**	0	**0.00**	0	**0.00**
comp12	29	90.33	32	89.33	53	82.74	101	**66.33**	95	**68.33**	95	**68.33**	43	85.67	78	74.00	84	72.00
comp13	33	44.06	39	33.90	41	30.50	52	**11.86**	52	11.86	54	8.47	46	22.03	53	**10.17**	55	**6.78**
comp14	40	21.57	41	19.61	46	**9.80**	41	**19.60**	42	17.65	42	17.65	41	19.61	43	**15.69**	43	15.69
Average gap		56.13		43.51		32.63		37.50		29.03		26.65		35.09		27.17		25.24

number k of partitions with an iterative Tabu Search heuristic. In that way, we obtain k sub-problems which are formulated as integer linear problems using the formulation of Lach and Lübbecke [12]. In the third stage, these k sub-problems are solved (exactly or approximatively) with an IP solver. The sum of the solutions of these sub-problems leads to a lower bound of the initial problem.

Applied to the whole set of 21 ITC-2007 benchmark instances, our partition-based approach has improved on the current best lower bounds for 12 instances, proved optimality for the first time for 1 instance and achieved the previously known optimal bounds for 5 other instances. Only in two cases, our bounds are slightly worse than the best known bounds from the literature. Our new bounds have reduced the average gap to the best known upper bounds from 35.03% to 22.37%.

Acknowledgment

This work was partially supported by the Region of "*Pays de la Loire*" within Radapop and LigeRO projects. We would like to thank the anonymous referees for their helpful comments and questions on the paper. We are also grateful to the following people for their inputs and feedbacks: the ITC-2007 organizers for providing the benckmark instances, the authors of [4], in particular Dr. J. Marecek and Dr. A. J. Parkes for answering our requests on their work. We are particularly grateful to Prof. F. Glover for his valuable help in preparing the final version of the paper.

References

[1] S.M. Al-Yakoob and H.D. Sherali. A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations. *European Journal of Operational Research*, 180(3): 1028–1044, 2007.

[2] M. Atsuta. General Purpose Approach to Various Types of Timetabling Problems (in Japanese). Master thesis, Department of Informatics, Kwansei Gakuin University, March 2009.

[3] A. Bonutti. Curriculumbased Course Timetabling. Retrieved November 2010, from `http://tabu.diegm.uniud.it/ctt/index.php`.

[4] E.K. Burke, J. Marecek, A.J. Parkes, and H. Rudova. Decomposition, Reformulation, and Diving in University Course Timetabling. *Computers & Operations Research*, 37(3): 582–597, 2010.

[5] P. De Causmaecker, P. Demeester, and G. Vanden Berghe. A decomposed metaheuristic approach for a real-world university timetabling problem. *European Journal of Operational Research*, 195(1): 307–318, 2009.

[6] M. Clark, M. Henz and B. Love. QuikFix–A Repair-based Timetable Solver. *Proceedings of the Seventh International Conference on the Practice and Theory of Automated Timetabling*, http://www.comp.nus.edu.sg/~henz/publications/ps/PATAT2008.pdf.

[7] L. Di Gaspero, B. McCollum, and A. Schaerf. The Second International Timetabling Competition (ITC-2007): CBCT (Track 3). Technical Report 2007/08/01, University of Udine DIEGM, Udine, Italy, 2007. http://www.cs.qub.ac.uk/itc2007.

[8] L. Di Gaspero and A. Schaerf. Neighborhood Portfolio Approach for Local Search Applied to Timetabling Problems. *Journal of Mathematical Modeling and Algorithms*, 5(1): 65–89, 2006.

[9] M. Garey and D. Johnson. Computers & Intractability: A Guide to the Theory of NP-Completeness. In W.H. Freeman and Company (Eds), 1979.

[10] M.J. Geiger. Applying the Threshold Accepting Metaheuristic to Curriculum based Course Timetabling. To appear in *Annals of Operations Research*. DOI:10.1007/s10479-010-0703-4.

[11] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, Boston, 1997.

[12] G. Lach and M.E. Lübbecke. Curriculum Based Course Timetabling: New Solutions to Udine Benchmark Instances. *Annals of Operations Research*. DOI:10.1007/s10479-010-0700-7.

[13] Z. Lü and J.K. Hao. A Critical Element-Guided Perturbation Strategy for Iterated Local Search. In C. Cotta and P. Cowling. (Eds) EvoCOP-2009, LNCS 5482: 1–12, Springer, 2009.

[14] Z. Lü and J.K. Hao. Adaptive Tabu Search for Course Timetabling. *European Journal of Operational Research*, 200(1): 235–244, 2010.

[15] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A.J. Parkes, L. Di Gaspero, R. Qu, and E.K. Burke. Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. *INFORMS Journal on Computing*, 22(1): 120–130, 2010.

[16] T. Muller. ITC2007 Solver Description: A Hybrid Approach. *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, http://www.asap.cs.nott.ac.uk/patat/patat08/patat08-full-papers.shtml.

[17] K. Nonobe and T. Ibaraki. An Improved Tabu Search Method for the Weighted Constraint Satisfaction Problem. *INFOR*, 39(2): 131–151, 2001.

[18] J. van den Broek, C. Hurkens and G. Woeginger. Timetabling problems at the TU Eindhoven. *European Journal of Operational Research*, 196(3): 877–885, 2009.

[19] M. Vasquez and J.K. Hao. Upper Bounds for the SPOT 5 Daily Photograph Scheduling Problem. *Journal of Combinatorial Optimization*, 7(1): 87–103, 2003.