Discrete Applied Mathematics 157 (2009) 1364–1378

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Polynomial reduction of time–space scheduling to time scheduling[☆]

J. Studenovský [*]

Institute of Informatics, P.J. Šafárik University, Jesenná 5, 04154 Košice, Slovakia

ARTICLE INFO

Article history:
Received 25 August 2006
Received in revised form 20 October 2008
Accepted 24 October 2008
Available online 6 December 2008

Keywords:
University course timetabling problem
Polynomial reduction
Computational complexity
Timetabling
Scheduling

ABSTRACT

We study the University Course Timetabling Problem (UCTP). In particular we deal with the following question: is it possible to decompose UCTP into two problems, namely, (i) a time scheduling, and (ii) a space scheduling. We have arguments that it is not possible. Therefore we study UCTP with the assumption that each room belongs to exactly one type of room. A type of room is a set of rooms, which have similar properties. We prove that in this case UCTP is polynomially reducible to time scheduling. Hence we solve UCTP with the following method: at first we solve time scheduling and subsequently we solve space scheduling with a polynomial $O(n^3)$ algorithm. In this way we obtain a radical (exponential) speed-up of algorithms for UCTP. The method was applied at P.J. Šafárik University.

1. Introduction

A mathematical model for School Timetabling can be found in [12]. The University Course Timetabling Problem (UCTP) is a 2-dimensional generalization of School Timetabling. The integer programming model for UCTP has been investigated in [9,10].

According to [18], a large number of works about UCTP arises out of the Metaheuristics Network – a European Commission project undertaken jointly by five European institutes – which seeks to compare metaheuristics on different combinatorial optimization problems. In one phase of the four-year project, UCTP was considered. In the course of the Metaheuristics Network, several metaheuristics were evaluated and compared on instances of a certain reduction of UCTP [16,18]. The metaheuristics evaluated included: Genetic Algorithm [13], Simulated Annealing [2], Tabu Search [4], Iterated Local Search [7], Ant Colony Optimization [19] and MAX–MIN Ant System [18]. The mentioned papers also present experiments that were performed in order to evaluate the algorithm's performance. Problem instances used in experiments have been proposed as a part of the International Timetabling Competition. A development of metaheuristics continues also after completion of the project Metaheuristics Network [5,6,8,14,15,17].

It is difficult to provide a uniform and generic definition of UCTP. Due to the fact that a course organization as well as additional preferences may vary from case to case, the number and type of soft and hard constraints changes. Hence, the algorithmic solutions proposed for this problem usually concentrate on a particular subproblem.

For the purpose of evaluating metaheuristics in the course of the Metaheuristics Network, the following reduction of UCTP has been defined [16,18]. The problem consists of a set of n events $E = \{e_1, e_2, \dots, e_n\}$ to be scheduled in a set of k timeslots $T = \{\tau_1, \tau_2, \dots, \tau_k\}$ ($k = 45$, 5 days of 9 h each), a set of rooms R in which events can take place (rooms are of a certain capacity), a set of students S who attend the events, and a set of features F satisfied by rooms and required by events.

[☆] This work was supported by the Slovak Grant Agency for Science (VEGA) under contract #1/3128/06 "Combinatorial Structures and Complexity of Algorithms"

[*] Tel.: +421 55 2346 177.
E-mail address: jozef.studenovsky@upjs.sk.

Each student is already preassigned to a subset of events. Each event takes exactly one timeslot. All students and all rooms are available in each timeslot. A feasible timetable is a mapping which assigns to each event $e \in E$ a timeslot $t_e \in T$ and a room $r_e \in R$ so that the following hard constraints are satisfied:

- no student attends more than one event at the same time;
- only one event takes place in each room at a given time;
- the room is big enough for all the attending students and satisfies all the features required by the event.

The objective is to minimize the number of the following soft constraint violations in a feasible timetable:

- a student has a course in the last slot of the day;
- a student has more than two consecutive courses;
- a student has exactly one course during a day.

In the present paper we show how real UCTP differs from the reduction of UCTP defined above. In fact, the following conditions are also to be taken into account.

(1) The problem includes also a set of teachers U who attend the events. Each teacher is already preassigned to a subset of events.
(2) Universities use own formalisms for features F satisfied by rooms and required by events. A very simple formalism is that each event $e \in E$ has a preassigned subset of allowed rooms R_e (i.e. we require $r_e \in R_e$).
(3) Events need not take exactly one timeslot. Each event $e \in E$ has a preassigned natural number d_e. The event e takes d_e timeslots.
(4) Students and teachers are not available in each timeslot. Hence each event $e \in E$ has a preassigned subset of timeslots T_e, which are allowed for a timeslot t_e (i.e. we require $t_e \in T_e$).

A feasible timetable (schedule) in a real UCTP is a mapping Υ which assigns to each event $e \in E$ a timeslot $t_e \in T_e$ and a room $r_e \in R_e$ so that the following hard constraints are satisfied:

- no student attends more than one event at the same time;
- no teacher attends more than one event at the same time;
- only one event takes place in each room at a given time.

The schedule Υ has two components: a time schedule $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ and a space schedule $R^\Upsilon = (r_{e_1}, r_{e_2}, \dots, r_{e_n})$.

In view of [11,1], a very primitive version of the School Timetabling is NP-hard. This explains the fact that for UCTP only exponential algorithms are known.

We want to decompose UCTP into two problems:

(1) A time scheduling, whose solution is a time schedule T^Υ.
(2) A space scheduling, whose solution is a space schedule R^Υ.

We study the following questions:

(1) Is it possible to solve UCTP in the way that we solve the first problem and subsequently the second problem?
(2) Is it true that some of these problems has a polynomial algorithm?

We have arguments that the answers to both questions are negative. Therefore we study UCTP, for which the following assumption

$$\text{for all } i, j \in E, \text{ either } R_i = R_j \text{ or } R_i \cap R_j = \emptyset \tag{1}$$

is satisfied. We prove that in this case, the answers to both questions are positive. We found a polynomial $O(n^3)$ algorithm for the second problem. These results are missing in the literature. The first problem is NP-hard.

We can also consider the question when the assumption (1) is satisfied. Suppose that a university uses some system of types of rooms. We denote this system by R'. A type of room is a set of rooms, which have similar properties. We prove that the assumption (1) is satisfied if and only if the system R' is a partition of the set of the rooms R. In this case we say that we deal with UCTP1.

Our method can be characterized as follows.

(1) We define a vector T^Υ to be a time schedule if there exists a vector R^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a schedule for UCTP. Further, we define a time scheduling as the problem to find a time schedule. Analogously, we define a space schedule R^Υ and a space scheduling problem. A special space scheduling is defined to be the problem to find a space schedule R^Υ for a given time schedule T^Υ.
(2) We prove that UCTP1 is polynomially reducible to time scheduling. Hence we solve UCTP1 with the following method: at first we solve the time scheduling and subsequently we solve the special space scheduling with a polynomial $O(n^3)$ algorithm. In this way we obtain a radical (exponential) speed-up of algorithms for UCTP.

The organization of the paper is as follows. Section 2 defines a time–space schedule for UCTP. In Section 3 we prove that UCTP is UCTP1 if and only if the assumption (1) is satisfied. In the following sections we study UCTP1. Section 4 describes a time schedule. In Section 5 we study interchanges of blocks in a time–space schedule. Section 6 shows that we can use interchanges of blocks for improving the quality of a time–space schedule. We present an algorithm which improves the quality of a time–space schedule so that there exists a free room of a desired type and in a desired time. This algorithm is applied in Section 7, where we present a polynomial $O(n^3)$ algorithm for the special space scheduling. Section 8 describes necessary and sufficient conditions for a time schedule. In Section 9 we prove that UCTP1 is polynomially reducible to time scheduling. We present a theorem saying that it is possible to decompose UCTP1 into two problems, namely, (i) time scheduling, and (ii) special space scheduling. Also, we present arguments for the validity of the following hypothesis: if assumption (1) is not satisfied, then it is not possible to decompose UCTP into two problems with the properties as above. In Section 10 we study the complexity of UCTP. Section 11 considers the question when the assumption (1) is satisfied in practice. Section 12 concludes the paper.

2. UCTP—University Course Timetabling Problem

We define the University Course Timetabling Problem (UCTP) as follows.

Notation 1. We will apply the following notation:

U— a set of teachers,
S— a set of students,
T— a linearly ordered set of timeslots, $T = \{\tau_1, \tau_2, \dots, \tau_k\}$, $(\tau_i < \tau_j$ if $i < j)$,
R— a set of rooms, $|R| = g$,
E— a set of events, $|E| = n$, $E = \{e_1, e_2, \dots, e_n\}$,
I— a set of indices, $I = \{1, 2, \dots, n\}$.

As usual, $\mathbb{N}$ denotes the set of all positive integers.

Definition 1. An *event* $e \in E$ is an ordered quintuple $(U_e, S_e, T_e, R_e, d_e)$, where $U_e \subseteq U$, $S_e \subseteq S$, $T_e \subseteq T$, $R_e \subseteq R$, $d_e \in \mathbb{N}$ and $1 \le d_e \le k$.

The teachers of the set U_e and the students of the set S_e attend the event $e \in E$. The event e takes d_e timeslots.

Definition 2. A mapping which assigns to each event $e \in E$ a pair (t_e, r_e), where $t_e \in T_e$ and $r_e \in R_e$, is called a *Υ-mapping*. The value t_e is called the *timeslot of the event* e and r_e the *room of the event* e. Denote $\Upsilon = (T^\Upsilon, R^\Upsilon) = (t_{e_i}, r_{e_i})_{i=1}^n$. The vector $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ is said to be the *time component* and the vector $R^\Upsilon = (r_{e_1}, r_{e_2}, \dots, r_{e_n})$ is said to be the *space component* of the Υ-mapping.

We interpret the Υ-mapping in the way that the event $e \in E$ starts in the timeslot t_e and takes places in the room r_e.

Notation 2. For the timeslot $\tau \in T$, $\tau = \tau_j$ and the integer d, $1 \le j + d \le k$ denote $\Theta(\tau, d) = \tau_{j+d}$.

Notation 3. For the event $e \in E$ denote $H_e = \{t_e, \Theta(t_e, 1), \dots, \Theta(t_e, d_e - 1)\} = \{\tau_j, \tau_{j+1}, \dots, \tau_{j+d_e-1}\}$, where $\tau_j = t_e$.

The set H_e is the set of timeslots in which the event $e \in E$ is realized. The timeslot t_e is *start timeslot* and the timeslot $\Theta(t_e, d_e - 1)$ is *end timeslot* of the event e.

Remark 3. We require $H_e \subseteq T$ for each event $e \in E$. Therefore we assume that the set T_e satisfies the condition $\Theta(t, d_e - 1) \in T$ for each timeslot $t \in T_e$ and each event $e \in E$.

Definition 4. If in the Υ-mapping for a teacher $u \in U$ there exist events $i, j \in E$, $i \ne j$ such that $u \in U_i \cap U_j$ and $H_i \cap H_j \ne \emptyset$, then we say that the teacher u *has a conflict*. Analogously we define a *conflict for a student* $s \in S$. If for a room $r \in R$ there exist events $i, j \in E$, $i \ne j$ such that $r = r_i = r_j$ and $H_i \cap H_j \ne \emptyset$, then we say that the room r *has a conflict*.

Definition 5. If the Υ-mapping contains no conflict, then it is called a *time–space schedule of the set of events* E.

Definition 6. The problem to find a time–space schedule of a given set of events E is called a *time–space scheduling* or also a *University Course Timetabling Problem* (UCTP, for short).

Notation 4. We denote $\eta = max\{n, k, g\}$. In a real UCTP, we always have $\eta = n$.

We remark that the notion introduced in Definition 5 can be alternatively defined as follows.

Definition 7. Let $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$, $R^\Upsilon = (r_{e_1}, r_{e_2}, \dots, r_{e_n})$. The pair (T^Υ, R^Υ) is a *time–space schedule of the set of events* E if the following conditions are satisfied:

(1) $t_e \in T_e$ for all $e \in E$,
(2) $r_e \in R_e$ for all $e \in E$,
(3) if $U_i \cap U_j \ne \emptyset$ or $S_i \cap S_j \ne \emptyset$ or $r_i = r_j$, then $H_i \cap H_j = \emptyset$, for all $i, j \in E$, $i \ne j$.

Table 1
The set of events.

e	U_e	S_e	T_e	R_e	d_e
e_1	u_1	s_2, s_3	τ_1	a_1	3
e_2	u_1, u_3	s_2, s_3, s_5, s_6, s_7	τ_5, τ_6	a_2, a_3	2
e_3	u_2	s_1, s_8, s_9	τ_1, τ_3	a_2, a_3	3
e_4	u_1	$s_2, s_3, s_4, s_5, s_6, s_7$	τ_2, τ_4	a_1	2
e_5	u_2	s_1, s_8, s_9	τ_3, τ_4	a_2, a_3	3
e_6	u_3	s_4, s_5, s_6, s_7	τ_1, τ_2	a_2, a_3	3
e_7	u_2	s_1, s_4, s_8, s_9	τ_1, τ_3, τ_7	a_2, a_3	1

Table 2
The timetables of teachers, students and rooms.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
u_1	e_1	e_1	e_1	e_4	e_4	e_2	e_2
u_2	e_3	e_3	e_3	e_5	e_5	e_5	e_7
u_3	e_6	e_6	e_6			e_2	e_2
s_1	e_3	e_3	e_3	e_5	e_5	e_5	e_7
s_2	e_1	e_1	e_1	e_4	e_4	e_2	e_2
s_3	e_1	e_1	e_1	e_4	e_4	e_2	e_2
s_4	e_6	e_6	e_6	e_4	e_4		e_7
s_5	e_6	e_6	e_6	e_4	e_4	e_2	e_2
s_6	e_6	e_6	e_6	e_4	e_4	e_2	e_2
s_7	e_6	e_6	e_6	e_4	e_4	e_2	e_2
s_8	e_3	e_3	e_3	e_5	e_5	e_5	e_7
s_9	e_3	e_3	e_3	e_5	e_5	e_5	e_7
a_1	e_1	e_1	e_1	e_4	e_4		
a_2	e_3	e_3	e_3			e_2	e_2
a_3	e_6	e_6	e_6	e_5	e_5	e_5	e_7

In fact, the first and the second conditions of Definition 7 are equivalent to the fact that the mapping which assigns a pair (t_e, r_e) to each event $e \in E$ is a Υ-mapping. The third condition of Definition 7 is equivalent to the condition that this Υ-mapping does not contain any conflict.

The following assertion is obvious.

Lemma 8. *Definitions* 5 and 7 *are equivalent.*

Example 9. Let $U = \{u_1, u_2, u_3\}$, $S = \{s_1, s_2, \dots, s_9\}$, $T = \{\tau_1, \tau_2, \dots, \tau_7\}$, $R = \{a_1, a_2, a_3\}$ be sets of teachers, students, timeslots and rooms, respectively. The set of events $E = \{e_1, e_2, \dots, e_7\}$ is given by Table 1.

Let $T^\Upsilon = (\tau_1, \tau_6, \tau_1, \tau_4, \tau_4, \tau_1, \tau_7)$ and $R^\Upsilon = (a_1, a_2, a_2, a_1, a_3, a_3, a_3)$. The pair (T^Υ, R^Υ) satisfies all conditions of Definition 7. Hence the pair (T^Υ, R^Υ) is a time–space schedule of the set of events E. The timetables of teachers, students and rooms are in Table 2.

3. UCTP1

A type of room is a set of rooms which have similar properties. In this context the set R_e for $e \in E$ is also a type of room.

Definition 10. We consider a system of subsets of the set of rooms $R' = \{R'_1, R'_2, \dots, R'_\mu\}$ which satisfies the following conditions:

(1) for each $e \in E$ there exists exactly one number $y_e \in M = \{1, 2, \dots, \mu\}$ such that $R_e = R'_{y_e}$,
(2) for each $m \in M$ there exists $e \in E$ such that $R'_m = R_e$.

We say that R' is a *system of types of rooms*.

From this definition it follows that all elements of the system R' are pairwise distinct and $\mu \le n$.

For UCTP from Example 9 we have $\mu = 2$, $M = \{1, 2\}$, $R'_1 = \{a_1\}$, $R'_2 = \{a_2, a_3\}$, $R' = \{R'_1, R'_2\}$, and $y_{e_1} = 1$, $y_{e_2} = 2$, $y_{e_3} = 2$, $y_{e_4} = 1$, $y_{e_5} = 2$, $y_{e_6} = 2$, $y_{e_7} = 2$.

Algorithm 1. *System of types of rooms*
Input: the system $R^\diamond = \{R_{e_1}, R_{e_2}, \dots, R_{e_n}\}$
Output: $R' = \{R'_1, R'_2, \dots, R'_\mu\}$ *and the function* $y = (y_{e_1}, y_{e_2}, \dots, y_{e_n})$
Method:
First we set $R' = \emptyset$, $\mu = 0$. For $i = 1, 2, \dots, n$ check whether R_{e_i} is equal to some $R'_m \in R'$. In such case we set $y_{e_i} = m$. In the opposite case we place R_{e_i} at the tail of the system R' and we set $y_{e_i} = \mu$.

```
procedure SYSTEM($R^\diamond$, $R'$, $y$);
begin
    $\mu := 0$;
    for $i := 1$ to $n$ do begin
        $m := 0$;
        $equal :=$ false;
        while not $equal$ and $m < \mu$ do begin
            $m := m + 1$;
            if $R_{e_i} = R'_m$ then begin
                $y_{e_i} := m$;
                $equal :=$ true
            end
        end;
        if not $equal$ then begin
            $\mu := \mu + 1$;
            $R'_\mu := R_{e_i}$;
            $y_{e_i} := \mu$
        end
    end
end;
```

Lemma 11. *Algorithm 1 computes the system R' and the function y in a time $O(\eta^4)$.*

Proof. The proof of the correctness of the algorithm is trivial. By Definition 10, $\mu \le n$. By Notation 4, $n \le \eta$ and $g \le \eta$. The test $R_{e_i} = R'_m$ requires a time $O(\eta^2)$. Hence the loop **while** requires a time $O(\eta^3)$. The command **if not** requires a time $O(\eta)$. The algorithm performs n steps in the loop **for**. Each step requires a time $O(\eta^3)$. Consequently, the algorithm requires a time $O(\eta^4)$. $\square$

Algorithm 1 computes the system R' at the beginning of an algorithm for UCTP. Thus E, n, k, g, μ, R', y are global variables. This means that they are disposable in each procedure presented in this paper.

Assumption 12. For all $i, j \in E$, either $R_i = R_j$ or $R_i \cap R_j = \emptyset$.

Lemma 13. *Assumption 12 is satisfied if and only if $R'_m \cap R'_l = \emptyset$ for all $m, l \in M$, $m \neq l$.*

Proof. *Necessity.* Suppose that Assumption 12 is satisfied and that the condition of the lemma is not valid. Then there exist $m, l \in M$, $m \neq l$ such that $R'_m \cap R'_l \neq \emptyset$. From the definition of the system R' it follows that there exist $i, j \in E$ such that $R'_m = R_i$ and $R'_l = R_j$. Since the elements of the system R' are pairwise distinct and $m \neq l$ it must be $R'_m \neq R'_l$ and then also $R_i \neq R_j$. From $R'_m \cap R'_l \neq \emptyset$ we get $R_i \cap R_j \neq \emptyset$. Hence $R_i \neq R_j$ and $R_i \cap R_j \neq \emptyset$, which is a contradiction.

Sufficiency. Suppose that the condition $R'_m \cap R'_l = \emptyset$ for all $m, l \in M$, $m \neq l$ is satisfied and that Assumption 12 does not hold. Then there exist $i, j \in E$ such that $R_i \neq R_j$ and $R_i \cap R_j \neq \emptyset$. From the definition of the system R' it follows that there exist $m, l \in M$ such that $R'_m = R_i$ and $R'_l = R_j$. From $R_i \neq R_j$ we get $R'_m \neq R'_l$. Therefore $m \neq l$. From $R_i \cap R_j \neq \emptyset$ we obtain $R'_m \cap R'_l \neq \emptyset$, which is a contradiction. $\square$

Lemma 14. $R'_1 \cup R'_2 \cup \cdots \cup R'_\mu = R$.

Proof. We denote $R^\cup = R_{e_1} \cup R_{e_2} \cup \cdots \cup R_{e_n}$. Obviously, $R^\cup \subseteq R$. If $R^\cup \neq R$, then there exists a room $r \in R - R^\cup$. If we remove the room r from the set R, then we get an equivalent UCTP. Thus without loss of generality, we can assume that $R^\cup = R$. The definition of the system R' implies that $R'_1 \cup R'_2 \cup \cdots \cup R'_\mu = R^\cup$. Hence $R'_1 \cup R'_2 \cup \cdots \cup R'_\mu = R$. $\square$

Corollary 15. *Assumption 12 is satisfied if and only if the system R' is a partition of the set R.*

Corollary 16. *Assumption 12 is satisfied if and only if for each event e and each $r \in R_e$ there exists exactly one R'_m such that $r \in R'_m$, where $m = y_e$.*

Corollary 17. *Assumption 12 is satisfied if and only if each room belongs to exactly one type of room.*

Definition 18. University Course Timetabling Problem which satisfies Assumption 12 will be called *UCTP1*.

4. Time schedule

In this section we define time schedule and time scheduling. In Theorem 23 we present the main property of a time schedule.

Definition 19. A vector T^Υ is called a *time schedule of the set of events E*, if there exists a vector R^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a time–space schedule of the set of events E.

A natural question arises how to test whether a given vector T^Υ is a time schedule. We distinguish two cases.

(1) Assumption 12 is not satisfied.
We write an exponential algorithm (cf. Proposition 49), which returns an answer whether there exists a vector R^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a time–space schedule. Clearly, this algorithm also constructs R^Υ (if the answer is positive). The algorithm must test whether $\Upsilon = (T^\Upsilon, R^\Upsilon)$ satisfies the conditions of Definition 7 which are dependent on R^Υ. Summarizing, this method constructs R^Υ, is dependent on R^Υ and requires an exponential time.

(2) Assumption 12 is satisfied.
We find necessary and sufficient conditions for the vector T^Υ to be a time schedule (cf. Theorem 42). These conditions are independent on R^Υ. The test whether the given vector T^Υ satisfies the conditions requires a time $O(n^4)$. Summarizing, this method does not construct R^Υ, is independent on R^Υ and requires a polynomial time.

Definition 20. The problem to find a time schedule of a given set of events E is called a *time scheduling*.

Definition 21. Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^n$ be a time–space schedule of the set of events E and $r \in R$, $t \in T$. We say that the room r is *free* in the timeslot t in the schedule Υ if there does not exist any event $e \in E$ such that $r_e = r$ and $t \in H_e$.

Definition 22. Let $\Upsilon = (T^\Upsilon, R^\Upsilon)$ be a time–space schedule of the set of events E. The symbol Q will denote the matrix of type $\mu \times k$, in which for $m \in M$ and $t \in T$ the element Q_{mt} determines the number of rooms of the set R'_m free at the timeslot t in the schedule Υ. The matrix Q is called the *matrix of numbers of free rooms* for the schedule Υ.

To the computation of the matrix Q both components T^Υ and R^Υ of the schedule Υ are necessary. Further, we will prove that if Assumption 12 is fulfilled, then to the computation of the matrix Q it is sufficient to know the time component T^Υ of the schedule Υ.

Algorithm 2. *Matrix of numbers of free rooms*
Input: $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$, *where* $t_e \in T_e$ *for all* $e \in E$.
Output: Q
Assumption: Assumption 12 *is satisfied.*
Method:
At first we set $Q_{mt} = |R'_m|$ for all $m \in M$ and $t \in T$, which corresponds to the empty time–space schedule. Next, for each event $e \in E$ we decrement Q_{mt} for all $t \in H_e$, where $m = y_e$.

```
procedure MATRIX($T^\Upsilon$, $Q$);
begin
    for $m := 1$ to $\mu$ do
        for $t := \tau_1$ to $\tau_k$ do
            $Q_{mt} := |R'_m|$;
    for $i := 1$ to $n$ do begin
        $m := y_{e_i}$;
        for all $t \in H_{e_i}$ do
            $Q_{mt} := Q_{mt} - 1$
    end
end;
```

Theorem 23. *Let Assumption 12 be satisfied. If T^Υ is a time schedule of the set of events E, then Algorithm 2 for the input T^Υ computes the matrix of numbers of free rooms Q for each time–space schedule $\Upsilon = (T^\Upsilon, R^\Upsilon)$ of the set of events E with the time component T^Υ. Algorithm 2 requires a time $O(\eta^2)$.*

Proof. Let $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ be the time schedule of the set of events E. By Definition 19, there exists a vector $R^\Upsilon = (r_{e_1}, r_{e_2}, \dots, r_{e_n})$ such that $\Upsilon = (T^\Upsilon, R^\Upsilon) = (t_{e_j}, r_{e_j})_{j=1}^n$ is the time–space schedule of the set of events E. Obviously, for all $i \in I$, $\Upsilon^i = (t_{e_j}, r_{e_j})_{j=1}^i$ is the time–space schedule of the set of events $E^i = \{e_1, e_2, \dots, e_i\}$. Let Υ^0 be the empty time–space schedule. This means that Υ^0 is the time–space schedule of the empty set of events.

We denote by Q^i a state of the matrix Q after i steps of the algorithm for the input T^Υ (more precisely, after i steps of the loop **for** i). We proceed by induction.

The induction hypothesis is that after $i \le n$ steps, the algorithm computes Q^i, where Q^i is the matrix of numbers of free rooms for the time–space schedule Υ^i.

Basis. At the 0-th step (before the loop **for** i) the algorithm computes the matrix Q^0, where $Q^0_{mt} = |R'_m|$ for all $(m, t) \in M \times T$. Obviously, Q^0 is the matrix of numbers of free rooms for the time–space schedule Υ^0. Hence the induction hypothesis is true for $i = 0$.

Inductive step. Assume that after $i - 1 < n$ steps, the algorithm computes Q^{i-1}. The event e_i occupies the room $r_{e_i} \in R_{e_i}$ for all timeslots $t \in H_{e_i}$ in the time–space schedule Υ^i. According to Corollary 16, the room r_{e_i} belongs exactly to one set R'_m, where $m = y_{e_i}$. Hence the following equations hold:

$$\begin{aligned} Q^i_{mt} &= Q^{i-1}_{mt} - 1 \quad \text{for } (m,t) \in \{y_{e_i}\} \times H_{e_i} \\ Q^i_{mt} &= Q^{i-1}_{mt} \quad \text{for } (m,t) \in M \times T - \{y_{e_i}\} \times H_{e_i}. \end{aligned} \tag{2}$$

These equations are applied at the i-th step of the algorithm. Therefore after i steps, the algorithm computes Q^i.

From the induction principle it follows that after n steps the algorithm computes Q^n. Clearly, $Q^n = Q$.

By Definition 10, $\mu \le n$. By Notation 4, $n \le \eta$ and $k \le \eta$. The matrix Q has μk elements. Hence at the 0-th step, the algorithm computes the matrix Q^0 in a time $O(\eta^2)$. By Definition 1, $d_{e_i} \le k$. Therefore $|H_{e_i}| = d_{e_i} \le k \le \eta$. Consequently, at the i-th step, the algorithm decrements d_{e_i} elements of the matrix Q in a time $O(\eta)$. Hence the algorithm performs n steps in the loop **for** i in the time $O(\eta^2)$. It follows that the algorithm requires the time $O(\eta^2)$. □

The matrix Q for the time–space schedule Υ from Example 9 is as follows:

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

5. Interchanges of blocks in a schedule for UCTP1

If we construct a schedule then we can interchange blocks for improving its quality. We apply the following method.

Let $\tau, \varrho \in T$. As usual, the interval $[\tau, \varrho]$ is the set of all $t \in T$ such that $\tau \le t \le \varrho$.

Definition 24. Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^n$ be a time–space schedule of the set of events E, and $a \in R$, $\tau \in T$. We denote:

(1) $START(a, \tau)$—is the set of events $e \in E$ with a room $r_e = a$ and with a start timeslot $t_e \in [\tau, \tau_k]$.
(2) $END(a, \tau)$—is the set of events $e \in E$ with a room $r_e = a$ and with an end timeslot $\Theta(t_e, d_e - 1) \in [\tau, \tau_k]$.

Lemma 25. *For $a \in R$ and $\tau \in T$, the relation $START(a, \tau) \subseteq END(a, \tau)$ is valid.*

Proof. Let $e \in START(a, \tau)$. Then $r_e = a$ and $\tau \le t_e \le \tau_k$. By Definition 1, $d_e \ge 1$. This implies $\tau \le \Theta(t_e, d_e - 1)$. According to Remark 3, $\Theta(t_e, d_e - 1) \le \tau_k$. Therefore $\Theta(t_e, d_e - 1) \in [\tau, \tau_k]$. Consequently, $e \in END(a, \tau)$. □

Definition 26. If $START(a, \tau) = END(a, \tau)$, then the set $START(a, \tau)$ is called a *block*.

For the time–space schedule Υ from Example 9 we get that $START(a_2, \tau_4) = \{e_2\}$ is a block, $START(a_3, \tau_4) = \{e_5, e_7\}$ is a block and $START(a_3, \tau_3) = \{e_5, e_7\}$ fails to be a block because $END(a_3, \tau_3) = \{e_6, e_5, e_7\}$.

Lemma 27. *Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^n$ be a time–space schedule of the set of events E. If $START(a, \tau)$ is a block then $H_e \subseteq [\tau, \tau_k]$ for each event $e \in START(a, \tau)$.*

Proof. Let $START(a, \tau)$ be a block and $e \in START(a, \tau)$. By Definition 26, $START(a, \tau) = END(a, \tau)$. Consequently, $e \in END(a, \tau)$. Hence $\Theta(t_e, d_e - 1) \in [\tau, \tau_k]$. From $e \in START(a, \tau)$ it follows $t_e \in [\tau, \tau_k]$. Therefore $H_e \subseteq [\tau, \tau_k]$. □

Lemma 28. *Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^n$ be a time–space schedule of the set of events E. If a room a is free in a timeslot τ then $START(a, \tau)$ is a block.*

Proof. Let a room a be free in a timeslot τ in the time–space schedule Υ. By way of contradiction, assume that $START(a, \tau)$ fails to be a block. Then by Definition 26, $START(a, \tau) \ne END(a, \tau)$ and by Lemma 25, $START(a, \tau) \subset END(a, \tau)$. Consequently, there exists an event $e \in END(a, \tau) - START(a, \tau)$. Therefore $r_e = a$, $\tau \le \Theta(t_e, d_e - 1)$ and $t_e < \tau$. This yields $\tau \in H_e$. Consequently, the event e occupies the room $r_e = a$ in the timeslot τ. This is a contradiction with the assumption that the room a is free in the timeslot τ. □

Lemma 29. *Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^n$ be a time–space schedule of the set of events E and let $\tau \in T$. If a room a is free in a timeslot $\Theta(\tau, -1)$ then $START(a, \tau)$ is a block.*

Proof. Let a room a be free in a timeslot $\Theta(\tau, -1)$ in the time–space schedule Υ. By way of contradiction, assume that $START(a, \tau)$ fails to be a block. Then by Definition 26, $START(a, \tau) \neq END(a, \tau)$ and by Lemma 25, $START(a, \tau) \subset END(a, \tau)$. Consequently, there exists an event $e \in END(a, \tau) - START(a, \tau)$. Therefore $r_e = a, \tau \leq \Theta(t_e, d_e - 1)$ and $t_e < \tau$. This yields $\Theta(\tau, -1) \in H_e$. Consequently, the event e occupies the room $r_e = a$ in the timeslot $\Theta(\tau, -1)$. This is a contradiction with the assumption that the room a is free in the timeslot $\Theta(\tau, -1)$. □

Definition 30. Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^{n}$ be a time–space schedule of the set of events E, and $a_1, a_2 \in R'_m$, $m \in M$, $\tau \in T$. Further, assume that $START(a_1, \tau)$ and $START(a_2, \tau)$ are blocks. The following process is called an *interchange of the block $START(a_1, \tau)$ with the block $START(a_2, \tau)$*:

(1) assign $r'_e = r_e$ for each $e \in E - START(a_1, \tau) - START(a_2, \tau)$,
(2) assign $r'_e = a_2$ for each $e \in START(a_1, \tau)$,
(3) assign $r'_e = a_1$ for each $e \in START(a_2, \tau)$.

Algorithm 3. *Interchange of blocks*
Input: $a_1, a_2, \tau, n, \Upsilon = (t_{e_i}, r_{e_i})_{i=1}^{n}$
Output: $\Upsilon' = (t_{e_i}, r'_{e_i})_{i=1}^{n}$
Assumptions: the assumptions are presented in Definition 30.
Method: the computation by Definition 30.

```
procedure INTERCHANGE($a_1, a_2, \tau, n, \Upsilon, \Upsilon'$);
begin
    for $i := 1$ to $n$ do begin
        $r'_{e_i} := r_{e_i}$;
        if $r_{e_i} = a_1$ and $t_{e_i} \geq \tau$ then
            $r'_{e_i} := a_2$;
        if $r_{e_i} = a_2$ and $t_{e_i} \geq \tau$ then
            $r'_{e_i} := a_1$;
    end
end;
```

In the remaining part of the present section we write Algorithm in stead of Algorithm 3.

For example, let $\Upsilon = (T^{\Upsilon}, R^{\Upsilon})$ be the time–space schedule from Example 9, $a_2, a_3 \in R'_2$, $\tau_4 \in T$. We interchange the block $START(a_2, \tau_4)$ with the block $START(a_3, \tau_4)$. We obtain a new time–space schedule $\Upsilon' = (T^{\Upsilon'}, R^{\Upsilon'})$ with the time component $T^{\Upsilon'} = T^{\Upsilon}$ and with the new space component $R^{\Upsilon'} = (a_1, a_3, a_2, a_1, a_2, a_3, a_2)$.

Theorem 31. *Let Assumption 12 be satisfied. Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^{n}$ be a time–space schedule of the set of events E, and $a_1, a_2 \in R'_m$, $m \in M$, $\tau \in T$. Suppose that $START(a_1, \tau)$ and $START(a_2, \tau)$ are blocks. Algorithm interchanges the block $START(a_1, \tau)$ with the block $START(a_2, \tau)$ in the linear time $O(\eta)$. Its output $\Upsilon' = (t_{e_i}, r'_{e_i})_{i=1}^{n}$ is a time–space schedule of the set of events E.*

Proof. The Algorithm computes in accordance with Definition 30. Hence the Algorithm interchanges the block $START(a_1, \tau)$ with the block $START(a_2, \tau)$. It performs n steps in the loop **for**. Every step requires a constant time. By Notation 4, $n \leq \eta$. Hence the time of the Algorithm is $O(\eta)$.

Let $\Upsilon = (t_{e_i}, r_{e_i})_{i=1}^{n}$ be the time–space schedule of the set of events E and $\Upsilon' = (t_{e_i}, r'_{e_i})_{i=1}^{n}$ be the output of the Algorithm. We prove that Υ' is a time–space schedule of the set of events E.

If $a_1 = a_2$ then from the Algorithm it follows $\Upsilon' = \Upsilon$. Consequently, Υ' is a time–space schedule of the set of events E. Next we assume that $a_1 \neq a_2$.

The first condition of Definition 7, i.e., the condition $t_e \in T_e$ for all $e \in E$, is valid in the time–space schedule Υ. The Algorithm does not change the time component of the time–space schedule Υ. Hence this condition is valid also in Υ'. Consequently, Υ' satisfies the first condition of Definition 7.

Further, by Definition 7, the condition $r_e \in R_e$ for all $e \in E$ is valid in the time–space schedule Υ. By the assumption, we have $a_1, a_2 \in R'_m$, $m \in M$. From $a_1 \neq a_2$ we obtain $START(a_1, \tau) \cap START(a_2, \tau) = \emptyset$. We denote $E^{\cup} = START(a_1, \tau) \cup START(a_2, \tau)$. We distinguish three cases.

(1) Let $e \in E - E^{\cup}$.
The Algorithm gives $r'_e = r_e$. Consequently, $r'_e = r_e \in R_e$. Hence $r'_e \in R_e$.
(2) Let $e \in START(a_1, \tau)$.
Then by Definition 24, $r_e = a_1$ and $t_e \geq \tau$. The Algorithm gives $r'_e = a_2$. From the assumption $a_1 \in R'_m$ and from the fact that $a_1 = r_e \in R_e$, by using Corollary 16 we get $R_e = R'_m$. Consequently, $r'_e = a_2 \in R'_m = R_e$. Hence $r'_e \in R_e$.
(3) Let $e \in START(a_2, \tau)$.
Then by Definition 24, $r_e = a_2$ and $t_e \geq \tau$. The Algorithm gives $r'_e = a_1$. The proof of the fact that $r'_e \in R_e$ can be performed analogously as in the case (2).

Summarizing, we obtain $r'_e \in R_e$ for all $e \in E$. Consequently, Υ' satisfies the second condition of Definition 7.

Now we prove that Υ' contains no conflict. By way of contradiction, assume that there exists a conflict in Υ'. First we assume that there exists a conflict in some room r in Υ'. It means that there exists a room r and events $i, j \in E$, $i \neq j$ such that $r = r'_i = r'_j$ and $H_i \cap H_j \neq \emptyset$. We distinguish nine cases.

(1) Let $i, j \in E - E^{\cup}$.
Then $r'_i = r_i$, $r'_j = r_j$ and $r = r'_i = r'_j$. This yields $r = r_i = r_j$ and $H_i \cap H_j \neq \emptyset$. It means that there is a conflict in the room r in Υ. This is a contradiction with the assumption that Υ is the time–space schedule.
(2) Let $i, j \in START(a_1, \tau)$.
Then $r_i = a_1$, $r_j = a_1$. We obtain $a_1 = r_i = r_j$ and $H_i \cap H_j \neq \emptyset$. It means that there is a conflict in the room a_1 in Υ. It is a contradiction with the assumption that Υ is the time–space schedule.
(3) Let $i, j \in START(a_2, \tau)$.
The proof can be performed analogously as in the case (2).
(4) Let $i \in START(a_1, \tau)$, $j \in START(a_2, \tau)$.
Then $r_i = a_1$, $r_j = a_2$. From the Algorithm we get $r'_i = a_2$, $r'_j = a_1$. From $a_1 \neq a_2$ we obtain $r'_i \neq r'_j$. It is a contradiction with the assumption $r'_i = r'_j$.
(5) Let $i \in START(a_2, \tau)$, $j \in START(a_1, \tau)$.
The proof can be performed analogously as in the case (4).
(6) Let $i \in START(a_1, \tau)$, $j \in E - E^{\cup}$.
From $i \in START(a_1, \tau)$ we get $r_i = a_1$, $r'_i = a_2$, $\tau \leq t_i \leq \tau_k$. This and the assumption $r'_i = r'_j$ implies $r'_j = a_2$. For $j \in E - E^{\cup}$ the Algorithm gives $r'_j = r_j$. Consequently, $r_j = a_2$. By the assumption, $START(a_2, \tau)$ is a block. Hence $START(a_2, \tau) = END(a_2, \tau)$. From $j \notin START(a_2, \tau)$ we obtain $j \notin END(a_2, \tau)$. From $r_j = a_2$ and $j \notin END(a_2, \tau)$ it follows $\Theta(t_j, d_j - 1) < \tau$. This and $\tau \leq t_i$ implies $H_i \cap H_j = \emptyset$. It is a contradiction with the assumption $H_i \cap H_j \neq \emptyset$.
(7) Let $i \in START(a_2, \tau)$, $j \in E - E^{\cup}$.
The proof can be performed analogously as in the case (6).
(8) Let $j \in START(a_2, \tau)$, $i \in E - E^{\cup}$.
The proof can be performed analogously as in the case (6).
(9) Let $j \in START(a_1, \tau)$, $i \in E - E^{\cup}$.
The proof can be performed analogously as in the case (6).

Summarizing, we obtain that Υ' contains no conflict in any room $r \in R$. In other words:

$$\text{if } r'_i = r'_j \quad \text{then } H_i \cap H_j = \emptyset, \text{ for all } i, j \in E, i \neq j.$$

By the assumption, Υ is a time–space schedule of the set of events E. Hence Υ satisfies the third condition of Definition 7, namely:

$$\text{if } U_i \cap U_j \neq \emptyset \text{ or } S_i \cap S_j \neq \emptyset \text{ or } r_i = r_j, \quad \text{then } H_i \cap H_j = \emptyset, \text{ for all } i, j \in E, i \neq j.$$

Consequently,

$$\text{if } U_i \cap U_j \neq \emptyset \text{ or } S_i \cap S_j \neq \emptyset \text{ or } r'_i = r'_j, \quad \text{then } H_i \cap H_j = \emptyset, \text{ for all } i, j \in E, i \neq j.$$

This means that Υ' satisfies the third condition of Definition 7. We proved that Υ' satisfies all conditions of Definition 7. Consequently, Υ' is a time–space schedule of the set of events E. $\square$

6. Improvement of a schedule for UCTP1

This section presents an algorithm which improves the quality of a schedule so that in a new schedule there exists a free room of a desired type and in a desired time. The main result of the section is Theorem 32. The motivation of this theorem can be described as follows.

A time–space schedule can be constructed by the following method. We start with an empty schedule. At the first step, we assign a timeslot t_{e_1} and a room r_{e_1} to the first event. At the second step, we assign a timeslot t_{e_2} and a room r_{e_2} to the second event. After $i - 1$ steps we have a time–space schedule $\Upsilon = (t_{e_j}, r_{e_j})_{j=1}^{i-1}$ of the set of events $E^{i-1} = \{e_1, e_2, \ldots, e_{i-1}\}$ with the matrix of numbers of free rooms Q. At i-th step we want to assign a timeslot τ to t_{e_i}. It is possible that we will have no room of set R_{e_i} free in all $t \in H_{e_i}$. In this case, if $Q_{mt} > 0$ for $m = y_{e_i}$ and all $t \in H_{e_i}$, then we improve the quality of the schedule Υ by a sequence of interchanges of blocks. The details are described in the proof of Theorem 32. We get a new schedule $\Upsilon' = (t_{e_j}, r'_{e_j})_{j=1}^{i-1}$ of the set of events E^{i-1} with a room $r \in R_{e_i}$ which is free during all timeslots $t \in H_{e_i}$. Then we assign $t_{e_i} = \tau$ and $r_{e_i} = r$. The i-th step is completed and we continue at the step $i + 1$. These sketched remarks serve to point out on the idea of the following theorem.

Theorem 32. *Let Assumption 12 be satisfied. Let $i \in I$ and $\Upsilon = (t_{e_j}, r_{e_j})_{j=1}^{i-1}$ be a time–space schedule of the set of events $E^{i-1} = \{e_1, e_2, \ldots, e_{i-1}\}$ with the matrix of numbers of free rooms Q. Suppose that $t_{e_i} \in T_{e_i}$. If $Q_{mt} > 0$ for $m = y_{e_i}$ and all $t \in H_{e_i}$ then there exists a time–space schedule $\Upsilon' = (t_{e_j}, r'_{e_j})_{j=1}^{i-1}$ of the set of events E^{i-1} with a room $r \in R_{e_i}$ which is free during all $t \in H_{e_i}$.*

Proof. In the sequel, we write τ instead of t_{e_i}. Similarly, we write d instead of d_{e_i}.

Let $i \in I$. By the assumption, $\tau \in T_{e_i}$. Consequently by Remark 3, $H_{e_i} \subseteq T$. By Definition 1, $d \in \mathbb{N}$ and $1 \leq d \leq k$. Hence $H_{e_i} \neq \emptyset$.

Suppose that $Q_{mt} > 0$ for $m = y_{e_i}$ and all $t \in H_{e_i}$. It means that for each timeslot $t \in H_{e_i}$ there exists at least one free room of the set R'_m in the time–space schedule Υ. We denote these free rooms by $a_0, a_1, \ldots, a_{d-1}$. The room $a_0 \in R'_m$ is free in the timeslot τ, the room $a_1 \in R'_m$ is free in the timeslot $\Theta(\tau, 1)$ etc., the room $a_{d-1} \in R'_m$ is free in the timeslot $\Theta(\tau, d-1)$. As $m = y_{e_i}$, from Definition 10 we obtain $R'_m = R_{e_i}$.

If $d = 1$ then we set $\Upsilon' = \Upsilon$ and $r = a_0$. The room $r = a_0 \in R'_m = R_{e_i}$ is free during all timeslots $t \in H_{e_i} = \{\tau\}$ in the time–space schedule Υ'. Hence the assertion of the theorem is valid for $d = 1$.

Next we assume $d > 1$. We proceed by induction. We set $\Upsilon^0 = \Upsilon$.

The induction hypothesis is that after $l < d$ steps, we compute $\Upsilon^l = (t_{e_j}, r^l_{e_j})_{j=1}^{i-1}$, where Υ^l is a time–space schedule of the set of events E^{i-1} and the room a_{d-1-l} is free in all timeslots $t \in [\Theta(\tau, d-1-l), \Theta(\tau, d-1)]$ in the schedule Υ^l.

Basis. By the assumption of the theorem, Υ^0 is the time–space schedule of the set of events E^{i-1}. The room a_{d-1} is free in the timeslot $\Theta(\tau, d-1)$ in the schedule Υ^0. Hence the induction hypothesis is true for $l = 0$.

Inductive step. Assume that after $l-1$ steps, for $0 < l < d$ we have computed $\Upsilon^{l-1} = (t_{e_j}, r^{l-1}_{e_j})_{j=1}^{i-1}$, where Υ^{l-1} is the time–space schedule of the set of events E^{i-1} and the room a_{d-l} is free in all timeslots $t \in [\Theta(\tau, d-l), \Theta(\tau, d-1)]$ in the schedule Υ^{l-1}.

At the l-th step we deal with the schedule Υ^{l-1}. From the relations $d > 1$ and $0 < l < d$ it follows $\Theta(\tau, d-l) \in H_{e_i}$, $\Theta(\tau, d-l-1) \in H_{e_i}$ and that the rooms a_{d-l}, a_{d-l-1} belong to R'_m. By the induction assumption, the room a_{d-l} is free in the timeslot $\Theta(\tau, d-l)$ in the schedule Υ^{l-1}. By Lemma 28 we obtain that $START(a_{d-l}, \Theta(\tau, d-l))$ is a block. The room a_{d-l-1} is free in the timeslot $\Theta(\tau, d-l-1)$ in the schedule Υ^0. At previous steps we performed no change before the timeslot $\Theta(\tau, d-l)$. Hence the room a_{d-l-1} is free in the timeslot $\Theta(\tau, d-l-1)$ also in the schedule Υ^{l-1}. By Lemma 29 we get that $START(a_{d-l-1}, \Theta(\tau, d-l))$ is a block. The assumptions of Theorem 31 are satisfied for the schedule Υ^{l-1}, the rooms a_{d-l}, a_{d-l-1} and the timeslot $\Theta(\tau, d-l)$.

At the l-th step we interchange the block $START(a_{d-l}, \Theta(\tau, d-l))$ with the block $START(a_{d-l-1}, \Theta(\tau, d-l))$ by Algorithm 3 in the schedule Υ^{l-1}. We denote $\Upsilon^l = (t_{e_j}, r^l_{e_j})_{j=1}^{i-1}$ the output of Algorithm 3. It means that we perform the command $INTERCHANGE(a_{d-l}, a_{d-l-1}, \Theta(\tau, d-l), i-1, \Upsilon^{l-1}, \Upsilon^l)$.

By Theorem 31, Υ^l is a time–space schedule of the set of events E^{i-1}. By the induction assumption, the room a_{d-l} is free in all timeslots $t \in [\Theta(\tau, d-l), \Theta(\tau, d-1)]$ in the schedule Υ^{l-1}. Consequently, after the interchange, the room a_{d-l-1} is free in all timeslots $t \in [\Theta(\tau, d-l), \Theta(\tau, d-1)]$ in the schedule Υ^l. The room a_{d-l-1} is free in the timeslot $\Theta(\tau, d-l-1)$ in the schedule Υ^0. At previous steps and at the l-th step we performed no change before the timeslot $\Theta(\tau, d-l)$. Hence the room a_{d-l-1} is free in the timeslot $\Theta(\tau, d-l-1)$ also in the schedule Υ^l. It follows that the room a_{d-l-1} is free in all timeslots $t \in [\Theta(\tau, d-l-1), \Theta(\tau, d-1)]$ in the schedule Υ^l.

From the induction principle it follows that after $d-1$ steps we computed a time–space schedule $\Upsilon^{d-1} = (t_{e_j}, r^{d-1}_{e_j})_{j=1}^{i-1}$ of the set of events E^{i-1} and the room a_0 is free in all timeslots $t \in [\tau, \Theta(\tau, d-1)]$ in the schedule Υ^{d-1}. Consequently, the room a_0 is free in all timeslots $t \in H_{e_i}$ in the schedule Υ^{d-1}.

We set $\Upsilon' = \Upsilon^{d-1}$ and $r = a_0$. Because $r = a_0 \in R'_m = R_{e_i}$ we get $r \in R_{e_i}$. The room $r \in R_{e_i}$ is free in all timeslots $t \in H_{e_i}$ in the time–space schedule Υ' of the set of events E^{i-1}. □

Algorithm 4. *Improvement of a time–space schedule*

Input: $i, \tau, d, \Upsilon = (t_{e_j}, r_{e_j})_{j=1}^{i-1}$

Output: $\Upsilon' = (t_{e_j}, r'_{e_j})_{j=1}^{i-1}, r$

Assumptions: the assumptions are presented in Theorem 32.

Method: the method is presented in the proof of Theorem 32.

```
procedure IMPROVEMENT$(i, \tau, d, \Upsilon, \Upsilon', r)$;
begin
    find free rooms $a_0, a_1, \ldots, a_{d-1} \in R'_m$ ;
    $\Upsilon^0 := \Upsilon$;
    for $l := 1$ to $d-1$ do
        INTERCHANGE$(a_{d-l}, a_{d-l-1}, \Theta(\tau, d-l), i-1, \Upsilon^{l-1}, \Upsilon^l)$;
    $\Upsilon' := \Upsilon^{d-1}$;
    $r := a_0$
end;
```

Corollary 33. *Let Assumption 12 be satisfied. Algorithm 4 improves a time–space schedule $\Upsilon = (t_{e_j}, r_{e_j})_{j=1}^{i-1}$ to a time–space schedule $\Upsilon' = (t_{e_j}, r'_{e_j})_{j=1}^{i-1}$ in accordance with Theorem 32. A time of Algorithm 4 is $O(\eta^2)$.*

Proof. Obviously, Algorithm 4 performs a process described in the proof of Theorem 32. Hence Algorithm 4 improves a time–space schedule Υ to a time–space schedule Υ' in accordance with Theorem 32.

By Theorem 31, the command *INTERCHANGE*$(a_{d-l}, a_{d-l-1}, \Theta(\tau, d-l), i-1, \Upsilon^{l-1}, \Upsilon^{l})$ requires a time $O(\eta)$. The loop **for** has $d-1$ steps. From Definition 1 we get $d-1 < k$. By Notation 4 we obtain $d-1 < \eta$. Thus the loop **for** requires a time $O(\eta^2)$.

The algorithm which finds free rooms $a_0, a_1, \dots, a_{d-1} \in R'_m$ requires a time $O(\eta^2)$ (cf. Remark 36). Hence Algorithm 4 requires a time $O(\eta^2)$. □

Definition 34. Let $\Upsilon = (t_{e_j}, r_{e_j})_{j=1}^{n}$ be a time–space schedule of the set of events E. The symbol C will denote the matrix of type $g \times k$, whose elements C_{rt} are defined as follows. If there exists an event e which occupies the room r in the timeslot t, then we put $C_{rt} = e$; otherwise we set $C_{rt} = 0$. The matrix C is called the *timetable of rooms* for the time–space schedule Υ.

For example, last 3 rows of Table 2 represent the timetable of rooms (instead of the symbol 0 the corresponding place is empty).

The following assertion is obvious.

Lemma 35. *Let $\Upsilon = (t_{e_j}, r_{e_j})_{j=1}^{n}$ be a time–space schedule of the set of events E. An algorithm which computes the timetable of rooms for Υ requires a time $O(\eta^2)$.*

Remark 36. Now we will present a method, which founds free rooms $a_0, a_1, \dots, a_{d-1} \in R'_m$ mentioned in the proof of Theorem 32. We compute the timetable of rooms C for the schedule Υ. We check the column τ of the matrix C. So we find a room $a_0 \in R'_m$, which is free in the timeslot τ. Similarly, we found $a_1, a_2, \dots, a_{d-1}$. The column τ has g elements. By Notation 4, $g \le \eta$. Hence we find a_0 in the linear time $O(\eta)$. From $d \le k \le \eta$ we obtain that we found d free rooms in a time $O(\eta^2)$.

7. Special space scheduling for UCTP1

In this section we will deal with UCTP1. We present a polynomial algorithm which constructs a time–space schedule $\Upsilon = (T^\Upsilon, R^\Upsilon)$ for a given vector T^Υ. In Theorem 40 we give sufficient conditions for the existence of Υ. These conditions are independent on R^Υ. The test whether T^Υ satisfies these conditions requires a time $O(n^4)$.

Definition 37. A vector R^Υ is called a *space schedule of the set of events E* if there exists a vector T^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a time–space schedule of the set of events E.

Definition 38. The problem to find a space schedule of a given set of events E is called a *space scheduling*.

Definition 39. A *special space scheduling* problem is defined as follows. Let a time schedule T^Υ of the set of events E be given. Find a vector R^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a time–space schedule of the set of events E.

Algorithm 5. *Special space scheduling*
Input: $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$
Output: $R^\Upsilon = (r_{e_1}, r_{e_2}, \dots, r_{e_n})$ and $\Upsilon = (T^\Upsilon, R^\Upsilon) = (t_{e_j}, r_{e_j})_{j=1}^{n}$
Assumptions: the assumptions are presented in Theorem 40.
Method: the method is presented in the proof of Theorem 40.

```
procedure SPECIAL-SPACE-SCHEDULING ($T^\Upsilon, R^\Upsilon, \Upsilon$);
begin
    $\Upsilon^0 := (t_{e_j}, r^0_{e_j})_{j=1}^{0}$;   // $\Upsilon^0$ is the empty time–space schedule
    for $i := 1$ to $n$ do begin
        IMPROVEMENT$(i, t_{e_i}, d_{e_i}, \Upsilon^{i-1}, \Upsilon', r)$;   // $\Upsilon' = (t_{e_j}, r'_{e_j})_{j=1}^{i-1}$
        for $j := 1$ to $i-1$ do $r^i_{e_j} := r'_{e_j}$;
        $r^i_{e_i} := r$;
        $\Upsilon^i := (t_{e_j}, r^i_{e_j})_{j=1}^{i}$
    end;
    $\Upsilon := \Upsilon^n$;
    $R^\Upsilon := (r^n_{e_j})_{j=1}^{n}$
end;
```

Theorem 40. *Let Assumption 12 be satisfied and a vector $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ be given. Suppose that Q is the matrix computed by Algorithm 2 for the input T^Υ. If the conditions*

(1) $t_e \in T_e$ *for all* $e \in E$,

(2) *if* $U_i \cap U_j \neq \emptyset$ *or* $S_i \cap S_j \neq \emptyset$*, then* $H_i \cap H_j = \emptyset$*, for all* $i, j \in E$, $i \neq j$,
(3) $Q_{mt} \geq 0$ *for all* $(m, t) \in M \times T$

are satisfied then there exists a vector R^{Υ} *such that* $\Upsilon = (T^{\Upsilon}, R^{\Upsilon})$ *is a time–space schedule of the set of events* E. *Algorithm 5 computes* R^{Υ} *and* Υ *in a time* $O(\eta^3)$.

Proof. Let the vector $T^{\Upsilon} = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ satisfy all conditions of the theorem. Let Q^i be the state of the matrix Q after i steps of Algorithm 2 for the input T^{Υ} for $i = 0, 1, \dots, n$. In the proof of Theorem 23 it was shown that the computation of Q^i from Q^{i-1} is made by Eqs. (2). Clearly, $Q^n = Q$. From the third condition of the theorem and from Eqs. (2) we obtain

$$Q^i_{mt} \geq 0 \quad \text{for all } (m, t) \in M \times T \text{ and all } i = 0, 1, \dots, n. \tag{3}$$

We proceed by induction.

The induction hypothesis is that after $i \leq n$ steps, the algorithm computes a time–space schedule $\Upsilon^i = (t_{e_j}, r^i_{e_j})^i_{j=1}$ of the set of events $E^i = \{e_1, e_2, \dots, e_i\}$ and that Q^i is the matrix of numbers of free rooms for Υ^i.

Basis. At the 0-th step (before the loop **for** i) the algorithm assigns the empty time–space schedule to Υ^0. It means that Υ^0 is the time–space schedule of the empty set of events. From Algorithm 2 we get $Q^0_{mt} = |R'_m|$ for $(m, t) \in M \times T$. Consequently, Q^0 is the matrix of numbers of free rooms for Υ^0. Hence the induction hypothesis is true for $i = 0$.

Inductive step. Assume that after $i - 1 < n$ steps, the algorithm computes a time–space schedule $\Upsilon^{i-1} = (t_{e_j}, r^{i-1}_{e_j})^{i-1}_{j=1}$ of the set of events $E^{i-1} = \{e_1, e_2, \dots, e_{i-1}\}$ and Q^{i-1} is the matrix of numbers of free rooms for Υ^{i-1}. Algorithm 2 computes the matrix Q^i from the matrix Q^{i-1} according to Eqs. (2). From (2) and (3) we get

$$Q^{i-1}_{mt} > 0 \quad \text{for } (m, t) \in \{y_{e_i}\} \times H_{e_i}. \tag{4}$$

Consequently, Υ^{i-1} satisfies the conditions of Theorem 32 for $i \in I$. At the i-th step, the algorithm performs the command *IMPROVEMENT*$(i, t_{e_i}, d_{e_i}, \Upsilon^{i-1}, \Upsilon', r)$. By Corollary 33, this command computes Υ' and r, where Υ' is a time–space schedule of the set of events E^{i-1} and $r \in R_{e_i}$ is free during all timeslots $t \in H_{e_i}$ in Υ'. Next, the algorithm computes $\Upsilon^i = (t_{e_j}, r^i_{e_j})^i_{j=1}$, where Υ^i extends Υ' by scheduling of the event e_i to the room r (namely, we have in mind the command $r^i_{e_i} := r$). This and the first and the second condition of the theorem imply that Υ^i satisfies all conditions of Definition 7. Consequently, Υ^i is a time–space schedule of the set of events E^i. Obviously, Q^i is the matrix of numbers of free rooms for Υ^i.

From the induction principle it follows that after n steps the algorithm computes a time–space schedule $\Upsilon^n = (t_{e_j}, r^n_{e_j})^n_{j=1}$ of the set of events $E^n = E$. We set $\Upsilon = \Upsilon^n$.

The 0-th step of the algorithm requires a constant time. By Corollary 33, the command *IMPROVEMENT*$(i, t_{e_i}, d_{e_i}, \Upsilon^{i-1}, \Upsilon', r)$ requires a time $O(\eta^2)$. Consequently, the i-th step of the algorithm for $i > 0$ requires a time $O(\eta^2)$. Hence the algorithm performs $n \leq \eta$ steps in the loop **for** in a time $O(\eta^3)$. Thus, the algorithm requires a time $O(\eta^3)$. □

Corollary 41. *If Assumption 12 is satisfied, then Algorithm 5 solves the special space scheduling in a time* $O(\eta^3)$.

8. Conditions for a time schedule concerning UCTP1

Theorem 42. *Let Assumption 12 be satisfied and a vector* $T^{\Upsilon} = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ *be given. Suppose that* Q *is the matrix computed by Algorithm 2 for the input* T^{Υ}. *The following conditions are necessary and sufficient for the vector* T^{Υ} *to be a time schedule of the set of events* E:

(1) $t_e \in T_e$ *for all* $e \in E$,
(2) *if* $U_i \cap U_j \neq \emptyset$ *or* $S_i \cap S_j \neq \emptyset$*, then* $H_i \cap H_j = \emptyset$*, for all* $i, j \in E$, $i \neq j$,
(3) $Q_{mt} \geq 0$ *for all* $(m, t) \in M \times T$.

Proof. *Necessity*. Let T^{Υ} be the time schedule of the set of events E. According to Definition 19, there exists a vector R^{Υ} such that $\Upsilon = (T^{\Upsilon}, R^{\Upsilon})$ is a time–space schedule of the set of events E. By Definition 7, the first and the second condition of the theorem are necessarily satisfied. In view of Theorem 23, the matrix Q computed by Algorithm 2 for the input T^{Υ} is the matrix of numbers of free rooms for Υ.

By way of contradiction, assume that the third condition of the theorem is not satisfied. Then there exists $(m, t) \in M \times T$ such that $Q_{mt} < 0$. Consequently, the number of free rooms of the set R'_m in the timeslot t is negative, which is not possible. We arrived at a contradiction.

Sufficiency. Let us assume that the vector T^{Υ} satisfies all conditions of the theorem. These conditions are equivalent to the conditions of Theorem 40. Hence the vector T^{Υ} satisfies all conditions of Theorem 40. By this theorem, there exists a vector R^{Υ} such that $\Upsilon = (T^{\Upsilon}, R^{\Upsilon})$ is a time–space schedule of the set of events E. By Definition 19, T^{Υ} is a time schedule of the set E. □

9. Polynomial reduction of UCTP1 to the time scheduling

The idea of reducibility is a fundamental concept in the theory of complexity. (Cf. [3].)

A problem Π_1 is reducible to a problem Π_2 if there exists a transformation (algorithm) A, which transforms each instance π_1 of Π_1 to an equivalent instance π_2 of Π_2. The equivalence is understood so that π_1 has a solution if and only if π_2 has a solution. The algorithm A allows one to transfer an arbitrary algorithm for the problem Π_2 to an algorithm for the problem Π_1. A reduction is polynomial if the algorithm A is polynomial.

Theorem 43. *If Assumption 12 is satisfied, then the time–space scheduling is polynomially reducible to the time scheduling.*

Proof. Let us suppose that a set of events E is given and that Assumption 12 is satisfied. We transform time–space scheduling to time scheduling. Both problems have the same input—the set of events E. Hence the time of this transformation is a constant (equal to 0). We shall show that time–space scheduling has a solution for the input E if and only if time scheduling has a solution for the input E.

Necessity. We assume that time–space scheduling has a solution for the input E. This solution is a time–space schedule $\Upsilon = (T^\Upsilon, R^\Upsilon)$ of the set of events E. According to Definition 19, T^Υ is a time schedule of the set of events E. Consequently, time scheduling has a solution for the input E.

Sufficiency. We assume that time scheduling has a solution for the input E. This solution is a time schedule T^Υ of the set of events E. According to Theorem 42, T^Υ satisfies all conditions of Theorem 42. These conditions are equivalent to the conditions of Theorem 40. By this theorem, there exists a vector R^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a time–space schedule of the set of events E. Consequently, time–space scheduling has a solution for the input E. □

Corollary 44. *The problem UCTP1 is polynomially reducible to time scheduling.*

Theorem 45. *The problem UCTP1 is possible to decompose into two problems, namely,* (i) *a time scheduling, and* (ii) *a special space scheduling.*

Proof. Let E be a given set of events. We will solve UCTP1 in the following way.

We solve time scheduling for the input E. According to Theorem 42, it means that we construct a vector T^Υ, which satisfies all conditions of Theorem 42. We use an algorithm which generates and tests all candidates for a time schedule (cf. Proposition 49). Also, we can use other algorithms or metaheuristics. There are two possibilities:

(1) *Time scheduling has a solution for the input E.*
This solution is a time schedule T^Υ of the set of events E. According to Theorem 42, T^Υ satisfies all conditions of Theorem 42. These conditions are equivalent to the conditions of Theorem 40. By this theorem, there exists a vector R^Υ such that $\Upsilon = (T^\Upsilon, R^\Upsilon)$ is a time–space schedule of the set of events E, and Algorithm 5 computes R^Υ and Υ in a time $O(\eta^3)$. By Corollary 41, Algorithm 5 solves the special space scheduling problem. Consequently, UCTP1 has a solution for the input E.

(2) *Time scheduling has no solution for the input E.*
In this case, there does not exist any vector T^Υ satisfying all conditions of Theorem 42. Then there cannot exist a time–space schedule $\Upsilon = (T^\Upsilon, R^\Upsilon)$ of the set of events E. Consequently, UCTP1 has no solution for the input E. □

Definition 46. University Course Timetabling Problem which does not satisfy Assumption 12 will be called *UCTP2*.

We will deal with the following hypothesis.

Hypothesis 47. The problem UCTP2 is not possible to decompose into two problems, namely, (i) a time scheduling, and (ii) a special space scheduling.

The arguments for the validity of this hypothesis can be described as follows.

(1) It is obvious that the definition of a time schedule must be as in Definition 19.
(2) Similarly, it is obvious that the condition $Q_{mt} \geq 0$ for all $(m, t) \in M \times T$ is necessary for a time schedule T^Υ.
(3) A test whether a vector T^Υ is a time schedule requires to know the matrix Q.
(4) If Assumption 12 is not satisfied, then to the computation of the matrix Q are necessary both components T^Υ and R^Υ of the schedule Υ.
(5) If Assumption 12 is not satisfied, then a test whether a vector T^Υ is a time schedule requires to know the space schedule R^Υ.
(6) Other arguments are presented below Definition 19.

10. Complexity of UCTP

We will study the complexity of UCTP. For a speed-up of algorithms for UCTP we use the matrix D, which is defined as follows.

Definition 48. The symbol D will denote the matrix of type $n \times n$, in which $D_{ii} = 0$ for each $i \in E$, and for all $i, j \in E$ with $i \neq j$ we have $D_{ij} = 1$ if

$$U_i \cap U_j \neq \emptyset \quad \text{or} \quad S_i \cap S_j \neq \emptyset$$

and $D_{ij} = 0$ otherwise.

We compute the matrix D at the beginning of an algorithm for UCTP in a time $O(n^4)$.

Proposition 49. *There exists an algorithm, which solves UCTP1 and has a time complexity $O(n^2k^n)$; further, there exists an algorithm, which solves UCTP2 and has a time complexity $O(n^2k^ng^n)$. Both these algorithms have the same basic method.*

Proof. First we construct an algorithm for UCTP1. By Theorem 45, UCTP1 is possible to decompose into two problems, namely, (i) a time scheduling, and (ii) a special space scheduling. We design an algorithm for the time scheduling problem as follows. A vector $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ where $t_e \in T$ for all $e \in E$ is a candidate for a time schedule. A candidate T^Υ is a time schedule, if it satisfies the conditions of Theorem 42. It is clear that these conditions can be verified in a time $O(n^4)$. If we use the matrix D, then this time is possible to improve on $O(n^2)$. We are able to give an algorithm which generates all such candidates. Generating a candidate requires a time $O(n)$. Generating and testing a candidate requires a time $O(n) + O(n^2) \leq O(n^2)$. The number of all candidates is k^n. Hence a time complexity of this algorithm for time scheduling is $O(n^2k^n)$. A special space scheduling problem has a time complexity $O(n^3)$. (cf. Corollary 41.) Therefore the time of this algorithm for UCTP1 is $O(n^2k^n) + O(n^3) \leq O(n^2k^n)$.

Now we construct an algorithm for UCTP2 in the same manner, the difference is only in technical details. A pair of vectors (T^Υ, R^Υ) such that $T^\Upsilon = (t_{e_1}, t_{e_2}, \dots, t_{e_n})$ and $R^\Upsilon = (r_{e_1}, r_{e_2}, \dots, r_{e_n})$, $t_e \in T$, $r_e \in R$ for all $e \in E$, is a candidate for a time–space schedule. A candidate (T^Υ, R^Υ) is a time–space schedule, if it satisfies the conditions of Definition 7. If we use the matrix D, then these conditions can be verified in a time $O(n^2)$. We are able to give an algorithm which generates all such candidates. Generating and testing a candidate requires a time $O(n) + O(n^2) \leq O(n^2)$. The number of all candidates is k^ng^n. Hence a time complexity of this algorithm for UCTP2 is $O(n^2k^ng^n)$. □

A natural question arises if any quicker algorithm exists for UCTP. We know that UCTP is NP-hard. (cf. [1,11].) Hence for UCTP only exponential algorithms are known.

We constructed an algorithm for UCTP1 based on methods of the theory of artificial intelligence (backtracking strategies with heuristics). We tested the algorithm on random generated instances of UCTP1. We tested an interactive version of this algorithm on real instances of practise. The result of experiments are as follows. If an instance of UCTP1 has a solution, then the algorithm requires the polynomial time. If an instance of UCTP1 has no solution, then the algorithm requires the exponential time. The time complexity of the algorithm is as in Proposition 49.

Hypothesis 50. If $T_1(n)$ is a time complexity of UCTP1 and $T_2(n)$ is a time complexity of UCTP2 then $T_2(n) = O(g^nT_1(n))$.

The argument for the validity of this hypothesis is that we know only algorithms with a time complexity as in Hypothesis 50.

11. Practice

Now we will consider the question when is Assumption 12 satisfied in practice. The author for several years prepared a schedule at P.J. Šafárik University (mainly at Faculty of Science) in Košice. Basic parameters of this faculty are $k = 65$, $n \cong 700$, $g \cong 70$. Our experiences are as follows. The ideas presented in this article have been used before the article was written. This means that Assumption 12 was fulfilled and the problem was decomposed into two problems according to Theorem 45. The situation can be described in the following way.

The input data for UCTP are prepared on three levels. On the first level, the input data are given by users—departments, teachers and students. These data need not be consistent, namely from the reason that time requirements of events (the sets T_e) can lead to the nonexistence of a schedule. On this level users prepare space requirements of some events (the sets R_e). It is concerned on events, which require special rooms, for example laboratories.

On the second level, the input data are prepared by a commission for the schedule (people constructing the schedule). On this level, the commission prepares space requirements of mostly events. Here the commission performs steps to obtain the validity of Assumption 12.

On the third level, the commission searches for events leading to nonexistence of a schedule. The commission modifies the sets T_e of these events after consultations with users. This is done before and during the construction of a schedule. Hence only interactive methods (algorithms) are applied in practice.

It can be assumed that the situation at other universities is analogous. This means that input data can be prepared such that Assumption 12 would be satisfied. In this way we obtain a radical (exponential) speed-up of algorithms for UCTP (cf. Proposition 49 and Hypothesis 50).

12. Conclusion

We studied 2-dimensional University Course Timetabling Problem (UCTP). The following two cases were distinguished:

(1) UCTP1— Assumption 12 is satisfied.
(2) UCTP2— Assumption 12 is not satisfied.

Assumption 12 means that each room belongs to exactly one type of room. This assumption is natural and accessible in practice.

We proved that UCTP1 is possible to decompose into two problems, namely, (i) a time scheduling and (ii) a special space scheduling. We argued that UCTP2 is not possible to decompose into these two problems. Also, we showed that algorithms for UCTP2 require exponentially more time than analogous algorithms for UCTP1.

The method for solution UCTP1 was applied at P.J. Šafárik University. People responsible for the schedule are able to do arrangements leading to fulfillment of Assumption 12. An open problem is if the arrangements can be expressed by conditions, algorithms or metaheuristics. This is a matter for further research.

We showed that the polynomial reduction is a strong tool for achievement of a decomposition of a 2-dimensional problem onto two 1-dimensional problems.

Not only timetabling problems, but also other combinatorial problems are 2-dimensional. A description of a decomposition of such problems onto 1-dimensional problems or a decision whether such decomposition exists, can be a subject of a further research. We will focus on the theory of operating systems of multiprocessor systems, where Assumption 12 for resources is satisfied (for example, the deadlock problem). Further, it is possible to investigate the application of methods of this paper to the theory of networks for building distributed systems and clustered systems.

Acknowledgements

The author wishes to thank the reviewers for many constructive suggestions.

References

[1] A.S. Asratian, D. De Werra, A generalized class-teacher model for some timetabling problems, European Journal of Operational Research 143 (2002) 531–542.
[2] R. Bai, E.K. Burke, G. Kendall, B. McCollum, A simulated annealing hyper-heuristic for university course timetabling, in: Proc. PATAT'06, Brno, 2006, pp. 345–350.
[3] D.P. Bovet, P. Crescenzi, Introduction to the Theory of Complexity, Prentice Hall, London, 1994.
[4] E.K. Burke, G. Kendall, E. Soubeiga, A tabu-search hyper-heuristic for timetabling and rostering, Journal of Heuristics 9 (2003) 451–470.
[5] E.K. Burke, B. MacCarthy, S. Petrovic, R. Qu, Multiple-retrieval case-based reasoning for course timetabling problems, Journal of the Operational Research Society 57 (2006) 148–162.
[6] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for timetabling problems, European Journal of Operational Research 176 (2007) 177–192.
[7] E.K. Burke, J.P. Newall, Enhancing timetable solutions with local search methods, in: Proc. PATAT'02, in: Lecture Notes in Computer Science, vol. 2740, Springer, Berlin, 2003, pp. 195–206.
[8] M. Chiarandini, K. Socha, M. Birattari, O. Rossi-Doria, An effective hybrid approach for the university course timetabling problem, Journal of Scheduling 9 (2006) 403–432.
[9] S. Daskalaki, T. Birbas, Efficient solutions for a university timetabling problem through integer programming, European Journal of Operational Research 160 (2005) 106–120.
[10] M. Dimopoulou, P. Miliotis, An automated university course timetabling system developed in a distributed environment: A case study, European Journal of Operational Research 153 (2004) 136–147.
[11] S. Even, A. Itai, A. Shamir, On the comlexity of timetable and multicommodity flow problems, SIAM Journal on Computing 5 (1976) 691–703.
[12] C.C. Gotlieb, The construction of class-teacher time-tables, in: Proc. IFIP Cong.'62, Amsterdam, 1962, pp. 73–77.
[13] R. Lewis, B. Paechter, Application of the grouping genetic algorithm to university course timetabling, in: Proc. EvoCOP'05, in: Lecture Notes in Computer Science, vol. 3448, Springer, Berlin, 2005, pp. 144–153.
[14] P. Kostuch, The university course timetabling problem with a three-phase approach, in: Proc. PATAT'04, in: Lecture Notes in Computer Science, vol. 3616, Springer, Berlin, 2005, pp. 109–125.
[15] R. Perzina, Solving the university timetabling problem with optimized enrollment of students by a self-adaptive genetic algorithm, in: Proc. PATAT'06, in: Lecture Notes in Computer Science, vol. 3867, Springer, Berlin, 2007, pp. 248–263.
[16] O. Rossi-Doria, M. Samples, M. Birattari, M. Chiarandini, M. Dorigo, L.M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. Stützle, A comparison of the performance of different metaheuristics on the timetabling problem, in: Proc. PATAT'02, in: Lecture Notes in Computer Science, vol. 2740, Springer, Berlin, 2003, pp. 329–351.
[17] H. Rudová, K. Murray, University course timetabling with soft constraints, in: Proc. PATAT'02, in: Lecture Notes in Computer Science, vol. 2740, Springer, Berlin, 2003, pp. 310–328.
[18] K. Socha, J. Knowles, M. Samples, A max–min ant system for the university course timetabling problem, in: Proc. ANTS'02, in: Lecture Notes in Computer science, vol. 2463, Springer, Berlin, 2002, pp. 63–67.
[19] K. Socha, M. Samples, M. Manfrin, Ant algorithm for the university course timetabling problem with regard to the state-of-the-art, in: Proc. EvoCOP'03, in: Lecture Notes in Computer science, vol. 2611, Springer, Berlin, 2003, pp. 334–345.