Solving the Course Timetabling Problem with a Hybrid Heuristic Algorithm[⋆]

Zhipeng Lü[1,2] and Jin-Kao Hao[1]

[1] LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
[2] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
`zhipeng.lui@gmai.com, hao@info.univ-angers.fr`

Abstract. The problem of curriculum-based course timetabling is studied in this work. In addition to formally defining the problem, we present a hybrid solution algorithm (Adaptive Tabu Search–ATS), which is aimed at minimizing violations of soft constraints. Within ATS, a new neighborhood and a mechanism for dynamically integrating Tabu Search with perturbation (from Iterated Local Search) are proposed to ensure a continuous tradeoff between intensification and diversification. The performance of the proposed hybrid heuristic algorithm is assessed on two sets of 11 public instances from the literature. Computational results show that it significantly improves the previous best known results on two problem formulations.

Keywords: Timetabling, hybrid heuristic, tabu search, iterated local search, constraint solving.

1 Introduction

In recent decades, timetabling has become an area of increasing interest in the community of both research and practice [11]. In essence, timetabling consists of assigning a number of events, each with a number of features, to a limited number of timeslots and rooms subject to certain (hard and soft) constraints. In this paper, we consider one of the problems in the category of educational timetabling–the so-called curriculum-based course timetabling (CCT), the formulation of which was recently proposed as the third track of the Second International Timetabling Competition (ITC–2007)[1]. This competition aims to close the gap between research and practice within the area of educational timetabling.

The general course timetabling problem is known to be difficult and has been proved to be NP-hard [4]. In this context, exact solutions would be only possible for problems of limited sizes. Instead, algorithms based on metaheuristics have shown to be a highly effective approach to this kind of problems (see e.g. [1,3,15]).

[⋆] This algorithm is ranked the second place for the track 3 of the Second International Timetabling Competition (ITC–2007).

[1] http://www.cs.qub.ac.uk/itc2007/

Interested readers are referred to [9] for a comprehensive survey of the automated approaches for university course timetabling presented in recent years.

In this paper, we present a hybrid Adaptive Tabu Search (ATS) algorithm for the CCT problem. We introduce a combined use of two neighborhoods including an original and very powerful double Kempe chain neighborhood. Moreover, we devise a mechanism for dynamically combining TS with a perturbation operator in order to adaptively escape from local optima and to automatically make more intensive search when promising regions of the search space are visited. Consequently, it ensures a continuous tradeoff between intensification and diversification for the search process. The performance of the ATS algorithm is assessed with the two sets of 11 instances from the literature, showing very competitive results.

The rest of this paper is organized as follows. Section 2 describes the first mathematical formulation of the CCT problem. Section 3 presents the hybrid heuristic–Adaptive Tabu Search algorithm. In Section 4 computational experiments are carried out. Conclusions are drawn in the last section.

2 Curriculum-Based Course Timetabling

2.1 Problem Description

The CCT problem of the ITC–2007 consists of scheduling all lectures of a set of courses into a weekly timetable, where each lecture of a course must be assigned a period and a room in accordance with a given set of constraints. In this problem, all hard constraints must be strictly satisfied and the number of soft constraint violations should be minimized. A feasible timetable is one in which all lectures have been scheduled at a period and a room, so that the hard constraints are satisfied. The four hard constraints H1∼H4 and four soft constraints S1∼S4 are defined as follows:

- **H1. Lectures.** All lectures of a course must be scheduled to a distinct period and a room.
- **H2. Room Occupancy.** Any two lectures cannot be assigned in the same period and the same room.
- **H3. Conflicts.** Lectures of courses in the same curriculum or taught by the same teacher cannot be scheduled in the same period, i.e., any period cannot have an overlapping of students or teachers.
- **H4. Availability.** If the teacher of a course is not available at a given period, then no lectures of the course can be assigned to that period.

- **S1. Room Capacity.** For each lecture, the number of students attending the course should not be greater than the capacity of the room hosting the lecture.
- **S2. Room Stability.** All lectures of a course should be scheduled at the same room. If this is impossible, the number of occupied rooms should be as few as possible.

- **S3. Minimum Working Days.** The lectures of a course should be spread into the given minimum number of days.
- **S4. Curriculum Compactness.** For a given curriculum a violation is counted if there is one lecture not adjacent to any other lecture belonging to the same curriculum within the same day, which means the agenda of students should be as compact as possible.

We present below a first mathematical formulation of the problem which is missing in the literature.

2.2 Problem Formulation

The CCT problem consists of a set of n courses $C = \{c_1, c_2, \ldots, c_n\}$ to be scheduled in a set of p periods $T = \{t_1, t_2, \ldots, t_p\}$ and a set of m rooms $R = \{r_1, r_2, \ldots, r_m\}$. Each course c_i is composed of l_i lectures (of one timeslot) to be scheduled. A period is a pair composed of a day and a timeslot and p periods are distributed in d week days and h daily timeslots, i.e., $p = d \times h$. In addition, there are a set of s curricula $CR = \{cr_1, cr_2, \ldots, cr_s\}$ where each curriculum cr_k is a group of courses that share common students.

For the solution representation, we chose a direct solution representation to make things as simple as possible. A candidate solution consists of $p \times m$ matrix X where $x_{i,j}$ corresponds to the course label assigned at period t_i and room r_j. If there is no course assigned at period t_i and room r_j, then $x_{i,j} = -1$. With this representation we assure that there will be no more than one course assigned to a room in any period, meaning that the second hard constraint H2 is always satisfied. For courses, rooms, curricula and solution representation X, a number of notations and definitions are presented in table 1.

Table 1. Table of symbols and variables

Symbol	Description
std_i	the number of students attending course c_i
l_i	the number of lectures of course c_i
tc_i	the label of the teacher instructing course c_i
mwd_i	the number of minimum working days of course c_i
cap_j	the capacity of room r_j
cr_k	the kth curriculum including a set of courses $\{c_{k1}, \ldots, c_{kv}\}$
$una_{i,j}$	whether course c_i is unavailable at period t_j. $una_{i,j} = 1$ if it is unavailable, $una_{i,j} = 0$ otherwise
$x_{i,j}$	the label of the course assigned at period t_i and room r_j
$nr_i(X)$	the number of rooms occupied by course c_i for a candidate solution X
$nd_i(X)$	the number of working days that course c_i takes place at for a candidate solution X
$cra_{k,i}(X)$	$cra_{k,i}(X) = 1$ if one lecture of any course in curriculum cr_k is scheduled at period t_i, $cra_{k,i}(X) = 0$ otherwise.

Given these notations, we redescribe the CCT problem in a formal way for a candidate solution X. The four hard constraints and four soft constraints are:

- **H1. Lectures.** $\forall c_k \in C$,

$$\sum_{i,j} sl_k(x_{i,j}) = l_k$$

$$sl_k(x_{i,j}) = \begin{cases} 1, \text{ if } x_{i,j} = c_k; \\ 0, \text{ otherwise.} \end{cases}$$

- **H2. Room Occupancy.** this hard constraint is automatically satisfied in our solution representation.
- **H3. Conflicts.** $\forall x_{i,j}, x_{i,k} \in X, x_{i,j} = c_u, x_{i,k} = c_v$,

$$(\forall cr_q, c_u \notin cr_q \vee c_v \notin cr_q) \wedge (tc_u \neq tc_v)$$

- **H4. Availability.** $\forall x_{i,j} \in X, x_{i,j} = c_k$,

$$una_{k,i} = 0$$

For the four soft constraints, their penalty costs are represented as:

- **S1. Room Capacity.** $\forall x_{i,j} \in X, x_{i,j} = c_k$,

$$f_{rc}(x_{i,j}) = \begin{cases} \alpha_1 \cdot (std_k - cap_j), \text{ if } std_k > cap_j; \\ 0, \qquad \text{otherwise.} \end{cases}$$

- **S2. Room Stability.** $\forall c_i \in C$,

$$f_{rs}(c_i) = \alpha_2 \cdot (nr_i(X) - 1)$$

- **S3. Minimum Working Days.** $\forall c_i \in C$,

$$f_{md}(c_i) = \begin{cases} \alpha_3 \cdot (mwd_i - nd_i(X)), \text{ if } nd_i(X) < mwd_i; \\ 0, \qquad \text{otherwise.} \end{cases}$$

- **S4. Curriculum Compactness.** $\forall x_{i,j} \in X, x_{i,j} = c_k$,

$$f_{cc}(x_{i,j}) = \alpha_4 \cdot \sum_{cr_q \in CR} c\_cr_{k,q} \cdot iso_{q,i}(X)$$

where

$$c\_cr_{k,q} = \begin{cases} 1, \text{ if } c_k \in cr_q; \\ 0, \text{ otherwise.} \end{cases}$$

$$iso_{q,i}(X) = \begin{cases} 1, \text{ if } (i\%h = 1 \vee cra_{q,i-1}(X) = 0) \wedge (i\%h = 0 \vee cra_{q,i+1}(X) = 0); \\ 0, \text{ otherwise.} \end{cases}$$

% is the modulo operator. One observes that in S4 the calculation is only limited within the same day. $iso_{q,i}(X) = 1$ means that there is no any course in the curriculum cr_q scheduled adjacent (before or after) to the timeslot $i\%h$ in the $[i/h]$th day (h denotes the total number of timeslots per day). More specifically, curriculum cr_q does not appear before (after) period t_i means that t_i is the first (last) timeslot of a working day or cr_q does not appear at t_{i-1} (t_{i+1}).

α_1, α_2, α_3 and α_4 are the unit penalty point for each of the soft constraints. Notice that $\alpha_1 \sim \alpha_4$ are fixed in the problem formulation and should not be confused with the penalty parameters used by some solution procedures. In the literature, there are two versions of the CCT—old version and ITC–2007 competition version. We call them formulation **I** and **II** respectively. The old version ignores the second soft constraint S2 and fixes the values of $\alpha_1 \sim \alpha_4$ as:

$$\alpha_1 = 1, \alpha_2 = 0, \alpha_3 = 5, \alpha_4 = 1$$

The competition version used in ITC–2007 sets $\alpha_1 \sim \alpha_4$ as:

$$\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 5, \alpha_4 = 2$$

With the above formulation, we can then calculate the total soft penalty cost for a given candidate feasible solution X according to the cost function f defined in Eq. (1). The goal is then to find a feasible solution X that minimizes the following function.

$$f(X) = \sum_{x_{i,j} \in X} f_{rc}(x_{i,j}) + \sum_{c_i \in C} f_{rs}(c_i) + \sum_{c_i \in C} f_{md}(c_i) + \sum_{x_{i,j} \in X} f_{cc}(x_{i,j}) \qquad (1)$$

3 Hybrid Heuristic Algorithm for CCT

The basic idea of our hybrid heuristic algorithm is to combine the advantageous features of Tabu Search (TS) [8] and Iterated Local Search (ILS) [10]. Similar to the idea of [13], we devise in this work an Adaptive TS algorithm whose components and mechanisms are described in the following subsections.

TS is based on the belief that intelligent searching should be systematically based on adaptive memory and learning. TS can be used with both long and short computing budgets. In general, long computing budgets would lead to better results. However, if the total computation time is limited, it would be preferred to combine short TS runs with some robust diversification operators.

Interestingly, ILS provides such diversification mechanisms to guide the search to escape from a local optimum and move towards new regions in the solution space. When the best known solution cannot be improved any more using the TS algorithm, we employ a penalty-guided perturbation operator to destruct the obtained local optimum.

Note that starting from an empty timetable, we generate first an initial feasible solution by means of a graph coloring greedy heuristic. Because of the page limit, the details of this greedy heuristic are omitted here. We simply mention that for all the tested instances, this greedy heuristic can easily obtain feasible solutions. Once a feasible timetable that satisfies all the hard constraints is reached, our ATS algorithm is used to minimize the soft constraint cost function (Eq. (1)) without breaking hard constraints any more. Therefore, the search space of our ATS algorithm is limited to the feasible timetables.

One interesting issue concerns the influence of the initial solution on the final solution reached by ATS. Experimentations (not reported here) show that ATS is not sensitive to the quality of the initial solution.

3.1 Neighborhood Structure

In a neighborhood search procedure, applying a move *mv* to a candidate solution X leads to a new solution denoted by $X \bigoplus mv$. Let $M(X)$ be the set of all possible moves which can be applied to X and does not create any infeasibility, then the neighborhood of X is defined by: $N(X) = \{X \bigoplus mv | mv \in M(X)\}$. For the CCT problem, we use two distinct *move*s denoted by *SimpleSwap* and *KempeSwap*. Respectively, two neighborhoods denoted by N_1 and N_2 are defined as follows.

Neighborhood N_1**:** A *SimpleSwap* move consists in exchanging the hosting periods and rooms assigned to two lectures of different courses. Applying the *SimpleSwap* move to two different courses $x_{i,j}$ and $x_{i',j'}$ for the solution X consists in assigning the value of $x_{i,j}$ to $x_{i',j'}$ and inversely the value of $x_{i',j'}$ to $x_{i,j}$. Note that moving one lecture of a course to a free position is a special case of the *SimpleSwap* move where one of the swapping lectures is null and it is also included in our neighborhood N_1.

Neighborhood N_2**:** Our *KempeSwap* move is defined by interchanging two Kempe chains. If we focus only on courses and conflicts, each problem instance can be seen as a graph G where nodes are courses and edges connect courses with students or teacher in common. In a feasible timetable, a Kempe chain is the set of nodes that form a connected component in the subgraph of G induced by the nodes that belong to two periods. A *KempeSwap* produces a new feasible assignment by swapping the period labels assigned to the courses belonging to two specified Kempe chains. Once courses have been scheduled to periods, the room assignment can be done by solving a bipartite matching problem [15,16]. In this paper, we implement an exact algorithm–the augmenting paths algorithm introduced in [15,16].

More formally, let K_1 and K_2 be two Kempe chains in the subgraph with respect to two periods t_i and t_j, a *KempeSwap* produces an assignment by replacing t_i with $(t_i \backslash (K_1 \cup K_2)) \cup (t_j \cap (K_1 \cup K_2))$ and t_j with $(t_j \backslash (K_1 \cup K_2)) \cup (t_i \cap (K_1 \cup K_2))$. For instance, figure 1 depicts a subgraph deduced by two periods t_i and t_j and there are four Kempe chains: $K_a = \{c_1, c_2, c_7, c_8, c_{10}\}$, $K_b = \{c_3, c_6, c_9\}$, $K_c = \{c_4, c_{11}, c_{12}\}$ and $K_d = \{c_5\}$. Then, if we swap two Kempe chains K_b and K_c, a *KempeSwap* produces an assignment by moving $\{c_3, c_4, c_6\}$ to t_j and $\{c_9, c_{11}, c_{12}\}$ to t_i.

It is noteworthy to notice that our double Kempe chains interchange can be considered as a *generalization* of the single Kempe chain interchange known in the literature. In the previous definition of single Kempe chain neighborhood, each move concerns only one connected component, i.e., one of the two Kempe chains in our definition is empty [3,2,5,12]. Formally, it means replacing t_i with $(t_i \backslash K) \cup (t_j \cap K)$ and t_j with $(t_j \backslash K) \cup (t_i \cap K)$ where K is the non-empty Kempe chain [3,2,5,12]. Consequently, the single Kempe chain interchange is a special case of our *KempeSwap* move and it is included in our neighborhood N_2. Although not mentioned in this paper, a detailed analysis and comparison of those and other neighborhoods is conducted, showing the efficiency of the newly proposed double Kempe chains neighborhood.

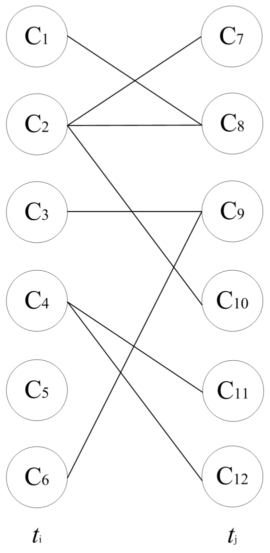

Fig. 1. Kempe chain illustrations

3.2 TS Using a Combined Neighborhood

The basic search engine of our ATS algorithm is of course based on TS. The TS procedure exploits the two neighborhoods N_1 and N_2 in a token-ring way [7]. More precisely, we start the TS procedure with one neighborhood. When the search ends with its best local optimum, we restart TS from this local optimum, but with the other neighborhood. This process is repeated until no improvement is possible and we say a TS phase is achieved. In our case, the TS procedure begins from the basic neighborhood N_1 and then neighborhood N_2: $N_1 \rightarrow N_2 \rightarrow N_1 \rightarrow N_2 ....$

Within TS, a *tabu list* is introduced to forbid the previously visited moves. At each iteration, a best non-tabu move mv is applied to the current solution X even if $X' = X \bigoplus mv$ does not improve the solution quality. In our TS, when moving one lecture from one position (period-room pair) to another (N_1), or from one period to another (N_2), this lecture cannot be moved back to the original position (N_1) or period (N_2) for the next tt iterations (tt is called tabu tenure). Tabu tenure tt of a lecture x is tuned adaptively according to the current solution quality f and the frequency of the move $freq(x)$, i.e.

$$tt(x) = f + \varphi \cdot freq(x)$$

where φ is a parameter that lies in [0, 1]. The aspiration criterion accepts a tabu move if it improves the current best result or the set of non-tabu moves is empty in the current neighborhood. The TS procedure based on each neighborhood stops when the best solution cannot be improved within a given number of steps (denoted by θ) and we call this number the *depth of TS*. The basic TS procedure is described in algorithm 1:

Algorithm 1. Tabu Search procedure: TS(X_0,θ)

```
1: //$X_0$ is the feasible initial solution
2: //$\theta$ is *the depth of TS*
3: **repeat**
4:     $X^* = TS_{N_1}(X_0)$ based on $N_1$ with *depth of TS* $\theta$
5:     $X^{*'} = TS_{N_2}(X^*)$ based on $N_2$ with *depth of TS* $\theta/3$
6:     $X_0 = X^{*'}$
7: **until** (no improvement is reached)
```

Because of the high computational effort in neighborhood evaluation of N_2, TS uses a much smaller depth (empirically fixed at $\theta/3$) (line 5) when N_2 is used. Note that the token-ring search based on TS stops when no improvement is possible. At this point, a TS phase is finished.

3.3 Perturbation

When a Tabu Search phase terminates, we employ a perturbation operator to destruct the reached local optimum in order to restart a new TS phase from this perturbed solution. Our perturbation operator consists of randomly selecting a given number of *SimpleSwap* or *KempeSwap* moves, where at least one of the moved lectures belongs to the first k highly-penalized ones. Specifically, when the current TS phase terminates, all the lectures are ranked in a non-increasing order according to their soft costs involved. Then, a certain number of lectures are selected from the first k-ranked (highly-penalized) ones. Notice that constraining the choice to highly-penalized lectures is essential because it is these lectures that contribute strongly to constraint violations (and the cost function).

Obviously, the *perturbation strength* (denoted by γ) is one of the most important ingredients of ILS and determines the quality gap between the two solutions before and after perturbation. In our case, γ is adaptively adjusted and takes values in an interval $[\gamma_{min}, \gamma_{max}]$ (set experimentally $\gamma_{min} = 4, \gamma_{max} = 15$). For acceptance criterion in the perturbation process, we use a strong exploitation technique, i.e., only better solutions are accepted.

3.4 Combination of TS with Perturbation

The *depth of TS* θ and the *perturbation strength* γ are two essential parameters which control the behavior of the ATS algorithm. On the one hand, a greater θ value ensures a more intensive search. On the other hand, greater γ corresponds to more possibility of escaping from the current local minimum. In order to get a continuous tradeoff between intensification and diversification, we devise a mechanism to dynamically and adaptively adjust these two important parameters according to the historical search records. Note that in this paper the initial values of these two and other parameters are empirically set and they are all instance-independent. It is possible that better solutions would be found by using a set of instance-dependent parameters. However, our aim is to design a robust solver which is able to solve efficiently a large panel of instances.

At the beginning of the search, we take a basic TS where the *depth of TS* θ is a small positive number, say $\theta = \theta_0$ ($\theta_0 = 10$). When TS cannot improve its best solution, perturbation is applied to this best solution with a weak strength ($\gamma = \gamma_{min}$). When the search progresses, we record the number of TS phase iterations (denoted by ξ) for which no improved solution has been found. The *depth of TS* θ and the *perturbation strength* γ are dynamically adjusted as follows: When the local minimum obtained by TS is promising, i.e., when it is close to the current best solution ($f \leq f_{best} + 2$), the *depth of TS* is gradually increased to ensure a more and more intensive search until no improvement is possible, i.e., $\theta = (1 + \eta)\theta$ at each iteration ($\eta = 0.6$). Similarly, *perturbation strength* is gradually increased so as to diversify more strongly the search if the number of non-improving TS phase iterations increases.

In this paper, we use the timeout condition required by the ITC–2007 competition rules (see next Section). Finally, our hybrid ATS algorithm is described in algorithm 2.

Algorithm 2. Adaptive Tabu Search scheme

```
1: $X_0$ is a feasible solution, $X^*$ is the best solution found so far
2: set $\xi = 0$, $\theta = \theta_0$, $\gamma = \gamma_{min}$
3: apply TS to $X_0$ with depth of TS $\theta$: $X^* = TS(X_0, \theta)$
4: repeat
5:    perturb $X^*$ with perturbation strength $\gamma$, get $X'$
6:    apply TS to $X'$ with depth of TS $\theta$, get $X^{*'}$
7:    if the local minimum solution $X^{*'}$ is promising, i.e., $f(X^{*'}) \leq f(X^*) + 2$ then
8:       repeat
9:          call the TS procedure with a gradually increased $\theta$: $\theta = (1 + \eta)\theta$
10:      until no better solution is obtained
11:   end if
12:   if better solution $X^{*'}$ has been found, i.e., $f(X^{*'}) < f(X^*)$ then
13:      accept $X^{*'}$ as the current best solution: $X^* = X^{*'}$
14:      reset to the basic TS ($\theta = \theta_0$) with weak perturbation ($\gamma = \gamma_{min}$)
15:   else
16:      reset to the basic TS: $\theta = \theta_0$
17:      $\xi = \xi + 1$
18:      update the perturbation strength: $\gamma = min\{\gamma_{min} + \lambda \cdot \xi, \gamma_{max}\}$
19:   end if
20: until (timeout condition is met)
```

4 Experiments and Comparisons

4.1 Problem Instances and Experimental Protocol

To evaluate the efficiency of our proposed ATS algorithm, we carry out experiments on two different data sets. The first set (4 instances named test1∼test4) was previously used in the literature for the old version of the CCT problem [7].

The second set (7 instances named comp01∼comp07) is from the Second International Timetabling Competition mentioned in the introduction. All these 11 instances can be downloaded from http://tabu.diegm.uniud.it/ctt/index.php.

Our algorithm is programmed in C and compiled using Dev C++ on a PC running Windows XP with 3.44GHz CPU and 2G RAM. To obtain our computational results, our ATS algorithm is run 100 times on each instance with different random seeds. The stop condition is just the timeout required by the ITC–2007 competition rules. On our PC, this corresponds to 390 seconds.

4.2 Comparative Results and Discussion

Table 2 shows the results of the ATS algorithm on the 11 instances of the two data sets for both formulations (see Section 2.2) as well as the previous best known results available in the literature [7,6,14]. For the ATS algorithm, we indicate the following information: the best score f_{min}, the average score f_{ave} and the standard deviation σ over 100 independent runs. For the reference algorithms in [7,6,14], only the best results f_{min} are available. It should be mentioned that the stop conditions of the three reference algorithms are also the timeout condition of the ITC–2007.

The algorithms in [7,6] are developed by the ITC–2007 organizers while that in [14] is the winner solver of the ITC–2007 competition. The algorithms of [7,6] employ a dynamic Tabu Search which allows unfeasible assignments during the problem solving and the neighborhood is the simple one that moves one lecture to a different period and/or a different room. One observes that this neighborhood is just the subset of our basic neighborhood N_1. The algorithm in [14] is composed of a constructive phase, a Hill Climbing algorithm and the Great Deluge technique and uses six specialized moves (neighborhoods).

From table 2, one observes that our ATS algorithm dominates the algorithms of [7,6] for 8 out of 11 instances (in bold). For the 3 remaining cases whose optimum is known, the optimum is reached by the ATS algorithm within several seconds. Moreover, the standard deviations of ATS for all the tested instances are small, showing its robustness.

If we compare ATS and the winner algorithm of the ITC–2007 in [14] on the 7 competition instances (results not available in [14] for the 11 instances of Formulation **I** and the first set of 4 instances of Formulation **II**), one observes that the results of both algorithms are quite comparable. For these 7 instances, ATS reaches better (respectively worse) results than the algorithm in [14] for 2 (respectively 3) instances, with equaling results for the 2 remaining instances. Notice that ATS is ranked the second place for the track 3 of ITC–2007 [2].

Let us mention that we also studied the behavior of the proposed ATS algorithm concerning the influence of the penalty-guided perturbation mechanism (Section 3.3) and the different adaptive mechanisms (Section 3.4). Moreover, a detailed study was conducted to analyze several neighborhoods (including those used in this paper) and the different ways of combining them. This analysis showed that the newly proposed double Kempe chains neighborhood N_4 and its

[2] This result is available at http://www.cs.qub.ac.uk/itc2007/winner/finalorder.htm

Table 2. Computational results and comparison on the 11 instances

Instance	Formulation **I**				Formulation **II**				
	ATS Heuristic			Best in [7, 6]	ATS Heuristic			Best in [6]	Best in [14]
	f_{min}	f_{ave}	σ	f_{min}	f_{min}	f_{ave}	σ	f_{min}	
test1	**212**	212	0	213	**224**	229.5	1.8	234	—
test2	**8**	8	0	**8**	**16**	17.1	1.0	17	—
test3	**35**	35.3	0.3	36	**73**	82.9	4.1	86	—
test4	**28**	32.8	2.1	43	**76**	89.4	5.8	132	—
comp01	**4**	4	0	**4**	**5**	5	0	**5**	**5**
comp02	**22**	24.5	2.3	35	**34**	60.6	7.5	75	43
comp03	**41**	44.8	3.5	52	**70**	86.6	6.3	93	72
comp04	**19**	21.8	2.8	21	38	47.9	4.0	45	**35**
comp05	**224**	229.4	5.1	**244**	**298**	328.5	11.7	326	**298**
comp06	**25**	27.6	3.1	27	47	69.9	7.4	62	**41**
comp07	**4**	6.3	2.4	13	19	28.2	5.6	38	**14**

token-ring combination with the simple neighborhood N_1 contribute greatly to the efficiency of the ATS algorithm.

5 Conclusions

We have provided a mathematical formulation of the curriculum-based course timetabling problem and presented a highly effective hybrid Adaptive Tabu Search algorithm for solving this difficult problem. The effectiveness of the ATS algorithm comes from a number of original features. First, we have introduced the double Kempe chain neighborhood structure for the CCT problem. Second, for our TS procedure, we have devised a combined exploitation strategy of the Kempe chain neighborhood and the basic swap neighborhood. Third, we have proposed a mechanism for adaptively combining TS and perturbation. The computational results on 11 instances on two formulations show that our hybrid ATS algorithm dominates the reference algorithms in [7,6] and competes very well with the winner solver of the ITC–2007 in [14]. Let us comment that most of the ingredients proposed in this paper remain general and would be directly applicable or adapted to other combinatorial problems.

Acknowledgment

This work was partially supported by "*Angers Loire Métropole*" and the Region of "*Pays de la Loire*" within the MILES Project (2006-2009). The authors would like to thank the referees for their helpful comments and questions.

References

1. Burke, E., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper heuristic for timetabling problems. European Journal of Operational Research 176, 177–192 (2007)
2. Casey, S., Thompson, J.: Grasping the examination scheduling problem. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 232–246. Springer, Heidelberg (2003)

3. Chiarandini, M., Birattari, M., Socha, K., Rossi-Doria, O.: An effective hybrid algorithm for university course timetabling. Journal of Scheduling 9, 403–432 (2006)
4. Cooper, T.B., Kingston, J.H.: The complexity of timetable construction problems. In: Burke, E.K., Ross, P. (eds.) PATAT 1995. LNCS, vol. 1153, pp. 283–295. Springer, Heidelberg (1996)
5. Côté, P., Wong, T., Sabourin, R.: Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 151–168. Springer, Heidelberg (2005)
6. De Cesco, F., Di Gaspero, L., Schaerf, A.: Benchmarking Curriculum-Based Course Timetabling: Formulations, Data Formats, Instances, Validation, and Results, University of Udine (2008), `http://tabu.diegm.uniud.it/ctt/DDS2008.pdf`
7. Di Gaspero, L., Schaerf, A.: Multi-neighbourhood local search with application to course timetabling. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 263–278. Springer, Heidelberg (2003)
8. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
9. Lewis, R.: A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30(1), 167–190 (2008)
10. Lourenço, H., Martin, O., Stützle, T.: Iterated local search. Handbook of Metaheuristics. Springer, Berlin (2003)
11. McCollum, B.: A perspective on bridging the gap between theory and practice in university timetabling. In: Burke, E.K., Rudová, H. (eds.) PATAT 2007. LNCS, vol. 3867, pp. 3–23. Springer, Heidelberg (2007)
12. Merlot, L.T.G., Boland, N., Hughes, B.D., et al.: A hybrid algorithm for the examination timetabling problem. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 207–231. Springer, Heidelberg (2003)
13. Misevicius, A., Lenkevicius, A., Rubliauskas, D.: Iterated tabu search: an improvement to standard tabu search. Information Technology and Control 35(3), 187–197 (2006)
14. Müller, T.: ITC2007: Solver Description, Technical Report, Purdue University (2008), `http://www.unitime.org/papers/itc2007.pdf`
15. Rossi-Doria, O., Paechter, B., Blum, C., Socha, K., Samples, M.: A local search for the timetabling problem. In: Proceedings of the 4th PATAT, pp. 124–127 (2002)
16. Sedgewick, R.: Algorithms, 2nd edn. Addison-Wesley, Reading (1988)