IEICE TRANS. INF. & SYST., VOL.E92–D, NO.2 FEBRUARY 2009

PAPER *Special Section on Foundations of Computer Science*

Computational Complexities of University Interview Timetabling*

Naoyuki KAMIYAMA[†a)], **Yuuki KIYONARI**[††b)], ***Nonmembers***, **Eiji MIYANO**[††c)], **Shuichi MIYAZAKI**[†††d)], ***and*** **Katsuhisa YAMANAKA**[††††e)], ***Members***

SUMMARY This paper introduces a new timetabling problem on universities, called interview timetabling. In this problem, some constant number, say three, of referees are assigned to each of $2n$ graduate students. Our task is to construct a presentation timetable of these $2n$ students using n timeslots and two rooms, so that two students evaluated by the same referee must be assigned to different timeslots. The optimization goal is to minimize the total number of movements of all referees between two rooms. This problem comes from the real world in the interview timetabling in Kyoto University. We propose two restricted models of this problem, and investigate their time complexities.

key words: *timetable, scheduling, optimization, computational complexity*

1. Introduction

The problem discussed in this paper is based on a real-world problem appeared in university time-scheduling. In the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, there are approximately 20 professors, and 40 graduate course students in each year. In the final year of the course, every student submits a research thesis, and presents his/her work in twenty minutes to obtain a master degree. This presentation meeting is scheduled in parallel using two rooms. All professors are required to attend a presentation in either room.

For each student, three professors are assigned as referees basically according to the presentation topic and professors' research fields. It is mandatory for referees to attend and evaluate the assigned students' presentations. Hence, if a professor evaluates two or more students and if their presentations are scheduled in different rooms, the professor needs to move from one room to the other. Our goal is to construct a time-schedule minimizing the *cost* of the schedule, that is, the total number of movements of all professors.

For the time-schedule to be feasible, no two students must be assigned to the same timeslot if they share a common referee. Observe that a feasible solution can be found in polynomial time (if any): What we have to do is only construct the following graph G, and find its maximum matching: Each vertex v_i of G corresponds to a student s_i. There exists an edge between v_i and v_j if and only if s_i and s_j do not share a common referee. Then, it is not hard to see that a perfect matching in G gives a feasible solution (assuming that the number of students is even).

So, if we do not care about the cost, obtaining a feasible schedule is somehow easy. Hence, we consider the problem of finding a minimum cost schedule, given an initial feasible solution. We will consider the following two problems, denoted by Room and Order (see Sect. 2 for their formal definitions). Room takes an initial feasible solution as an input, as well as an assignment of referees to students. We are allowed only to exchange the rooms of two students assigned to the same timeslot by the given initial solution, but are not allowed to change the assigned timeslot. The second problem, called Order, takes the same input as Room. It allows only to exchange the timeslots of two pairs of students, but does not allow to change the assigned rooms. (Note that, in each problem, an application of the operation does not break the feasibility of a schedule.) It is natural to assume that the number of students assigned to each referee and the number of referees assigned to each student are bounded by some constants, say s and t, respectively. We call such an instance (s, t)-bounded. The problem Room(s, t) is Room whose instances are (s, t)-bounded, and Order(s, t) is defined similarly. The purpose of this paper is to investigate the time complexity of Room(s, t) and Order(s, t).

Our Contributions. By definition, Room$(1, t)$ can be solved in polynomial time for any t. This paper shows that (i) Room$(2, 1)$ is also polynomial-time solvable, (ii) Room$(2, 2)$ is $\mathcal{NP}$-hard, and (iii) Room$(3, 1)$ is $\mathcal{NP}$-hard. Note that for $s \geq s'$ and $t \geq t'$, Room(s, t) includes Room(s', t'), and thus Room$(3, 1)$ and Room$(2, 2)$ are minimal superclasses of Room$(2, 1)$. So, the complexity of Room(s, t) is determined for any s and t by the above results.

Manuscript received March 19, 2008.
Manuscript revised June 23, 2008.

[†]The author is with the Department of Architecture and Architectural Engineering, Kyoto University, Kyoto-shi, 615–8540 Japan.
[††]The authors are with the Department of Systems Design and Informatics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.
[†††]The author is with Academic Center for Computing and Media Studies, Kyoto University, Kyoto-shi, 606–8501 Japan.
[††††]The author is with the Graduate School of Information Systems, The University of Electro-Communications, Chofu-shi, 182–8585 Japan.

[*]A part of this paper was presented at the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006).

a) E-mail: is.kamiyama@archi.kyoto-u.ac.jp
b) E-mail: kiyonari@theory.ces.kyutech.ac.jp
c) E-mail: miyano@ces.kyutech.ac.jp
d) E-mail: shuichi@media.kyoto-u.ac.jp
e) E-mail: yamanaka@is.uec.ac.jp

DOI: 10.1587/transinf.E92.D.130

As for the complexity of Order, Order$(2, t)$ can be solved in polynomial time for any t by definition. We show that (iv) Order$(3, 1)$ can be solved in polynomial time, (v) Order$(6, 1)$ is $\mathcal{NP}$-hard, and (vi) Order$(4, 2)$ is $\mathcal{NP}$-hard. This leaves the complexity of Order$(4, 1)$, Order$(5, 1)$, and Order$(3, t)$ $(t \geq 2)$ open.

Related Work. In educational timetabling, a set of resources such as teachers, students, rooms, and lectures must be assigned to a set of timeslots subject to certain hard and soft constraints. There are three main categories in educational timetabling, namely, school (or class-teacher), university course, and exam timetabling (e.g., see [9]). There are a large number of researchers investigating in detail the complexity of university timetabling [2]–[6], [10].

The interview timetabling problem treated in this paper can be regarded as the classical examination timetabling problem by considering students and referees in the former problem as exams and students in the latter problem, respectively. However, interesting parameter setting where we can derive a boundary between $\mathcal{P}$ and $\mathcal{NP}$-hard is the case when s and t are small. For such settings, it is natural to interpret the problem as the interview timetabling rather than examination timetabling.

2. Problem Definition

We formally define the problem Room and Order. Let $S = \{s_1, s_2, \ldots, s_{2n}\}$ be a set of *students*, and $P = \{p_1, p_2, \ldots, p_m\}$ be a set of *professors*, where n and m are positive integers. A *referee-assignment* A is a subset of $S \times P$. Intuitively, $(s_i, p_j) \in A$ means that professor p_j is assigned to student s_i as a referee by A. For integers s and t, a referee-assignment A is called (s, t)*-bounded* if every professor appears at most s times, and every student appears at most t times in A. $R = \{r_1, r_2\}$ is a set of *rooms*, and $T = \{t_1, t_2, \ldots, t_n\}$ is a set of *timeslots*. A *schedule* C is a one-to-one mapping from S to $T \times R$. Intuitively, if s_i is mapped to (t_j, r_k), s_i will give a presentation at timeslot t_j in room r_k. A schedule C is called *infeasible* if there are two students s_i and s_j, and a professor p such that both (s_i, p) and (s_j, p) are in A, and s_i and s_j are mapped to the same timeslot under C. If C is not infeasible, it is called *feasible*.

Let C be a feasible schedule, and p be a professor who appears k times in a referee-assignment A, namely, $(s_{i_j}, p) \in A$ $(1 \leq j \leq k)$. Suppose that C maps s_{i_j} to room r_{x_j} and timeslot t_{y_j} for each j. Without loss of generality, assume that $t_{y_1} < t_{y_2} < \cdots < t_{y_k}$. Then, the *cost of a professor p in a schedule C*, denoted by $cost(C, p)$, is the number of alternations between r_1 and r_2 in the string $r_{x_1} r_{x_2} \cdots r_{x_k}$. *The cost of a schedule C* is $\sum_{p \in P} cost(C, p)$.

For example, see a referee-assignment illustrated in Fig. 1: There are six students s_1 through s_6 and six professors p_1 through p_6. Student s_1 is evaluated by two professors p_1 and p_2, and so on. One example of a schedule, say C_1, is illustrated in Fig. 2. In C_1, students s_1 through s_3, and students s_4 through s_6 are scheduled to rooms r_1 and r_2, respectively, in this order of timeslots. Then the cost of p_1 in the schedule C_1 is 2 since p_1 has to move twice, from r_1 to r_2 and then from r_2 to r_1. The costs of p_2 and p_3 are 1 and 0, respectively. As a result, the cost of the schedule C_1 is $2 + 1 + 0 + 1 + 0 + 0 = 4$.

s_1	s_2	s_3	s_4	s_5	s_6
p_1, p_2	p_2, p_3	p_1, p_4	p_4, p_5	p_1, p_6	p_2, p_6

Fig. 1 Example of referee-assignment.

	r_1	r_2
t_1	s_1	s_4
t_2	s_2	s_5
t_3	s_3	s_6

C_1

	r_1	r_2
t_1	s_1	s_4
t_2	s_5	s_2
t_3	s_3	s_6

C_2

	r_1	r_2
t_1	s_3	s_6
t_2	s_1	s_4
t_3	s_2	s_5

C_3

Fig. 2 Schedules C_1, C_2 and C_3.

Now, we are ready to formally define the problems Room and Order.

Problem Room(s, t)
Input: S, P, R, T, A, and a feasible schedule C, where A is (s, t)-bounded.
Output: A feasible schedule C' such that every student is assigned to the same timeslot by C and C'.
Optimization criteria: Minimize $cost(C')$.

Problem Order(s, t)
Input: S, P, R, T, A, and a feasible schedule C, where A is (s, t)-bounded.
Output: A feasible schedule C' such that for any pair of students s_i and s_j, if s_i and s_j are assigned to the same timeslot by C, then they are assigned to the same timeslot by C'. Also, every student is assigned to the same room by C and C'.
Optimization criteria: Minimize $cost(C')$.

For example, C_2 in Fig. 2 is one possible output of Room when C_1 in Fig. 2 is an input schedule. C_2 is the result of exchanging rooms of s_2 and s_5, by which we can improve the cost of schedule to 3. C_3 in Fig. 2 is one possible output of Order when C_1 is an input, and its cost is 3.

3. Complexity of Room

It is trivial that Room$(1, t)$ can be solved in polynomial time for any t because no professor needs to move and hence the cost is 0 for any schedule.

3.1 Polynomial-time Solvability of Room$(2, 1)$

Let us call two students who are assigned to the same timeslot by an input schedule a *student-pair* (or simply, a *pair*). In the algorithm described later, if we write a student-pair $[s_i, s_j]$, it means that s_i and s_j are assigned to rooms r_1 and r_2, respectively, by the current schedule. By "*flip a student-*

pair $[s_i, s_j]$", we mean to exchange the rooms of s_i and s_j, namely, we change $[s_i, s_j]$ to $[s_j, s_i]$.

Without loss of generality, we assume that each student is evaluated by exactly one professor (namely, there is no student to whom no referee is assigned). So, if professor p is assigned to student s by A, we write $A(s) = p$ for convenience. For a student s, $room(s)$ denotes the room which s is assigned by the current schedule.

Starting from an initial schedule C, our algorithm decides, for each student-pair, whether to flip it or not, in a sequential manner. It first selects an arbitrary pair, say $[u, v]$, and fix the rooms of this pair. It next selects a pair $[x, y]$ such that either x or y is evaluated by $A(u)$. If it is x, namely $A(u) = A(x)$, we leave the rooms of x and y unchanged, so that the professor $A(u)$ does not have to move, and the next target professor is $A(y)$. If it is y, then we flip the pair $[x, y]$, so that $A(u)$ does not need to move, and the next target professor is $A(x)$. In this way, it continues determining the rooms of pairs, so that a target professor does not have to move. If there is no pair to be selected, we go back to the initial pair $[u, v]$, and start the same operations from the other professor $A(v)$. When there is no pair to be selected, the algorithm closes this "chain", and starts the next phase by selecting another initial pair.

During the execution, a pair takes one of two statuses, *processed* and *unprocessed*. Initially, all pairs are unprocessed, and eventually every pair turns processed. Once it becomes processed, it never becomes unprocessed again. The whole description of our algorithm GREEDY_Flip is given in Fig. 3.

Algorithm GREEDY_Flip

```
 1: Set all pairs be unprocessed.
 2: While there is an unprocessed pair
 3:     {Find an arbitrary unprocessed pair $[u,v]$.
 4:      Change the status of $[u,v]$ to processed.
 5:      Set $s_a = u$ and $s_b = v$.
 6:      While there is an unprocessed pair $[x,y]$ such that
           $A(s_a) = A(x)$ or $A(s_a) = A(y)$
 7:       {If $A(s_a) = A(x)$ then
 8:           {If $room(s_a) = room(x)$, let $[x,y]$ be processed,
                and let $s_a = y$.
 9:            If $room(s_a) \neq room(x)$, flip $[x,y]$, let $[x,y]$ be
                processed, and let $s_a = y$.}
10:        If $A(s_a) = A(y)$ then
11:          {If $room(s_a) = room(y)$, let $[x,y]$ be processed,
                and let $s_a = x$.
12:           If $room(s_a) \neq room(y)$, flip $[x,y]$, let $[x,y]$ be
                processed, and let $s_a = x$.}
13:       }
14:      While there is an unprocessed pair $[x,y]$ such that
           $A(s_b) = A(x)$ or $A(s_b) = A(y)$
15:       {If $A(s_b) = A(x)$ then
16:           {If $room(s_b) = room(x)$, let $[x,y]$ be processed,
                and let $s_b = y$.
17:            If $room(s_b) \neq room(x)$, flip $[x,y]$, let $[x,y]$ be
                processed, and let $s_b = y$.}
18:        If $A(s_b) = A(y)$ then
19:          {If $room(s_b) = room(y)$, let $[x,y]$ be processed,
                and let $s_b = x$.
20:           If $room(s_b) \neq room(y)$, flip $[x,y]$, let $[x,y]$ be
                processed, and let $s_b = x$.}
21:       }
22:     }
```

Fig. 3 Algorithm GREEDY_Flip.

Theorem 1: Algorithm GREEDY_Flip runs in polynomial time, and outputs an optimal solution for Room(2, 1).

Proof. Its time complexity is clearly polynomial. We show that it outputs an optimal solution. Call the set of pairs processed during the same round of the while-loop of lines 3 through 22 a *block*. It is not hard to see that if students s_i and s_j are in different blocks, then $A(s_i) \neq A(s_j)$. For, suppose not, i.e., s_i and s_j are in different blocks, say B and B', respectively, and $A(s_i) = A(s_j)(= p)$. Then, since A is (2, 1)-bounded, p appears once in B and once in B'. Without loss of generality, suppose that B was created before B'. Then, when the algorithm completes the block B, any pair in B' is unprocessed. This means that there is a pair satisfying the condition at line 6 or 14, a contradiction. So, there arises no cost between blocks, and hence, we only have to care about the cost within each block.

Consider an arbitrary block B. Let $[u, v]$ be the pair selected at line 3 to start constructing this block. Let $[s^a_{1,1}, s^a_{1,2}], [s^a_{2,1}, s^a_{2,2}], \cdots, [s^a_{\ell,1}, s^a_{\ell,2}]$ be pairs added to B in this order during the execution of the while-loop at lines 6 through 13. Then, it is not hard to see that $A(u) = A(s^a_{1,1})$, $A(s^a_{1,2}) = A(s^a_{2,2})$, $A(s^a_{2,1}) = A(s^a_{3,1}), \cdots$, and $A(s^a_{\ell-1,1}) = A(s^a_{\ell,1})$ if ℓ is odd, and $A(s^a_{\ell-1,2}) = A(s^a_{\ell,2})$ if ℓ is even. Similarly, let $[s^b_{1,1}, s^b_{1,2}], [s^b_{2,1}, s^b_{2,2}], \cdots, [s^b_{\ell',1}, s^b_{\ell',2}]$ be pairs added in this order to B, during the execution of the while-loop at lines 14 through 21. Then, $A(v) = A(s^b_{1,2})$, $A(s^b_{1,1}) = A(s^b_{2,1})$, $A(s^b_{2,2}) = A(s^b_{3,2}), \cdots$, and $A(s^b_{\ell'-1,1}) = A(s^b_{\ell',1})$ if ℓ' is even, and $A(s^b_{\ell'-1,2}) = A(s^b_{\ell',2})$ if ℓ' is odd. It results that the cost arising from this block is at most one.

Now, let C be the schedule obtained by the algorithm GREEDY_Flip, and let B be any block which causes the cost of one in C. By the above observation, we can list pairs in B as $[s_{1,1}, s_{1,2}], [s_{2,1}, s_{2,2}], \cdots, [s_{\ell,1}, s_{\ell,2}]$ such that $A(s_{1,2}) = A(s_{2,2})$, $A(s_{2,1}) = A(s_{3,1})$, $A(s_{3,2}) = A(s_{4,2}), \cdots, A(s_{\ell-1,1}) = A(s_{\ell,1})$, and $A(s_{\ell,2}) = A(s_{1,1})$, where ℓ is odd. Namely, the professor $A(s_{\ell,2})$ has to move once. One can easily see that in any schedule, this set of pairs causes the cost of one, which proves the optimality of C. □

3.2 $\mathcal{NP}$-hardness of Room(2, 2)

Theorem 2: Room(2, 2) is $\mathcal{NP}$-hard.

Proof. Let us consider the following problem, called MAX E2LIN2: We are given n variables $x_1, x_2, \cdots, x_n$ and m equations each with exactly two variables, $x_{i_1} \oplus x_{i_2} = a_i$ $(1 \leq i \leq m)$, where $a_i \in \{0, 1\}$. We are asked to assign 0 or 1 to variables so that the number of satisfied equations is maximized (we denote by $cost(C)$ the number of satisfied equations by an assignment C). MAX E2LIN2(3) is a restriction of MAX E2LIN2 where each variable appears at most three times in the given equations. It is known that MAX E2LIN2(3) is

$\mathcal{NP}$-hard [1]. $\mathcal{NP}$-hardness of Room$(2,2)$ is proved by a polynomial-time reduction from MAX E2LIN2(3).

Given an instance I of MAX E2LIN2(3) with n variables and m equations, we construct an instance I' of Room$(2,2)$. For each variable x_i $(1 \le i \le n)$, we create a timeslot t_i and a student-pair $[s_{i,1}, s_{i,2}]$ who are assigned to the timeslot t_i, and for each equation $e_j : x_{j_1} \oplus x_{j_2} = a_j$ $(1 \le j \le m)$, we create a professor p_j. Referee-assignment A is constructed as follows. Consider the j-th equation e_j $(1 \le j \le m)$. If it is of the form $x_{j_1} \oplus x_{j_2} = 0$, then either (a1) add $(s_{j_1,1}, p_j)$ and $(s_{j_2,1}, p_j)$ to A, or (a2) add $(s_{j_1,2}, p_j)$ and $(s_{j_2,2}, p_j)$ to A. (We will later show which one should be selected. For a while, consider to take an arbitrary one.) If $x_{j_1} \oplus x_{j_2} = 1$, then either (b1) add $(s_{j_1,1}, p_j)$ and $(s_{j_2,2}, p_j)$ to A, or (b2) add $(s_{j_1,2}, p_j)$ and $(s_{j_2,1}, p_j)$ to A.

For example, consider the following instance of MAX E2LIN2(3), consisting of five variables and seven equations: $x_1 \oplus x_5 = 1$, $x_2 \oplus x_3 = 0$, $x_1 \oplus x_3 = 0$, $x_4 \oplus x_5 = 1$, $x_2 \oplus x_4 = 1$, $x_1 \oplus x_2 = 1$, and $x_3 \oplus x_5 = 0$. Then, Fig. 4 is one possible instance constructed from these equations. For example, since the equation e_1 contains two variables x_1 and x_5, the corresponding professor p_1 is assigned to students in the first and the fifth timeslots. Furthermore, since e_j is of the form $x_1 \oplus x_5 = 1$, p_1 is assigned to different rooms. (Recall that we have another choice, namely, p_1 could be assigned to $s_{1,1}$ and $s_{5,2}$.)

Observe that each professor appears twice in I' since each equation contains two variables. However, three referees may be assigned to one student since a variable can appear three times (see $s_{3,2}$ of Fig. 4). Hence at this moment, a constructed instance is $(2,3)$-bounded. We will later show that it can be modified to $(2,2)$-bounded.

We will show that an optimal solution for I can be computed from an optimal solution for I' in polynomial time. Let C' be an optimal schedule for I'. We will construct a solution C for I. For a pair of students $s_{i,1}$ and $s_{i,2}$, if they are scheduled as $[s_{i,1}, s_{i,2}]$ (i.e., their assigned rooms are the same as the initial schedule), set $x_i = 0$. Otherwise, set $x_i = 1$. We show that $cost(C) + cost(C') = m$: Consider a professor p_j. We see that p_j contributes exactly once to $cost(C)$ or $cost(C')$. There are four possible ways to determine students he/she evaluates by the reduction, namely, (a1), (a2), (b1), and (b2). Here we check only (a1), namely, the j-th equation is $x_{j_1} \oplus x_{j_2} = 0$, and p_j evaluates $s_{j_1,1}$ and $s_{j_2,1}$; other cases can be checked similarly. Assume that p_j has to move in schedule C', i.e., he/she contributes to $cost(C')$. Then students $s_{j_1,1}$ and $s_{j_2,1}$ are assigned to different rooms by C', namely, one of them is assigned to the same room as the initial schedule, and the other to the different room. By the construction of C, $x_{j_1} = 1$ and $x_{j_2} = 0$, or $x_{j_1} = 0$ and $x_{j_2} = 1$ under C, namely, the equation e_j is unsatisfied, and p_j does not contribute to $cost(C)$. By a similar observation, if p_j does not need to move, then equation e_j is satisfied. To show the optimality of C by contradiction, assume that there is an assignment C_0 for I such that $cost(C_0) > cost(C)$. Then, it is not hard to see that we can construct a schedule C'_0 for I' such that $cost(C_0)+cost(C'_0) = m$ by the reverse operation of the above construction. Then, $cost(C'_0) = m - cost(C_0) < m - cost(C) = cost(C')$, which contradicts the optimality of schedule C'.

	room r_1		room r_2	
t_1	$s_{1,1}$	p_6	$s_{1,2}$	p_1, p_3
t_2	$s_{2,1}$	p_5	$s_{2,2}$	p_2, p_6
t_3	$s_{3,1}$		$s_{3,2}$	p_2, p_3, p_7
t_4	$s_{4,1}$	p_4	$s_{4,2}$	p_5
t_5	$s_{5,1}$	p_1	$s_{5,2}$	p_4, p_7

Fig. 4 An example of the translated instance of Room$(2,3)$.

It remains to show that the above reduction can be modified so that I' is $(2,2)$-bounded. Recall that when we construct a referee-assignment, we have two choices.

For $x_{j_1} \oplus x_{j_2} = 0$, (a1) $(s_{j_1,1}, p_j), (s_{j_2,1}, p_j) \in A$, or
(a2) $(s_{j_1,2}, p_j), (s_{j_2,2}, p_j) \in A$.
For $x_{j_1} \oplus x_{j_2} = 1$, (b1) $(s_{j_1,1}, p_j), (s_{j_2,2}, p_j) \in A$, or
(b2) $(s_{j_1,2}, p_j), (s_{j_2,1}, p_j) \in A$.

Consider a student-pair $[s_{i,1}, s_{i,2}]$. Since each variable appears at most three times, the number of professors who evaluates $s_{i,1}$ or $s_{i,2}$ is at most three in total. If, for example, two and one referees are assigned to $s_{i,1}$ and $s_{i,2}$, respectively, then the condition is satisfied. However, it is possible that, for example, three and zero referees are assigned to $s_{i,1}$ and $s_{i,2}$, respectively, if we choose (a1) or (a2) ((b1) or (b2)) arbitrarily. Using the following problem, called *Modified-NAE 3SAT*, we will resolve this problem and guarantee the resulting instance to be $(2,2)$-bounded.

Modified-NAE (Not-All-Equal) 3SAT is a modification of NAE 3SAT [8]. We are given a set of clauses, where each clause contains at most three literals. A clause with three literals is unsatisfied if and only if all three literals have the same value. A clause with one or two literals is satisfied by any assignment. Modified-NAE 3SAT(2) is a special case of Modified-NAE 3SAT where each variable appears exactly twice. Now, construct an instance f of Modified-NAE 3SAT(2) as follows: A variable y_j of f corresponds to an equation e_j of instance I of MAX E2LIN2(3). A clause C_i of f corresponds to a variable x_i of I. If the j-th equation of I is $x_{j_1} \oplus x_{j_2} = 0$, then add y_j to both C_{j_1} and C_{j_2}. If the j-th equation of I is $x_{j_1} \oplus x_{j_2} = 1$, then add y_j to C_{j_1} and its negation $\overline{y_j}$ to C_{j_2}. It is not hard to see that each clause contains at most three literals since each variable of I appears at most three times. Also, note that each variable of f appears twice since each equation of I contains exactly two variables. As will be shown in Lemma 1, f is always satisfiable and a satisfying truth assignment, say P, can be found in polynomial time. We decide which of (a1) and (a2) ((b1) and (b2)) to select according to this satisfying assignment P. For a variable x, we denote by $P(x)$ the value assigned to x by P. Suppose that the j-th equation is $x_{j_1} \oplus x_{j_2} = 0$. If $P(y_j) = 0$ then select (a1), otherwise select (a2). Suppose that the j-th equation is $x_{j_1} \oplus x_{j_2} = 1$. If $P(y_j) = 0$ then select (b1), otherwise select (b2).

The above construction results in a $(2, 2)$-bounded instance. For, suppose not, and assume that there is a pair of students $[s_{i,1}, s_{i,2}]$, where, for example, three referees are assigned to $s_{i,1}$. Then, by the construction of the above f, all literals in clause C_i have value 0, contradicting the fact that P is a satisfying assignment. Similarly, if $s_{i,2}$ receives three referees, then all literals in C_i are 1, and again, a contradiction. □

Lemma 1: Any instance of Modified-NAE 3SAT(2) is satisfiable, and furthermore, there is a polynomial-time algorithm to find a satisfying assignment.

Proof. We give a rough idea of the algorithm finding a satisfying assignment. Note that clauses with one or two literals are always satisfied. So, we first remove these clauses. We will determine the value of variables one by one. Select a clause C, and select any two variables in C, say, x and y. Decide the value of these two variables so that they satisfy C, namely, so that corresponding two literals in C take 0 and 1. Consider arbitrary one of these two variables, say x. If there is no clause having x, stop. Otherwise, let C_1 be the clause that contains x. If other two variables of C_1 are undetermined yet, then select one of these two variables, say z, and determine the value of z so that x and z satisfy C_1. If there is no clause having z, stop. Otherwise, select the clause having z, regard this clause as new C_1, and continue. Suppose, at some moment, we encounter a clause such that one variable has already been determined, namely, two variables in total are determined currently. If the clause is satisfied by these two variables, then we stop. If two literals have the same value, take the third variable, and determine its value so that it satisfies this clause, and continue. If we encounter a clause whose two literals are already determined, then we can see that it is already satisfied by those two literals. (For, during this procedure, we never leave a clause having two determined literals with the same value. In such a case, we determine the value of the third variable instantly, as stated above.) In this case, we stop. In this way, we will either find a new clause, or stop.

If this "chain" stops, then we start a new chain from the other variable y of C. If it also stops, then remove all clauses we have already seen (those clauses are satisfied by variables already determined). It is easy to see that all the remaining clauses have three undetermined literals. We will start a new chain by selecting an arbitrary clause, and repeat the same procedure. □

3.3 $\mathcal{NP}$-hardness of Room(3, 1)

Theorem 3: Room(3, 1) is $\mathcal{NP}$-hard.

Proof. We use a similar reduction to the proof of Theorem 2. Given an instance I of MAX E2LIN2(3), we construct an instance I_1 of Room(2, 3). This part is exactly the same to the proof of Theorem 2. Again, consider an example instance of MAX E2LIN2(3) we have used in the proof of Theorem 2. Following Fig. 5 is one possible instance constructed from MAX E2LIN2(3). (Only the assignment of professor p_2 differs from Fig. 4.)

We then construct a $(3, 1)$-bounded instance I_2 from I_1, and show that an optimal solution for I_1 can be computed in polynomial time from an optimal solution for I_2, which completes the proof of Theorem 3.

Consider two students s and s' in I_1, who are assigned to the same timeslot t_i. Recall that they are evaluated by at most three professors in total. If the total number of professors assigned to s and s' are exactly three, and if two and one professors are assigned to s and s', respectively, then we call the professor assigned to s' *critical in timeslot* t_i. For example, in Fig. 5, p_2 is critical in t_3. (Notice that neither p_4 nor p_5 is critical in t_4 because only two professors are assigned to this timeslot.) If a professor p is critical in both timeslots he/she appears, then p is called *critical*. For example, in Fig. 5, p_6 is critical. We will first modify I_1 so that there is no critical professor. This can be easily done as follows: If p is critical, assign p to the student of the other room in both timeslots p appears (see Fig. 6 (a)). Recall that we had two choices in assigning each professor, (a1) and (a2) ((b1) and

	room r_1		room r_2	
t_1	$s_{1,1}$	p_6	$s_{1,2}$	p_1, p_3
t_2	$s_{2,1}$	p_2, p_5	$s_{2,2}$	p_6
t_3	$s_{3,1}$	p_2	$s_{3,2}$	p_3, p_7
t_4	$s_{4,1}$	p_4	$s_{4,2}$	p_5
t_5	$s_{5,1}$	p_1	$s_{5,2}$	p_4, p_7

Fig. 5 An example of the translated instance of Room(2, 3).

	room r_1		room r_2	
t_1	$s_{1,1}$		$s_{1,2}$	p_1, p_3, p_6
t_2	$s_{2,1}$	p_2, p_5, p_6	$s_{2,2}$	
t_3	$s_{3,1}$	p_2	$s_{3,2}$	p_3, p_7
t_4	$s_{4,1}$	p_4	$s_{4,2}$	p_5
t_5	$s_{5,1}$	p_1	$s_{5,2}$	p_4, p_7

(a)

	room r_1		room r_2	
t_{1_1}	$s_{1_1,1}$	p^1	$s_{1_1,2}$	p_1
t_{1_2}	$s_{1_2,1}$	p^1	$s_{1_2,2}$	p_3
t_{1_3}	$s_{1_3,1}$	p^1	$s_{1_3,2}$	p_6
t_{2_1}	$s_{2_1,1}$	p_2	$s_{2_1,2}$	p^2
t_{2_2}	$s_{2_2,1}$	p_5	$s_{2_2,2}$	p^2
t_{2_3}	$s_{2_3,1}$	p_6	$s_{2_3,2}$	p^2
t_{3_1}	$s_{3_1,1}$	p_2	$s_{3_1,2}$	p_3
t_{3_2}	$s_{3_2,1}$	p_2	$s_{3_2,2}$	p_7
t_4	$s_{4,1}$	p_4	$s_{4,2}$	p_5
t_{5_1}	$s_{5_1,1}$	p_1	$s_{5_1,2}$	p_4
t_{5_2}	$s_{5_2,1}$	p_1	$s_{5_2,2}$	p_7

(b)

Fig. 6 (a) A modified Room(2, 3) instance obtained from Fig. 5, and (b) a Room(3, 1) instance constructed from Fig. 6 (a).

(b2)), from which we took arbitrary one in the previous subsection. The above operation means to replace the one with the other possibility, which resolves the criticalness of p but effects nothing to the reduction. We do this operation for all critical professors independently, and let I_1' be the resulting instance, (which is still only $(2,3)$-bounded).

Next, we modify I_1', so that each student is evaluated by at most one professor: Consider a timeslot t_i. Recall that at most three professors are assigned to this timeslot in total.

Case (1): If each student assigned to t_i is evaluated by at most one professor (as the timeslot t_4 in Fig. 6 (a)), then we leave this timeslot as it is.

Case (2): Suppose that three professors are assigned to t_i, all three for one student, say $s_{i,1}$, and zero for the other student, say $s_{i,2}$ (as the timeslots t_1 and t_2 in Fig. 6 (a)). We divide the timeslot t_i into three timeslots t_{i_1}, t_{i_2} and t_{i_3}, and replace the student $s_{i,1}$ by three students $s_{i_1,1}$, $s_{i_2,1}$, and $s_{i_3,1}$, and the student $s_{i,2}$ by $s_{i_1,2}$, $s_{i_2,2}$, and $s_{i_3,2}$ (see Fig. 6 (b)). Assign each $s_{i_j,k}$ to the timeslot t_{i_j} and the room r_k. Three professors originally assigned to $s_{i,1}$ will be re-assigned to three students $s_{i_1,1}$, $s_{i_2,1}$, and $s_{i_3,1}$ introduced instead of $s_{i,1}$, one professor for each student, in an arbitrary way. We then introduce a new professor, say p^i, independently of other timeslots, and he/she is assigned to three students $s_{i_1,2}$, $s_{i_2,2}$, and $s_{i_3,2}$.

Case (3): Suppose that three professors are assigned to t_i, two of them for one student, say $s_{i,1}$, and one for the other student, say $s_{i,2}$ (as the timeslot t_3 in Fig. 6 (a)). We divide the timeslot t_i into two timeslots t_{i_1} and t_{i_2}, and replace the student $s_{i,1}$ by two students $s_{i_1,1}$ and $s_{i_2,1}$, and the student $s_{i,2}$ by $s_{i_1,2}$ and $s_{i_2,2}$ (again, see Fig. 6 (b)). Assign each $s_{i_j,k}$ to the timeslot t_{i_j} and the room r_k. Two professors originally assigned to $s_{i,1}$ will be re-assigned to two students $s_{i_1,1}$ and $s_{i_2,1}$, one professor for each student, in an arbitrary way. One professor originally assigned to $s_{i,2}$ will be re-assigned to both students $s_{i_1,2}$ and $s_{i_2,2}$.

Case (4): Suppose that two professors are assigned to t_i, two of them for one student, say $s_{i,1}$, and zero for the other student, say $s_{i,2}$. We will do a similar operation: Divide the timeslot t_i into two timeslots. Replace students $s_{i,1}$ and $s_{i,2}$ by new (four) students. Assign two professors originally assigned to $s_{i,1}$ to two new students for $s_{i,1}$, one professor for one student, in an arbitrary way. Introduce a new professor p^i, and assign p^i to both of two new students introduced for $s_{i,2}$.

Now the translation is completed. It is not hard to see that the modified instance is now $(3,1)$-bounded: Each student is evaluated by exactly one professor. Since we have already removed "critical professors", who would appear four times if we apply the above transformation, there can be no professor appearing more than three times in the current instance. It then remains to show that an optimal solution for I_1 can be computed by an optimal solution for I_2 in polynomial time.

Let C be a feasible solution for I_2. Consider a timeslot t_i of I_1, which is divided into two or three timeslots by the above transformation. If, in C, all students in these timeslots (created from t_i) are scheduled in the same rooms as the initial schedule, or all of them are scheduled in rooms different from the initial schedule, we say that *block i of C is consistent*. We show that consistency is the best policy; namely, given any feasible solution C for I_2, we can modify C in polynomial time without increasing the cost, so that all of its blocks become consistent. If this claim is true, we can compute an optimal solution for I_1 from an optimal solution for I_2 in the following way: Given an optimal solution C_2 for I_2, we make all blocks of C_2 consistent without increasing the cost. Then, we naturally translate C_2 into a solution C_1 for I_1. Namely, for each timeslot t_j of I_1, we consider its corresponding timeslots (a block) in C_2. Under C_1, we assign two students in t_j to the same rooms as the initial schedule of I_1 if and only if all students in the corresponding block are scheduled under C_2 in the same rooms as the initial schedule of I_2. Then, it is not hard to see that the costs of C_1 and C_2 are the same. (Note that professors newly introduced in I_2 do not cause any cost since each block of C_2 is consistent and hence they do not need to move.) To show the optimality of C_1 in I_1, suppose that there is a feasible solution whose cost is smaller than $cost(C_1)$. Then, using the reverse operation of the above one, we can construct a solution for I_2 whose cost is smaller than $cost(C_2)$, contradicting the optimality of C_2.

Now, we will prove the above claim. We have three cases to consider, corresponding to Cases (2) through (4). We start from the easiest case, Case (4).

Case (4): Let $s_{i,1}$ and $s_{i,2}$ be students assigned to the timeslot t_i, and p_1 and p_2 be the professors assigned to $s_{i,1}$.

	room r_1		room r_2	
t_i	$s_{i,1}$	p_1, p_2	$s_{i,2}$	

Suppose that, by the above transformation, t_i is divided into t_{i_1} and t_{i_2}, students are replaced, and professors are re-assigned as the following table. (p^i is the newly introduced professor.)

	room r_1		room r_2	
t_{i_1}	$s_{i_1,1}$	p_1	$s_{i_1,2}$	p^i
t_{i_2}	$s_{i_2,1}$	p_2	$s_{i_2,2}$	p^i

Let C be a solution for I_2, and suppose that the block i of C is not consistent. Without loss of generality, assume that the assignment looks like the following:

	room r_1		room r_2	
t_{i_1}	$s_{i_1,1}$	p_1	$s_{i_1,2}$	p^i
t_{i_2}	$s_{i_2,2}$	p^i	$s_{i_2,1}$	p_2

Then, by exchanging the rooms of two students assigned to t_{i_2}, the cost decreases by at least one (since the cost of p^i decreases by one), but the cost increases by at most one. (Note that p^i does not appear in other places. Also, p_2 cannot appear in both before and after t_{i_2}, since p_2 appeared twice in

I_1, including in the timeslot t_i. So, the cost increase caused by p_2 is at most one.) So, in this case, we can make the block i consistent without increasing the cost.

Case (3): The corresponding block looks as follows. (All of p_1, p_2 and p_3 are professors originally existed in I_1.)

	room r_1		room r_2	
t_{i_1}	$s_{i_1,1}$	p_1	$s_{i_1,2}$	p_3
t_{i_2}	$s_{i_2,1}$	p_2	$s_{i_2,2}$	p_3

If the block is inconsistent, then, without loss of generality, we only need to consider the following case:

	room r_1		room r_2	
t_{i_1}	$s_{i_1,1}$	p_1	$s_{i_1,2}$	p_3
t_{i_2}	$s_{i_2,2}$	p_3	$s_{i_2,1}$	p_2

If we flip two students in t_{i_2}, then we can similarly decrease the cost of schedule by one, but this time, we may increase the cost by two; this can happen when p_3 is assigned to the room r_1 at the timeslot after t_{i_2}, and p_2 is assigned to r_2 at any timeslot. In such a case, however, we can claim that p_3 is not assigned to a timeslot before t_{i_1} since p_3 has already appeared three times. Then, we flip two students in the timeslot t_{i_1}, which decreases the cost of schedule by at least one, but may increase the cost of p_1 by at most one.

Case (2): We can treat this case similarly. By symmetry, the inconsistent situations are the following two types:

	room r_1		room r_2	
t_{i_1}	$s_{i_1,1}$	p_1	$s_{i_1,2}$	p^i
t_{i_2}	$s_{i_2,2}$	p^i	$s_{i_2,1}$	p_2
t_{i_3}	$s_{i_3,1}$	p_3	$s_{i_3,2}$	p^i

	room r_1		room r_2	
t_{i_1}	$s_{i_1,1}$	p_1	$s_{i_1,2}$	p^i
t_{i_2}	$s_{i_2,1}$	p_2	$s_{i_2,2}$	p^i
t_{i_3}	$s_{i_3,2}$	p^i	$s_{i_3,1}$	p_3

In the former case, flipping two students in t_{i_2} will decrease the cost by at least two, but increases the cost by at most one. In the latter case, flipping two students in t_{i_3} will decrease the cost by at least one, but increases the cost by at most one (since p_3 does not appear both before and after t_{i_3}).

□

4. Complexity of Order

Recall that the operation we are allowed in this problem is only to decide the timeslots of student-pairs. Hence, as the simplest example, if each professor evaluates at most two students, any exchange operation of student-pairs does not change the cost of the schedule. It follows that Order(s, t) can be solved in polynomial time for $s \leq 2$ and any t since any solution is optimal.

4.1 Polynomial-time Solvability of Order(3, 1)

In this section we present a polynomial-time algorithm to find an optimal solution for Order(3, 1). Note that if a professor p appears at most twice in an input referee-assignment, $cost(C, p)$ is the same for any schedule C as mentioned previously. Even if p appears three times, $cost(C, p) = 0$ for any C if all three students are assigned to the same room by the input schedule. These professors are called *non-potential* professors. If p appears three times, and if two of his/her students are assigned to one room, and the other one is assigned to the other room, his/her cost can be one or two depending on the schedule. We call these professors *potential* professors. As in the case of Room(2, 1), let $A(s)$ denote the referee assigned to student s.

Similarly to the algorithm GREEDY_Flip, a student-pair takes one of two statuses, *processed* and *unprocessed*. Our algorithm constructs several blocks of student-pairs. Starting from an arbitrary initial student-pair $[u, v]$ (line 3), it selects a student-pair $[x, y]$ where $A(u) = A(x)$ and $A(u)$ is a potential professor, if any. It schedules $[x, y]$ to the timeslot next to $[u, v]$, so that two students evaluated by professor $A(u)$ are assigned at continuous timeslots in the same room. Next, it selects a pair $[w, z]$ such that $A(y) = A(z)$ and $A(y)$ is a potential professor, if any, and schedules $[w, z]$ to the timeslot next to $[x, y]$, and so on (lines 6 through 10). When there is no pair to be selected, it then goes back to $[u, v]$, and performs the same operations by starting from $A(v)$. However, this time, it schedules new pairs to *previous* timeslots of $[u, v]$, one by one (lines 11 through 16). When there is no pair to be selected, the algorithm completes the block, and proceeds to the next while-loop (line 3) to construct a new block. Finally, blocks are scheduled in an arbitrary order. Figure 7 describes Algorithm GREEDY_Select.

Theorem 4: Algorithm GREEDY_Select runs in polynomial time, and outputs an optimal solution for Order(3, 1).

Proof. It is easy to see that GREEDY_Select runs in polynomial time. Recall that non-potential professors cause the same cost in any schedule. Hence, we are interested in only potential professors. Also, recall that each potential professor causes a cost of one or two. We show that every potential professor causes cost one in our output schedule, which completes the optimality proof.

To see this, consider an arbitrary potential professor p, and suppose that p evaluates three students s_{p1}, s_{p2}, and s_{p3}. Without loss of generality, suppose that two students, say s_{p1} and s_{p2} are scheduled in the same room. First, assume that it is r_1. Then three student-pairs must be $[s_{p1}, s_x]$, $[s_{p2}, s_y]$, and $[s_z, s_{p3}]$ for some students s_x, s_y, and s_z. Without loss of generality, assume that $[s_{p1}, s_x]$ is processed before $[s_{p2}, s_y]$.

First, suppose that $[s_{p1}, s_x]$ is processed at line 4. Then, $[s_{p2}, s_y]$ is selected right after this process (line 6), and scheduled to the timeslot next to $[s_{p1}, s_x]$ (line 7), namely, $[s_z, s_{p3}]$ is not assigned to the timeslot between those of $[s_{p1}, s_x]$ and $[s_{p2}, s_y]$. Next, observe that $[s_{p1}, s_x]$ is not pro-

```
Algorithm GREEDY_Select
 1: Set all pairs be unprocessed.
 2: While there is an unprocessed pair
 3:    {Find an arbitrary unprocessed pair $[u, v]$.
 4:     Change the status of $[u, v]$ to processed.
 5:     Set $s_a = u$ and $p = [u, v]$.
 6:     If there is an unprocessed pair $[x, y]$ such that
          $A(s_a) = A(x)$ and $A(x)$ is potential
 7:        {Assign $[x, y]$ to the timeslot next to $p$, and let
             $[x, y]$ be processed.
 8:         If there is an unprocessed pair $[w, z]$ such that
             $A(y) = A(z)$ and $A(z)$ is potential
 9:            {Assign $[w, z]$ to the timeslot next to $[x, y]$,
                 and let $[w, z]$ be processed.
10:             Let $p = [w, z]$, $s_a = w$, and goto 6. }}
11:     Set $s_b = v$ and $p = [u, v]$.
12:     If there is an unprocessed pair $[x, y]$ such that
          $A(s_b) = A(y)$ and $A(y)$ is potential
13:        {Assign $[x, y]$ to the timeslot before $p$, and let $[x, y]$
             be processed.
14:         If there is an unprocessed pair $[w, z]$ such that
             $A(w) = A(x)$ and $A(x)$ is potential
15:            {Assign $[w, z]$ to the timeslot before $[x, y]$,
                 and let $[w, z]$ be processed.
16:             Let $p = [w, z]$, $s_b = z$, and goto 12. }}
17:    }
```

Fig. 7 Algorithm GREEDY_Select.

cessed at line 7 since for this to happen, $[s_{p2}, s_y]$ must be processed just before this execution, a contradiction. Now, suppose that $[s_{p1}, s_x]$ is processed at line 9. Then, at the next execution of line 6, $[s_{p2}, s_y]$ is selected, and scheduled to the timeslot next to $[s_{p1}, s_x]$ at line 7. By a similar discussion, if $[s_{p1}, s_x]$ is processed at line 13, then $[s_{p2}, s_y]$ is scheduled to the timeslot previous to $[s_{p1}, s_x]$ at line 15. Finally, $[s_{p1}, s_x]$ is never processed at line 15. Hence, the cost of p is one. For the case that s_{p1} and s_{p2} are assigned to room r_2, we can do a similar argument. □

4.2 $\mathcal{NP}$-hardness of Order(6, 1)

Theorem 5: Order(6, 1) is $\mathcal{NP}$-hard.

Proof. Recall that we used NAE 3SAT in the proof of Theorem 2. In NAE 3SAT, we are given a set of clauses, where each clause contains exactly three literals. A clause is unsatisfied if and only if all literals it contains have the same value. It asks if there exists an assignment satisfying all clauses. This problem is known to be $\mathcal{NP}$-hard [8].

Here we use another modification of NAE 3SAT. Although the modification here is different from what we used in the proof of Theorem 2, we abuse the notation and write the problem used here as *Modified-NAE 3SAT(3)*. In Modified-NAE 3SAT(3), each clause contains two or three literals, each variable appears exactly three times, and for each variable x, each of x and $\bar{x}$ appears at least once (namely, twice positively and once negatively, or vice versa). It is not difficult to prove that Modified-NAE 3SAT(3) is $\mathcal{NP}$-hard by a reduction from NAE 3SAT: The proof is the same to the $\mathcal{NP}$-hardness proof for SAT where the appearance of each variable is restricted to three (e.g., see Proposition 9.3 of [7]). Given a NAE 3SAT instance I, we replace each occurrence of variable x_i (that appears k times) by $y_{i,1}, y_{i,2}, \ldots, y_{i,k}$. Then, we add clauses $(y_{i,1} + \overline{y_{i,2}})(y_{i,2} + \overline{y_{i,3}}) \cdots (y_{i,k-1} + \overline{y_{i,k}})(y_{i,k} + \overline{y_{i,1}})$. Observe that each variable appears exactly three times, positively at least once, and negatively at least once. Note that to satisfy additional clauses, all variables $y_{i,j}$ $(1 \le j \le k)$ must take the same value.

Before showing a reduction, we give a few remarks. In Order(6, 1), each student is evaluated by one professor. So, we ignore students in the reduction: If we write "create a professor-pair $[p_1, p_2]$", it means that we create one timeslot and two students, and we assign professors p_1 and p_2 to students assigned to rooms r_1 and r_2, respectively. Furthermore, in Order, the relative order of timeslots in an input schedule is not important. So, in the following construction, we do not specify the timeslot.

Now we start the reduction. Let I be an instance of Modified-NAE 3SAT(3). We will construct an instance I_1 of Order(6, 1). For each literal in I, we create one professor; for the j-th literal of the i-th clause, we introduce professor $p_{i,j}$. For professor-pairs, we create two types of gadgets:

Variable gadgets. Consider variable x in I. It appears three times, and hence associated with x, three professors, say $p_{a,b}$, $p_{c,d}$, and $p_{e,f}$, have been introduced. (Namely, x appears as b-th, d-th, and f-th literals of a-th, c-th, and e-th clauses, respectively.) Suppose, without loss of generality, that $p_{a,b}$ and $p_{c,d}$ correspond to the same polarity of the literal, and $p_{e,f}$ the other (i.e., $p_{a,b}$ and $p_{c,d}$ correspond to x and $p_{e,f}$ corresponds to $\bar{x}$, or vice versa). Then, we create four professor-pairs $[p_{a,b}, p_{e,f}]$, $[p_{e,f}, p_{a,b}]$, $[p_{c,d}, p_{e,f}]$, and $[p_{e,f}, p_{c,d}]$.

Clause gadgets. Consider clause c_i in I, and first suppose that it contains two literals. Then professors $p_{i,1}$ and $p_{i,2}$ have been introduced. We create two professor-pairs $[p_{i,1}, p_{i,2}]$ and $[p_{i,2}, p_{i,1}]$. Next, suppose that c_i contains three literals. Then three professors $p_{i,1}$, $p_{i,2}$, and $p_{i,3}$ have been introduced. We create three professor-pairs $[p_{i,1}, p_{i,2}]$, $[p_{i,2}, p_{i,3}]$, and $[p_{i,3}, p_{i,1}]$.

Now the reduction is completed. It is not hard to see that each professor appears at most six times; twice or four times in variable gadgets, and exactly twice in the clause gadgets. As an example, consider the following instance of Modified-NAE 3SAT(3) consisting of three clauses: $c_1 = (x_1 + \overline{x_2} + \overline{x_3})$, $c_2 = (x_1 + x_2 + x_3)$, and $c_3 = (\overline{x_1} + x_2 + \overline{x_3})$. Then, Fig. 8 is the instance constructed from these clauses. As described above, we omit students. Timeslots t_1 through t_{12} correspond to variable gadgets; t_1 through t_4, t_5 through t_8, and t_9 through t_{12} correspond to x_1, x_2, and x_3, respectively. Timeslots t_{13} through t_{21} correspond to clause gadgets; t_{13} through t_{15}, t_{16} through t_{18}, and t_{19} through t_{21} correspond to c_1, c_2, and c_3, respectively.

Note that in the instance I_1, every professor is assigned to both rooms r_1 and r_2. Thus, the optimal cost of I_1 is at least the number of professors. We will prove that I is satisfiable if and only if the optimal value of I_1 is equal to

	room r_1	room r_2
t_1	$p_{1,1}$	$p_{3,1}$
t_2	$p_{3,1}$	$p_{1,1}$
t_3	$p_{2,1}$	$p_{3,1}$
t_4	$p_{3,1}$	$p_{2,1}$
t_5	$p_{2,2}$	$p_{1,2}$
t_6	$p_{1,2}$	$p_{2,2}$
t_7	$p_{3,2}$	$p_{1,2}$
t_8	$p_{1,2}$	$p_{3,2}$
t_9	$p_{1,3}$	$p_{2,3}$
t_{10}	$p_{2,3}$	$p_{1,3}$
t_{11}	$p_{3,3}$	$p_{2,3}$
t_{12}	$p_{2,3}$	$p_{3,3}$

	room r_1	room r_2
t_{13}	$p_{1,1}$	$p_{1,2}$
t_{14}	$p_{1,2}$	$p_{1,3}$
t_{15}	$p_{1,3}$	$p_{1,1}$
t_{16}	$p_{2,1}$	$p_{2,2}$
t_{17}	$p_{2,2}$	$p_{2,3}$
t_{18}	$p_{2,3}$	$p_{2,1}$
t_{19}	$p_{3,1}$	$p_{3,2}$
t_{20}	$p_{3,2}$	$p_{3,3}$
t_{21}	$p_{3,3}$	$p_{3,1}$

Fig. 8 An example of translated instance of ORDER(6, 1).

the number of professors. Notice that if the optimal value of I_1 is equal to the number of professors, every professor moves once, either from r_1 to r_2 or from r_2 to r_1.

If-part: Suppose that we are given a schedule C whose cost is equal to the number of professors. Then, as mentioned above, each professor moves exactly once. If professor $p_{i,j}$ moves from r_1 to r_2 (from r_2 to r_1, respectively), we set the value of literal corresponding to $p_{i,j}$ (namely, the j-th literal in the i-th clause) 0 (1, respectively). We will show that, under this assignment, (i) all literals corresponding to the same variable receive the consistent values, and (ii) each clause is satisfied.

(i) Consider variable x_i in I, and suppose that three professors $p_{a,b}$, $p_{c,d}$, and $p_{e,f}$ have been introduced for x_i, where $p_{a,b}$ and $p_{c,d}$ correspond to the same polarity. Then, we have four professor-pairs $[p_{a,b}, p_{e,f}]$, $[p_{e,f}, p_{a,b}]$, $[p_{c,d}, p_{e,f}]$, and $[p_{e,f}, p_{c,d}]$. Observe that under the condition that each professor moves only once, if $p_{e,f}$ moves from r_1 to r_2 (from r_2 to r_1, respectively), then both $p_{a,b}$ and $p_{c,d}$ must move from r_2 to r_1 (from r_1 to r_2, respectively).

(ii) Consider clause c_i in I, and suppose that it contains three literals. Then three professors $p_{i,1}$, $p_{i,2}$, and $p_{i,3}$ have been introduced, and we have three professor-pairs $[p_{i,1}, p_{i,2}]$, $[p_{i,2}, p_{i,3}]$, and $[p_{i,3}, p_{i,1}]$. Note that all these three professors cannot move in the same direction, which implies that c_i is satisfied by the above construction. The same argument holds when c_i contains two literals.

Only if-part: Suppose that we are given a satisfying assignment D of I. We will construct a schedule C such that each professor moves only once. Roughly speaking, the construction is the reverse operation of the "If-part": A professor moves from r_1 to r_2 (from r_2 to r_1, respectively), if and only if its corresponding literal has the value 0 (1, respectively) under D. We show that this construction is possible. (As an example, we give in Fig. 9, a schedule for the instance in Fig. 8 corresponding to $x_1 = 1$, $x_2 = 1$, and $x_3 = 0$.)

Suppose, for example, that x_i appears twice positively (corresponding professors are $p_{a,b}$ and $p_{c,d}$) and once negatively (corresponding professor is $p_{e,f}$). Then we have four professor-pairs $[p_{a,b}, p_{e,f}]$, $[p_{e,f}, p_{a,b}]$, $[p_{c,d}, p_{e,f}]$, and

	room r_1	room r_2	
t_2	$p_{3,1}$	$p_{1,1}$	Upper parts of variable gadgets
t_4	$p_{3,1}$	$p_{2,1}$	
t_6	$p_{1,2}$	$p_{2,2}$	
t_8	$p_{1,2}$	$p_{3,2}$	
t_{10}	$p_{2,3}$	$p_{1,3}$	
t_{12}	$p_{2,3}$	$p_{3,3}$	
t_{14}	$p_{1,2}$	$p_{1,3}$	Clause gadgets
t_{15}	$p_{1,3}$	$p_{1,1}$	
t_{13}	$p_{1,1}$	$p_{1,2}$	
t_{18}	$p_{2,3}$	$p_{2,1}$	
t_{16}	$p_{2,1}$	$p_{2,2}$	
t_{17}	$p_{2,2}$	$p_{2,3}$	
t_{19}	$p_{3,1}$	$p_{3,2}$	
t_{20}	$p_{3,2}$	$p_{3,3}$	
t_{21}	$p_{3,3}$	$p_{3,1}$	
t_1	$p_{1,1}$	$p_{3,1}$	Lower parts of variable gadgets
t_3	$p_{2,1}$	$p_{3,1}$	
t_5	$p_{2,2}$	$p_{1,2}$	
t_7	$p_{3,2}$	$p_{1,2}$	
t_9	$p_{1,3}$	$p_{2,3}$	
t_{11}	$p_{3,3}$	$p_{2,3}$	

Fig. 9 An example solution for the instance of Fig. 8.

$[p_{e,f}, p_{c,d}]$. If $x_i = 0$ under D, then we determine the order of timeslots corresponding to these four professor-pairs as $[p_{a,b}, p_{e,f}]$, $[p_{c,d}, p_{e,f}]$, $[p_{e,f}, p_{a,b}]$, and $[p_{e,f}, p_{c,d}]$. Otherwise, the order is determined as $[p_{e,f}, p_{a,b}]$, $[p_{e,f}, p_{c,d}]$, $[p_{a,b}, p_{e,f}]$, and $[p_{c,d}, p_{e,f}]$. Note that each professor's movement simulates the value of each literal. We call the former two timeslots (among four) an *upper part of* x_i, and the latter two timeslots a *lower part of* x_i. Since variable gadgets corresponding to different variables do not share the same professor, we can do this construction independently for each variable.

Next, we determine the order of timeslots associated with clause gadgets. Consider a clause c_i containing three literals. Then, associated with c_i, three professor-pairs $[p_{i,1}, p_{i,2}]$, $[p_{i,2}, p_{i,3}]$, and $[p_{i,3}, p_{i,1}]$ must have been constructed. There are $3! = 6$ possibilities of determining the order of these three professor-pairs. Also, note that c_i is satisfied under D, and there are six ways of satisfying c_i. One can easily see that there is a one-to-one correspondence between the order of professor-pairs and the way of satisfying c_i; for example, if c_i is satisfied as $c_i = (0 + 1 + 0)$, then the corresponding order is $[p_{i,1}, p_{i,2}]$, $[p_{i,3}, p_{i,1}]$, and $[p_{i,2}, p_{i,3}]$. If c_i contains two literals, then there are two professor-pairs $[p_{i,1}, p_{i,2}]$ and $[p_{i,2}, p_{i,1}]$. Similarly, we have two possibilities of the order of these professor-pairs, and two possibilities of satisfying c_i, between which, there is a one-to-one correspondence again. We determine the order of timeslots depending on how c_i is satisfied. Different clauses introduce different professors, and hence we can do this operation independently for each clause.

Finally, we combine clause gadgets and variable gadgets to obtain a complete schedule. We first schedule upper parts of all variable gadgets in any order, then clause gadgets of all clauses in any order of clauses, and finally lower parts of all variable gadgets in any order. We show that each professor needs to move only once.

Consider an arbitrary professor p. When determining the order of timeslots of variable gadgets and clause gadgets, p was processed differently. However, the movement of p was determined according to the value of the literal corresponding to p under D. So, p must move in the same direction within variable gadgets and within clause gadgets. Notice also that in variable gadgets, p does not move within upper parts, or within lower parts. p moves when the time passes from upper parts to lower parts. Since the clause gadgets are sandwiched by upper and lower parts, it is not hard to see that he/she needs to move only once. □

4.3 $\mathcal{NP}$-hardness of Order$(4, 2)$

Theorem 6: Order$(4, 2)$ is $\mathcal{NP}$-hard.

Proof. The reduction is almost the same to the proof of Theorem 5. The only difference is the construction of variable gadgets.

Consider variable x_i in I. Suppose that professors $p_{a,b}$, $p_{c,d}$, and $p_{e,f}$ are introduced for x_i, and $p_{a,b}$ and $p_{c,d}$ correspond to the same polarity of the literal, and $p_{e,f}$ the different polarity. Then we created professor-pairs $[p_{a,b}, p_{e,f}]$, $[p_{e,f}, p_{a,b}]$, $[p_{c,d}, p_{e,f}]$, and $[p_{e,f}, p_{c,d}]$ in the previous reduction. This time, instead of these four, we create two professor-pairs, $[(p_{a,b}, p_{c,d}), p_{e,f}]$ and $[p_{e,f}, (p_{a,b}, p_{c,d})]$, where $(p_{a,b}, p_{c,d})$ means that these two professors are assigned to the same student. This time, each professor appears exactly twice in variable gadgets, and hence four times in total. The correctness proof is obtained by a slight modification of the proof of Theorem 5. □

5. Concluding Remarks

In this paper, we have investigated the time complexity of Room(s, t) and Order(s, t). The apparent next step in this research is to investigate the time complexity of Order$(s, 1)$ for $s = 4, 5$ and Order$(3, t)$ for $t \geq 2$. Further, we may consider the problem allowing both operations for Room and Order, or more generally, the problem without receiving an initial schedule. Finally, considering approximability and inapproximability of these problems would be interesting.

Acknowledgments

The authors would like to thank anonymous referees for their helpful comments. This work was partially supported by KAKENHI (20500017, 17700015, and 16092215). The first author is supported by JSPS Research Fellowships for Young Scientists.

References

[1] P. Berman and M. Karpinski, "Improved approximation lower bounds on small occurrence optimization," ECCC Report, TR03-008, 2003.
[2] M.W. Carter and C.A. Tovey, "When is the classroom assignment problem hard?" Oper. Res., vol.40, no.1, pp.28–39, 1992.
[3] E. Cheng, S. Kruk, and M.J. Lipman, "Flow formulations for the student scheduling problem," Proc. 4th PATAT 2002, LNCS 2740, pp.299–309, 2003.
[4] T.B. Cooper and J.H. Kingston, "The complexity of timetable construction problems," Proc. 1st PATAT 1995, LNCS 1153, pp.283–295, 1996.
[5] H.M.M. ten Eikelder and R.J. Willemen, "Some complexity aspects of secondary school timetabling problems," Proc. 3rd PATAT 2000, LNCS 2079, pp.18–27, 2001.
[6] S. Even, A. Itai, and A. Shamir, "On the complexity of timetabling and multicommodity flow problems," SIAM J. Computing, vol.5, no.4, pp.691–703, 1976.
[7] C.H. Papadimitriou, Computational Complexity, Addison-Wesley Publishing Company, 1994.
[8] T.J. Schaefer, "The complexity of satisfiability problems," Proc. 10th ACM STOC, pp.216–226, 1978.
[9] A. Schaerf, "A survey of automated timetabling," Artificial Intelligence Review, vol.13, no.2, pp.87–127, 1999.
[10] D. de Werra, "Some combinatorial models for course scheduling," Proc. 1st PATAT 1995, LNCS 1153, pp.296–308, 1996.

Naoyuki Kamiyama received B.E. M.E. degrees in Architecture and Architectural Engineering from Kyoto University in 2004, 2006, respectively. He is now a student of doctor course in Kyoto University. His research interests include the design and analysis of combinatorial algorithms, architectural information systems. He is a student member of the Information Processing Society of Japan, Architectural Institute of Japan, and the Operations Research Society of Japan.

Yuuki Kiyonari received B.E. degree from Kyushu Institute of Technology in 2006. He is currently a graduate student of Systems Design and Informatics Department at Kyushu Institute of Technology.

Eiji Miyano is an associate professor of Systems Design and Informatics Department at Kyushu Institute of Technology. Received B.E., M.E., and D.E. degrees in computer science from Kyushu University in 1991, 1993, and 1995, respectively. His research interests are in the area of algorithms and complexity theory. IPSJ, ACM.

Shuichi Miyazaki is an associate professor of Academic Center for Computing and Media Studies, Kyoto University. He received BE, ME, and Ph.D. degrees from Kyushu University in 1993, 1995 and 1998, respectively. His research interests include algorithms and complexity theory.

Katsuhisa Yamanaka is an assistant professor of Graduate School of Information Systems, The University of Electro-Communications. He received B.E., M.E. and Ph.D. degrees from Gunma University in 2003, 2005 and 2007, respectively. His research interests include combinatorial algorithms and graph algorithms.