南山大学大学院
博士（数理科学）論文

A Study on Operations Research Methods to Improve Efficiency of University Administrations
– オペレーションズ・リサーチによる大学業務効率化の研究 –

July 23, 2021

Aino Ohnishi

Graduate Program of Systems and Mathematical Sciences
Graduate School of Science and Engineering
Nanzan University

Abstract

In many universities, the following three tasks are often performed manually by staff: assigning teaching assistants (TAs) to work shifts, assigning classrooms to classes, and arranging books on a shelf in the library. With constraints such as university's equipment and requirements of professors and students, it is complex and time-consuming to complete the tasks.

This research aims to propose practical methods using Operations Research (OR) to solve the above problems of the university, put these methods to practical use, and contribute to the reduction of the burden on the university staff and the cost of university management.

Nanzan University is also facing problems: the shift scheduling problem for TAs, the classroom assignment problem, and the book relocation problem. Therefore, we propose methods, as a concrete example, for solving the problems in Nanzan University: Additionally, we implemented support systems based on the methods for TA shift scheduling and classroom assignment. The actual data of Nanzan University are used for the analysis of the performance of the systems, and some computational experiments were reported to illustrate the behavior of the systems.

Our implemented system can flexibly respond to different conditions and provide an assignment quickly and reduce the workload of the staff. The system uses a model that consists of general conditions applicable to most universities and conditions specific to Nanzan University. Therefore, the system can be easily introduced to other universities only by modifying functions to acquire data from the on-campus database or customizing the system to exclude consider the conditions specific to Nanzan University. Thus, we can contribute to improving the operational efficiency in many universities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The environment surrounding universities has been changing due to the influence of global strategies and the strengthening of the universities' international competitiveness. In this situation, universities are required to meet society's expectations by implementing academic reforms that respond to the changing times and needs. Consequently, universities are required to improve their service levels every year, and the amount of work that university staff need to take is steadily increasing. Taking universities in Japan as an example, we determine that their business environment is becoming severe because of the declining population of 18 year olds, the age of university entrance. In fact, the number of applicants to private universities in 2021 has decreased by about 12%, which will inevitably have a serious impact on university management in the future. In anticipation of a society with a declining 18-year-old population, universities in Japan are beginning to consider collaboration and integration of national, public, and private universities.

To handle the ever-increasing business and improve the operational efficiency of administration and service quality for students, many universities currently use information systems. Several universities have improved their operational efficiency and service quality using Operations Research (OR), and some have developed business support systems for practical use [2, 5, 7, 8, 17, 21].

Nanzan University, the subject of our case studies, has about 170 staff, 9,000 students, and 790 teachers. In the past, OR has been introduced to its management to improve the operational efficiency and service quality [25]. Many assignment-related problems, such as scheduling problem of school bus and school timetable, and the problem of assigning proctors for entrance examinations, have been solved using various techniques. For example, the proctor assignment problem was solved using Linear Programming. The implemented system in existing literature [28] has been used at Nanzan University for 16 years. However, many assignment problems are still solved manually. The staff still spend a large amount of time and effort to solve the following problems:

- Shift scheduling problem for teaching assistants (TAs)
- Classroom assignment problem
- Book relocation problem

These assignment problems relate to properly distributing university resources with consideration of the equipment, capacity, quantity of resources, and requirements of professors, students, and staff. At many universities, the scheduling task mentioned above is manually performed by the staff. The larger the university, the more is the effort required. As we will explain later in Chapters 2 – 4, the problems must be solved considering not only the operational rules and requirements in the university but also the requirements of professors, students, and staff that must be satisfied. Therefore, these problems are complex and extremely difficult and have

burdened the staff for many years. Four years ago, Nanzan University introduced the quarter system, which has also been used by some universities in Japan. The quarter system divides the academic year into four terms (Q1, Q2, Q3, and Q4), where each quarter includes eight weeks. The system further complicates the assignment rules and increases the burden on the staff. This process related to educational support often burdens the limited number of experienced staff members, who attempt to determine a feasible assignment, as if solving a puzzle single-handedly. Therefore, the transfer or retirement of these professionals can cause major problems for the university, such as long working hours and the problem of labor costs.

Owing to the COVID-19 pandemic, the administration of Nanzan University has faced an exceptional situation and transitioned to on-line classes. However, some classes are offered on-campus under certain conditions, such as with a limited number of students in a classroom, in an attempt to curb the widespread of COVID-19. Therefore, as the number of students taking the classes from their home or in the classroom using their own devices has increased, the number of inquiries regarding network services and the amount of TAs work have also increased significantly. Furthermore, depending on the context of the infection spread, the staff need to consider more complex requirements for the classroom capacity when assigning classrooms. In recent years, in Nanzan University Library (NUL), a business support system has just been introduced as part of the university's information system. Meanwhile, the problem is maintaining a variety of books that have not yet been digitized over the years.

Several commercial systems can be used to reduce the effort required to perform an assignment [30, 31, 32, 33, 34]; however, there are concerns regarding the introduction cost, assignment quality, usability, and system maintenance. Staff may not be able to obtain assignments that completely fulfill the specific requirements of professors, students, and staff because the systems do not use OR. In addition, owing to frequently changing professors and students' requirements, the predetermined assignments may need to be changed according to new requirements. Even if commercial systems with maximum functionality are used, modifying the result is not usually supported. Furthermore, additional costs may be incurred for updating the conditions in the system to meet unsupported requirements. Therefore, the staff perform reassignment manually, which takes time and effort.

Thus, assignment support systems using OR have been developed in previous research. However, many universities have not yet used these systems and instead spend a long time manually performing assignments. One of the underlying reasons is that the staff do not have enough OR knowledge. They are, therefore, concerned about system maintenance if they change the conditions for problem-solving. Furthermore, they may not be able to reduce the workload commensurate with the cost of introducing the system.

We apply few conditions of the models proposed in previous research to Nanzan University. However, Nanzan University also has unique conditions; thus, the assignment problem cannot be solved using the model introduced in the literature. Therefore, the original models for solving the problems at Nanzan University must be constructed using OR to reduce the burden on the staff.

This research presents our implemented interactive assignment support systems: the support systems for shift scheduling for TAs and classroom assignment. The systems can provide an assignment quickly and reduce the staff's workload. We implemented the Microsoft Excel[1] and Access[1] systems using the Visual Basic for Applications[1] (VBA) programming language,

the Python programming language, and the Python optimization package, MIPCL. To develop the system, we considered the requirements of the staff, professors, and students. Then, we incorporated these requirements into the system as much as possible. Previous studies have demonstrated that systems are considered practical if they can modify conditions and assignments as necessary [11, 24]. Therefore, we designed the system so that the staff can perform the assignment by changing parameters. The staff can also fix a part of the obtained assignment and reassign it with the revised conditions. We expect the staff to feel comfortable using and accepting the system because they can flexibly respond to different conditions and obtain the final assignment while modifying them several times based on their experience and knowledge.

We will then receive feedback from the staff about improving the effectiveness and user-friendliness of the systems. Moreover, we will continue to improve the systems and verify the quality of the solution, usability, and usefulness.

The remainder of this paper is organized as follows: Chapters 2, 3, and 4 present the shift scheduling problem for TAs, the classroom assignment problem, and the book relocation problem, respectively. In particular, Chapters 3 and 4 are organized as follows: the background of the problem (Section 1), the approach to solve the problem including the description of the constraints and the objective function of the model (Section 2), and the results of computational experiments (Section 3). Additionally, we introduce the shift scheduling support system for TAs in Section 4 of Chapter 2 and the classroom assignment support system in Section 4 of Chapter 3. We conclude the paper in Chapter 5 with conclusions and future works.

Chapter 2

TA Shift Scheduling Support System
– Student Support Desk in Nanzan University –

This chapter explains the TA shift schedule at the student support desk in Nanzan University.

2.1 Background of the TA Shift Scheduling Problem

In Nanzan University, the campus network services (called AXIA: Advanced eXchange for Information Access) are provided to students and professors. The students perform connections to this service using their own devices or PCs in the classrooms. They use the service to register for classes, check their timetables, and submit their reports. However, if the students cannot connect to the network well, about 30 TAs at the student support desk will respond to questions and consultations regarding student network services.

Currently, outsourced staff and experienced TAs (i.e., Schedulers) are the ones assigning TAs at the student support desk in parallel with daily works and support works for students. As illustrated in Figure 2.1, the quarter system divides the academic year into three types of terms: vacation terms, four exam terms, and four class terms, respectively.

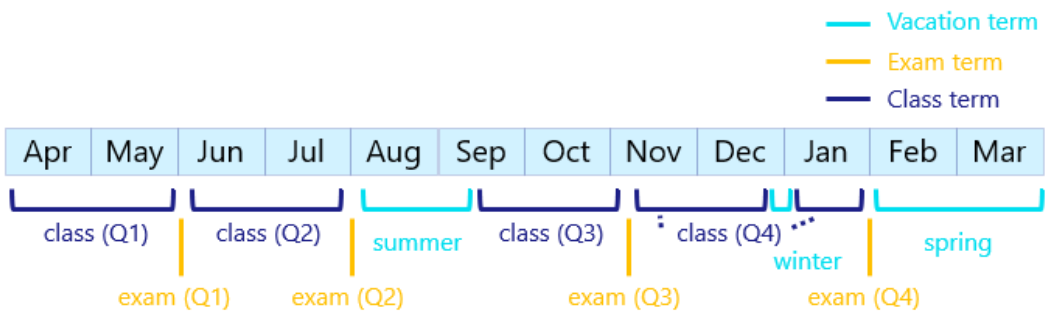

Figure 2.1: The academic calendar in quarter system

The support desk is opened in the following three terms (11 scheduling periods) in a year.

- Vacation term (spring, summer, and winter)
- Exam term (Q1, Q2, Q3, and Q4)
- Class term (Q1, Q2, Q3, and Q4)

Because of the introduction of the quarter system, the schedulers must manually perform the assignments of TAs 11 times a year, which takes time and effort.

We call each working day and each time zone "shift." The TAs repeatedly work for the assigned shifts during each scheduling period. Tables 2.1 – 2.3 present the shifts for each term.

As shown in Tables 2.1 – 2.3, the number of TAs required for each shift is different.

Table 2.1: The vacation term

					Number of TA required						
Shifts	Hours				Mon	Tue	Wed	Thu	Fri	Sat	Sun
Morning (Sat)	9:15	–	13:15	4.0h	2	2	2	2	2	1	0
Afternoon (Sat)	13:15	–	17:15	4.0h	2	2	2	2	2	1	0
Morning	9:15	–	13:15	4.0h	1	1	0	0	1	1	0
Afternoon	13:15	–	17:15	4.0h	1	1	0	0	1	0	0
Third period	13:30	–	15:00	1.5h	0	0	0	0	0	1	0

Table 2.2: The exam term

					Number of TAs required						
Shifts	Hours				Mon	Tue	Wed	Thu	Fri	Sat	Sun
A exam	9:20	–	11:50	2.5h	1	1	1	1	1	0	0
B exam	11:50	–	13:20	1.5h	1	1	1	1	1	0	0
C exam	13:20	–	16:50	3.5h	1	1	1	1	1	0	0
D exam	16:50	–	20:50	4.0h	1	1	1	1	1	0	0

Table 2.3: The class term

					Number of TAs required						
Shifts	Hours				Mon	Tue	Wed	Thu	Fri	Sat	Sun
First period	9:20	–	10:50	1.5h	3	3	3	3	3	0	0
Second period	11:05	–	12:35	1.5h	3	3	3	3	3	0	0
Lunch time	12:30	–	13:30	1.0h	3	3	3	3	3	0	0
Third period	13:30	–	15:00	1.5h	3	3	3	3	3	0	0
Fourth period	15:15	–	16:45	1.5h	3	3	3	3	3	0	0
After class	16:50	–	20:50	4.0h	2	2	2	2	2	0	0
Maintenance	16:50	–	19:20	2.5h	3	0	0	3	0	0	0
Support	17:00	–	18:30	1.5h	1	0	0	1	0	0	0
Morning	9:15	–	13:15	4.0h	0	0	0	0	0	1	0
Afternoon	13:15	–	17:15	4.0h	0	0	0	0	0	1	0

TAs should be assigned to the shifts during the time zones when they do not have classes. The schedulers receive the requirements of the TAs regarding the shifts before performing the assignment. Then, based on the requirements of TAs, they manually assign the TAs to the shifts for each scheduling period, considering the number of shifts and working hours.

The schedulers consider the following conditions when they perform the assignments:

(i) If there are TAs who require the same number of shifts or working hours, the schedulers

assign them to the shifts so that their number of shifts and working hours are equal.

(ii) For TAs who are assigned multiple shifts a day, the staff assign the TAs so that there is no idle time between shifts.

TAs should master their work equally because some TAs are graduates whereas others are beginners. Furthermore, each TA hands over the consultations received from the students to the next support TA at the end of his shift. It takes a bit of time because they need to discuss the type of support the students need. Thus, the schedulers perform the assignment while considering conditions (i) and (ii).

They also consider the following conditions:

- Saturday shifts assigned to each TA should be once or less in each quarter.
- After-school shifts assigned to each TA should be once or less a week in each quarter.
- No more than one beginner TA can be assigned to each shift.

Moreover, the schedulers need to consider a Labor Contract Based on Rules of Employment as follows:

- A TA can work within 7 hours a day.
- A TA can work four days or less a week from Monday to Saturday.

Assigning the TAs to the shifts is performed based on the staff experience; however, it takes a long time to satisfy the above conditions. The staff and 7 TAs have spent about 245 hours manually assigning the class term of Q1 and Q2 in 2018 (including the support times for students). The number of working hours and idle time between shifts have some deviations because the staff perform the assignment manually.

Furthermore, the current method also has difficulty changing the assignment. For example, if we want to modify only a part of the assignment for one TA, we need to review whether the entire assignment violates the conditions visually. In some cases, we may need to modify the entire assignment significantly. The current method is also very inefficient when modifying the assignment.

The shift scheduling problem has often been studied on the assignment decisions, that is, which staff to which work in which time zones and perform an assignment. The methods proposed in these studies are often applied to medical facilities [10, 20, 22, 23, 27], and a typical example is the shift scheduling for nurses [9]. Meanwhile, in many universities, the methods are still not applied to shift scheduling problems, and studies on shift scheduling for TAs are scant. For example, Adrianto [1] proposed particle swarm optimization to solve the problem of scheduling TAs at Binus University. The comparison using particle swarm optimization and genetic algorithm for timetable scheduling was discussed; however, it has not been put to practical use.

A scheduling support system for part-time employees that is similar to TAs with variable requirements has been developed in previous research [26], and several commercial systems also exist [30, 31, 32]. However, they do not take into account conditions (i) and (ii).

Thus, we have proposed a more efficient shift scheduling method. Specifically, we implemented a TA shift scheduling support system that can import the shift requirements gathered

from TAs and perform an assignment based on them. This system can be used to provide an assignment that satisfies the conditions and requirements quickly and reduces the staff's workload.

2.2 Model of the TA Shift Scheduling Problem

This section introduces the formulation of the TA shift scheduling problem as 0-1 integer programming problem.

We can solve the problems of vacation and exam terms using the same formulation. Meanwhile, the problem of class term can be solved by changing some formulations for the problems of vacation and exam terms. We first present a formulation of the vacation and exam terms' problem. Then, we describe the formulation for the class term.

Notation of the problems of vacation and exam terms

Index sets

T : the set of TAs

T_b : the set of beginner TAs, $T_b \subset T$

T_d : the set of dummy TAs

P : the set of days in a scheduling period, $P = E \cup G$

E : the set of days of weekday, $E = \{1, 2, ..., 5\} \subset P$

G : the set of days of Saturday, $G \subset P$

S : the set of shifts, $S = \{1, 2, ..., \overline{s}\}$

S_w : the set of shifts that have an upper bound of the number of times in a weekday, $S_w \subset S$

Parameters

R_t : the assignment priority of TA t (the lower takes priority)

L_s : the working hours of shift s

K : the number of weeks

H^+ : the upper bound of working hours in a day

M^+ : the upper bound of working hours on weekdays

Q^+ : the upper bound of working hours on Saturdays

I^+ : the upper bound of working days in a scheduling period

I^- : the lower bound of working days in a scheduling period

C^+ : the upper bound of the number of shifts in a week

A_t : the number of shift requirements of each TA

B^+ : the upper bound of the number of beginner TAs

N_{ps} : the number of TAs required for shift s on day p

$$O_{tps} = \begin{cases} 1 : \text{if TA } t \text{ can be assigned to shift } s \text{ on day } p \\ 0 : \text{otherwise} \end{cases}$$

$$F_{tps} = \begin{cases} 1 : \text{if TA } t \text{ is fixed to shift } s \text{ on day } p \\ 0 : \text{otherwise} \end{cases}$$

$\alpha,\ \beta,\ \gamma,\ \delta,\ \epsilon,\ \zeta,\ \eta$:
the weights that determine the priority of each term in the objective function,
$\alpha \geq 0, \quad \beta \geq 0, \quad \gamma \geq 0, \quad \delta \geq 0, \quad \epsilon \geq 0, \quad \zeta \geq 0, \quad \eta \geq 0$

Variables

$$x_{tps} = \begin{cases} 1 : \text{if TA } t \text{ is assigned to shift } s \text{ on day } p \\ 0 : \text{otherwise} \end{cases}$$

$$y_{tp} = \begin{cases} 1 : \text{if TA } t \text{ is assigned to shifts on day } p \\ 0 : \text{otherwise} \end{cases}$$

u_t^1 : the variable for relaxing the lower bound of working days in a scheduling period for TA t

u_t^2 : the variable for relaxing the lower bound of the total number of shifts for TA t

v_{tps} : the variable for minimizing idle time between shift s and shift s' on day p for TA t

Formulation of the problems of the vacation and exam terms

$$\begin{aligned} \min . \quad & \sum_{t\in T}\sum_{p\in P} y_{tp} + \alpha \sum_{t\in T_d}\sum_{p\in P}\sum_{s\in S} x_{tps} + \beta \sum_{t\in T}\sum_{p\in P}\sum_{s\in S} R_t x_{tps} \\ & + \gamma \sum_{t\in T} u_t^1 + \delta \sum_{t\in T} u_t^2 + \epsilon \sum_{t\in T}\sum_{p\in P}\sum_{s\in S} v_{tps} \end{aligned} \tag{2.1}$$

$$\text{s.t.} \quad x_{tps} \leq y_{tp}, \quad t \in T, \quad p \in P, \quad s \in S \tag{2.2}$$

$$I^- - u_t^1 \leq \sum_{p\in P} y_{tp} \leq I^+, \quad t \in T \tag{2.3}$$

$$\sum_{p\in P}\sum_{s\in S} x_{tps} \geq A_t - u_t^2, \quad t \in T \tag{2.4}$$

$$x_{tps} - x_{t,p,s+1} \leq v_{tps}, \quad t \in T, \quad p \in P, \quad s \in S \setminus \{\overline{s}\} \tag{2.5}$$

$$\sum_{s\in S} L_s x_{tps} \leq H^+, \quad t \in T, \quad p \in P \tag{2.6}$$

$$\sum_{t\in T_b} x_{tps} \leq B^+, \quad p \in P, \quad s \in S \tag{2.7}$$

$$\sum_{t\in T\cup T_d} x_{tps} = N_{ps}, \quad p \in P, \quad s \in S \tag{2.8}$$

$$x_{tps} \leq O_{tps}, \quad t \in T, \quad p \in P, \quad s \in S \tag{2.9}$$

$$x_{tps} \geq F_{tps}, \quad t \in T, \quad p \in P, \quad s \in S \tag{2.10}$$

$$x_{tps} \in \{0,1\}, \quad t \in T, \quad p \in P, \quad s \in S$$

$$y_{tp} \in \{0,1\}, \quad t \in T, \quad p \in P \tag{2.11}$$

$$u_t^1 \geq 0, \quad u_t^2 \geq 0, \quad t \in T \tag{2.12}$$

$$v_{tps} \geq 0, \quad t \in T, \quad p \in P, \quad s \in S \setminus \{\overline{s}\} \tag{2.13}$$

In the formulation above, we minimize the objective function (2.1) which is the sum of the relaxation variables for constraints (2.2) − (2.5). Considering the requirements of the TAs, we may not be able to assign any TA to a shift, making the problem infeasible. To avoid this, we add virtual dummy TAs with no limit that can be assigned to any shifts. Therefore, the second term of the objective function is the number of shifts that the dummy TAs are assigned. In addition, the sixth term of the objective function assigns the higher priority TAs as much as possible.

As shown in constraint (2.2), the variable y_{dp} is used to control the number of days for each TA in a scheduling period. By using y_{dp}, constraint (2.3) is the upper and lower bounds of the number of days that each TA can be assigned to shifts in a scheduling period. Constraint (2.4) indicates the necessity to satisfy the number of shift requirements for all TAs. Meanwhile, constraint (2.5) means that we minimize idle time between shifts on day d for each TA. Constraint (2.6) means that each TA has a upper bound of working hours in a day. Constraint (2.7) is the upper bound of the number of beginner TAs of shift s on day d. Constraint (2.8) is the number of TAs required for shift s on day d. Constraint (2.9) means that TA t can only be assigned to workable shift s on day d. Constraint (2.10) means that TA t is fixed to shift s on day d. Constraint (2.11) is a binary constraint, while constraints (2.12) and (2.13) are non-negativity constraints.

Each scheduling period of the class term is 8 weeks; however, because the weekday assignment is fixed, the fixed shifts from Monday to Friday is repeated every week. Therefore, the actual scheduling period is 5 days from Monday to Friday and the number of days on Saturday in a quarter. However, we need to consider the Labor Contract Based on Rules of Employment and calculate the number of shifts and working hours for the total number of weeks, even on weekdays. In addition, during the class term, some shifts that have the upper bound of the number of times in a week.

Hence, in the formulation of the problem of class term, we change the objective function (2.1) as follows:

$$\begin{aligned} \min. \quad & \sum_{t\in T}\sum_{p\in P} y_{td} + \alpha \sum_{t\in T_d}\sum_{p\in P}\sum_{s\in S} x_{tds} + \beta \sum_{t\in T}\sum_{p\in P}\sum_{s\in S} R_t x_{tds} \\ & + \gamma \sum_{t\in T} u_t^1 + \delta \sum_{t\in T} u_t^2 + \epsilon \sum_{t\in T}\sum_{p\in P}\sum_{s\in S} v_{tps} + \zeta \sum_{t\in T} u_t^3 + \eta \sum_{t\in T} u_t^4 \end{aligned} \tag{2.14}$$

Then, the following constraints are added to the formulation of the vacation and exam terms.

$$\text{s.t.} \quad \sum_{p\in E}\sum_{s\in S} K(L_s x_{tps}) \leq M^+ + u_t^3,$$

$$\sum_{p\in G}\sum_{s\in S} L_s x_{tps} \leq Q^+ + u_t^4, \quad t \in T \tag{2.15}$$

$$\sum_{p\in P}\sum_{s\in S_w} x_{tps} \leq C^+, \quad t \in T \tag{2.16}$$

$$u_t^3 \geq 0, \quad u_t^4 \geq 0, \quad t \in T \tag{2.17}$$

Additionally, we change constraint (2.3) as follows:

$$I^- - u_t^1 \leq \sum_{p \in E} y_{tp} + y_{tp'} \leq I^+, \quad t \in T, \quad p' \in G \tag{2.18}$$

In the formulation above, we add the sum of the relaxation variables for constraints (2.15) – (2.16) and change the objective function (2.1) as (2.14). Constraint (2.15) is the upper bound of working hours for each TA in a weekday and on Saturday. Constraint (2.16) is the upper bound of the number of times can be assigned to shift $s \in S_w$ in a week. Constraint (2.17) is a non-negativity constraint. Constraint (2.18) is the upper and lower bounds of the number of days that each TA can be assigned to shifts in a week.

2.3 Computational Experiments

This section presents the results of computational experiments using the model introduced in Section 2.2.

We assigned the TAs to the shifts based on the data of the class term at the student support desk at Nanzan University. We obtained an optimal assignment of Q4 in 2018 and assigned 37 TAs (including 4 dummy TAs) to 10 shifts on 5 weekdays and 8 Saturdays. It took 7 seconds to obtain the optimal solution.

We asked the staff to evaluate the assignment of Q3 and Q4 in 2018, and received the following comments: As a result of using the system, compared to their manual work in Q1 and Q2 of the same year, we reduced the working time from 245 hours to about 1/25 of 10 hours. By quickly obtaining an initial assignment that satisfies all the conditions and the requirements, we reduced the load of scheduling and improved the assignment preparation time.

We used CBC (COIN-OR Branch and Cut) as our solver with the model implemented in Python 3.7.2. The assignment was performed on a computer running Microsoft Windows 10[1], with an Intel® Core™ i7-8550U[2] CPU and 16 GB of RAM.

2.4 Implementation of the TA Shift Scheduling Support System

This section introduces a TA shift scheduling support system that we developed using the model introduced in Section 2.2.

We implemented the TA shift scheduling support system on Microsoft Excel using the VBA programming language in Excel, the Python programming language, and the Linear optimization wrapper module, mypulp. Figures 2.2 and 2.3 present the user interface of the system. The staff can operate the system by clicking the buttons on the screen.

We will enter the basic information of shifts: the working days, time zones, and the number of TAs for each shift. Then, we will enter the requirements of TAs. As illustrated in Figure 2.2, we will push the "Import" button and register everything in one bunch using Excel files gathered from TAs. Finally, we will push the "Scheduling" button and solve the formulation that we introduced.

As illustrated in the upper right of Figure 2.3, we designed the system so that staff can perform the assignment by changing the weights that determine the priority of objective function's

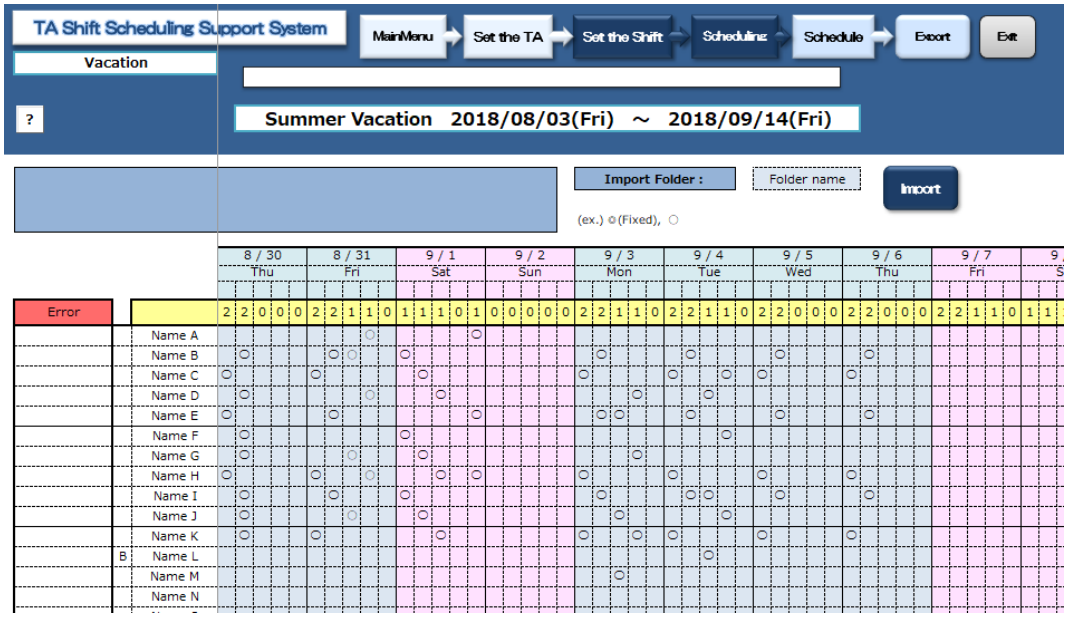

Figure 2.2: The user interface of the TA shift scheduling support system

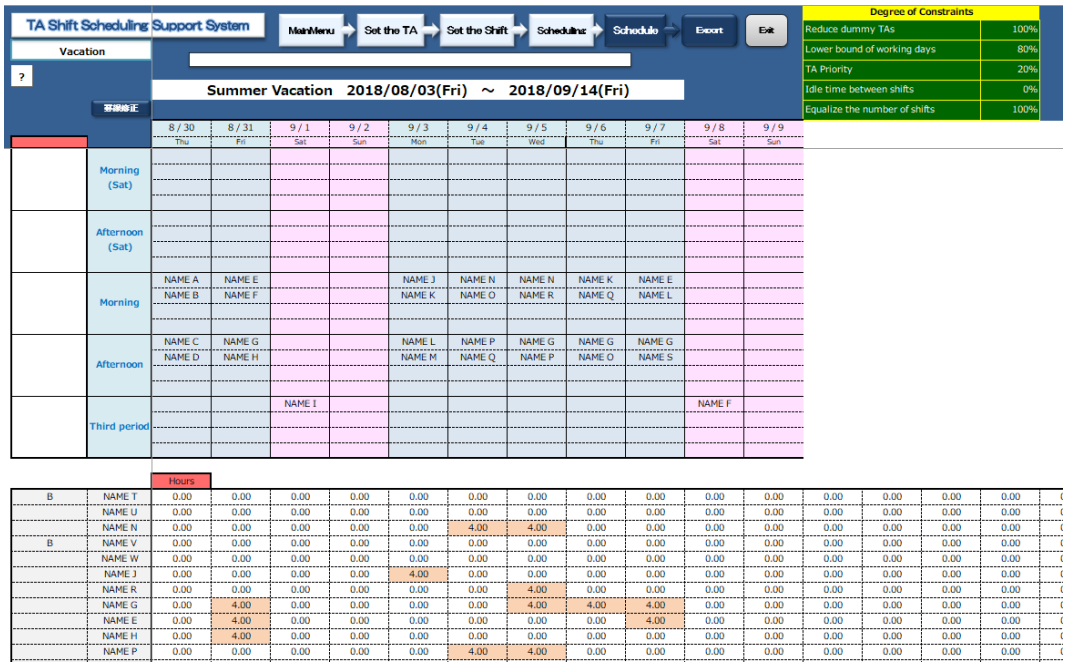

Figure 2.3: The user interface of the TA shift scheduling support system

term. The addition of this feature is the newest requirement from the staff.

If the dummy TAs are assigned to the shifts, the staff prefer not to assign these dummy TAs because they need to revise the conditions and reassign them. Therefore, we set the weight of the second term (the number of shifts to assign the dummy TAs) to the maximum in each term of the objective function.

We designed the system so that staff can assign by changing parameters, such as TAs and shifts. In addition, they can fix a part of the obtained assignment and reassign TAs with the revised conditions.

Chapter 3

Classroom Assignment Support System

This chapter explains the assignment of classrooms for the university timetable.

3.1 Background of the Classroom Assignment Problem

The classroom assignment problem is the problem of assigning classrooms to classes considering the classroom equipment, classroom capacity, number of students taking the classes, and requirements of professors and students for the classes. Not only the operational rules and requirements in the timetable, but also the requirements of professors and students must be met in this problem. Particularly, the professors' equipment requirements and students' desired classes must be considered. Therefore, this problem is complex and extremely difficult; it has been a burden on the staff for many years.

The classroom assignment problem has often been studied as part of the university timetabling problem and reported in the literature as a difficult problem [3, 12]. In the literature, researchers have proposed various methods to solve this problem; however, each university has its own curriculum and course rules. Therefore, proposing a general class assignment model and solution method is difficult. Under these circumstances, universities have considered their unique models. Most researchers have proposed a method for solving the timetable scheduling problem and classroom assignment problem simultaneously. Alternatively, they proposed a hierarchical two-phase solution approach: timetable scheduling followed by classroom assignment.

Nanzan University currently uses a hierarchical two-phase approach for performing timetable and classroom assignment, and few conditions of the models proposed in previous research are also applied. However, Nanzan University also has unique conditions; thus, the assignment problem cannot be solved using the model introduced in the literature. Therefore, an original model must be constructed for solving the classroom assignment problem at Nanzan University using OR to reduce the burden on the staff.

In 2019, researchers [6] formulated the classroom assignment problem based on the newly introduced quarter system of Nanzan University as a 0-1 integer programming problem. Although they solved the problem quickly using Gurobi Optimizer, they have not yet constructed a practical model. In the same year, we formulated the problem as a 0-1 integer programming problem and developed a system to solve the problem using the Python optimization package, MIPCL [19]. However, a long computation time was required to find a solution owing to the large problem size. In fact, we solved the problem of assigning 271 classes to 255 classrooms by using the model, and it takes about 900 seconds. Thus, in 2021, we improved this method and formulated the classroom assignment problem as a linear programming problem to provide a solution in a shorter time.

At Nanzan University, a specific staff member manually performs the timetable and classroom assignment. Although professional staff carefully perform the assignment, a manually performed

assignment is often incomplete, and professors and students are often dissatisfied with it.

This chapter considers a model that assigns classrooms to classes in a given timetable while satisfying strict curriculum constraints. We divide the model's conditions into general conditions applied to almost every university and conditions specific to Nanzan University. Then, we describe the conditions that can be easily applied to other universities and those specific to Nanzan University.

The following describes the conditions that must be considered in the classroom assignment problem.

3.1.1 Common conditions for universities

In many universities, the timetable and classroom assignment are almost the same every year to reduce the staff's burden unless the curriculum is revised. Nevertheless, the assignment must be changed according to the cancellation of classes, the beginning of new classes, changes to the professors of the classes, and changes to the number of registered students.

Classes are offered according to the following rules:

1. Classes are offered in any of the first to fifth periods from Monday to Friday (the first and second periods are in the morning; the third, fourth, and fifth periods are in the afternoon).
2. The academic year consists of two semesters (spring semester and fall semester), and each class is offered 15 times, once a week, in either semester.
3. Each class is offered in a fixed period on a fixed day of the week during the semester. Therefore, the staff must perform a weekly schedule of classroom assignments for each semester.

From these rules, classroom assignments for a period do not affect any other period's assignment. In other words, we can independently consider classroom assignments for each period each day of the week per semester, where each assignment has the following two constraints:

(i) One classroom should be assigned to each class.

(ii) One classroom cannot be assigned to more than one class at the same time.

In addition to these constraints, many universities also consider the following conditions.

(iii) Each professor has requirements for the equipment in the assigned classrooms.
- The staff must assign classrooms to classes considering those requirements as the equipment of each classroom differs, such as the types of seats and audiovisual equipment.

(iv) Every classroom has a capacity limit.
- After students register for or change classes, if the number of registered students exceeded classroom capacity, the staff must modify the assignment immediately.

3.1.2 Specific conditions at Nanzan University

Under the quarter system at Nanzan University, some classes offer two lectures per week to complete 15 lectures in a quarter, which we call "twin classes." We further classify twin classes into two types according to when the two lectures are offered in a week. One type offers two lectures in consecutive periods on the same day of a week, while the other offers two lectures in the same period on a different day of the week. We call these two types of classes "serial classes" and "parallel classes," respectively. In addition, we call a class that offers one lecture per week a "single class." Note that any class is not offered on Wednesday afternoon at Nanzan University.

- Serial classes are offered according to one of the following rules:
  - First and second periods on any day from Monday to Friday
  - Third and fourth periods on any day from Monday to Friday except Wednesday
- Parallel classes are offered according to one of the following rules:
  - Same period on Mondays and Thursdays
  - Same period on Tuesdays and Fridays

Figure 3.1 presents an example of the weekly timetable. Each symbol represents one class, and each alphabetical letter in parentheses represents a professor. If more than one professor leads a class, one of them is considered the representative professor of the class.

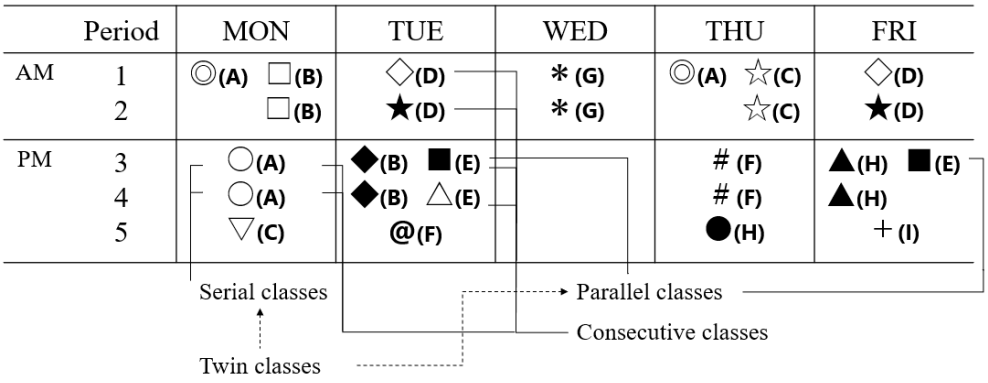

Figure 3.1: Example of a weekly timetable for classes

As illustrated in Figure 3.1, the classes that are offered by the same representative professor in consecutive periods are called "consecutive classes." At Nanzan University, although the staff perform a classroom assignment while considering up to three requirements of the professor (e.g., a whiteboard, a projector, and a lecture theater), much time and effort is required to satisfy all these requirements.

Additionally, the same classroom or nearby classrooms must be assigned to not only parallel classes, but also consecutive classes. This is because the staff must consider both the professors' requirements and the time it takes for the professor to travel to the next class. However, two classrooms located far away from each other are occasionally assigned to classes because the staff cannot find an appropriate assignment. If a class is both a parallel class and a consecutive class, we consider the class to be a consecutive class. For example, we must assign the same classroom or nearby classrooms to the following classes (Figure 3.1):

- Class ○ in the third and fourth periods on Mondays by professor A
- Class ◇ and class ★ in the first and second periods on Tuesdays by professor D
- Class ■ and class △ in the third and fourth periods on Tuesdays by professor E

The staff prepare an assignment based on the number of students registered in the previous year before the registration time period of each quarter. Students can register for the classes they want to take twice a year and change them four times a year before each quarter. More precisely, the number of students changes in the following two times:

1. The time period to register for classes (during the three days immediately before Q1 and Q3): Students register for the classes they want to take.
2. The time period to change the registered classes (for one week immediately after the start of each quarter): After the classes start, students who are unsatisfied with their registered classes can cancel them and register for different classes.

During time periods 1 and 2, the staff frequently check whether there are any classes in which the number of students exceeds the classroom capacity. If the staff discover a capacity shortage, they must seek a new assignment that satisfies the capacity requirements. In time period 1, the staff may easily reassign other classrooms to classes because the classes have not yet started. However, in time period 2, the staff cannot easily do so because classes have already started. If the staff start reassigning classrooms to classes in time period 1, they must complete the assignment quickly before starting. If the staff cannot assign the classrooms, they must limit the number of students in the classes. Consequently, some students may not be able to take the classes they want.

With the current assignment procedure, reassigning classrooms is not easy. Specifically, when no vacant classrooms are available for reassignment, the staff must find a candidate classroom for exchange among classrooms that are already assigned to another class. This procedure often leads to subsequent exchange until all conditions are satisfied, which is inefficient.

3.2 Model of the Classroom Assignment Problem

This section introduces the formulation of the classroom assignment problem as a 0-1 integer programming problem, which we first constructed. Then, we introduce the formulation as a linear programming problem.

Much computational time may be required to assign classrooms to all classes owing to the large problem size. Therefore, we divide the problem into small problems and solve the problems step by step. Thereby, we can solve the problems quickly, modify the assignment, and change the conditions easily. As mentioned in Section 3.1, under the semester system, we can independently consider classroom assignments for each period per day; however, we must assign the same classroom to twin classes under the quarter system. Instead, dividing the problem into each period per day, we divide it into the following seven problems:

Problem 1 : The first and second periods on Mondays and Thursdays

Problem 2 : The third and fourth periods on Mondays and Thursdays

Problem 3 : The fifth period on Mondays and Thursdays

Problem 4 : The first and second periods on Tuesdays and Fridays

Problem 5 : The third and fourth periods on Tuesdays and Fridays

Problem 6 : The fifth period on Tuesdays and Fridays

Problem 7 : The first and second periods on Wednesdays

Note that these problems are independent; that is, the solution of a problem does not affect other problems' solutions.

If the problem is infeasible, the classroom assignment support system returns an error message. To avoid this, we add virtual classrooms with no capacity limit that can be assigned to any classes. We refer to these virtual classrooms as dummy classrooms.

This problem must satisfy the following hard constraints:

(c.1) One of the classrooms (including dummy classrooms) is assigned to each class.

(c.2) One classroom cannot be assigned to more than one class simultaneously.

(c.3) A classroom with sufficient capacity must be assigned to accommodate all the students registered for the class.

Additionally, this problem satisfies the following soft constraints as much as possible:

(c.4) A class is assigned to the same classroom as in the previous year as often as possible.

(c.5) Twin classes led by the same professor are assigned to the same classroom or nearby classrooms as often as possible.

(c.6) A class is assigned to a classroom that satisfies the equipment requirements of the professor as often as possible.

(c.7) Classes are assigned to dummy classrooms as infrequently as possible.

First, we represent the formulation of the classroom assignment problem as a 0-1 integer programming problem using Problem 1 (morning classes on Monday and Thursday) as an example. We can formulate other problems (Problems 2 to 7) in the same manner.

Notation

Index sets

C^1 : the subset of one class of consecutive classes

C^2 : the subset of the other class of consecutive classes

C_m : the subset of one class of parallel classes

C_t : the subset of the other class of parallel classes

C^3 : the subset of classes other than C^1, C^2, C_m, and C_t

C : the set of classes, $\quad C = C^1 \cup C^2 \cup C_m \cup C_t \cup C^3$

I : the set of groups of the classroom

R : the subset of available classrooms

R_d : the set of dummy classrooms

R_i : the subset of classrooms of group i, $\quad i \in I$

P_m : the set of periods on Monday, $\quad P_m = \{p_m^1, p_m^2\}$

P_t : the set of periods on Thursday, $\quad P_t = \{p_t^1, p_t^2\}$

P : the set of periods, $\quad P = P_m \cup P_t$

Parameters

$$O_{cp} = \begin{cases} 1 : \text{if class } c \text{ will be offered at period } p \\ 0 : \text{otherwise} \end{cases}$$

$$A_{rp} = \begin{cases} 1 : \text{if classroom } r \text{ is available at period } p \\ 0 : \text{otherwise} \end{cases}$$

$$E_{cr} = \begin{cases} 1 : \text{if classroom } r \text{ fulfills the professor's equipment requirements of class } c \\ 0 : \text{otherwise} \end{cases}$$

$$B_{ri} = \begin{cases} 1 : \text{if classroom } r \text{ belongs to the group } i \\ 0 : \text{otherwise} \end{cases}$$

$$L_{crp} = \begin{cases} 1 : \text{if class } c \text{ was offered at period } p \text{ in classroom } r \text{ in the previous year} \\ 0 : \text{otherwise} \end{cases}$$

$$Q_{cc'} = \begin{cases} 1 : \text{if class } c' \text{ is the other class of consecutive class } c \\ 0 : \text{otherwise} \end{cases}$$

$$T_{cc'} = \begin{cases} 1 : \text{if class } c' \text{ is the other class of parallel class } c \\ 0 : \text{otherwise} \end{cases}$$

$$D_{ii'} = \begin{cases} 1 : \text{if classroom of group } i \text{ and classroom of group } i' \text{ are reachable} \\ 0 : \text{otherwise} \end{cases}$$

Cap_r : capacity of classroom r

N_c : number of students who registered for class c

$$F_{crp} = \begin{cases} 1 : \text{if class } c \text{ is fixed to classroom } r \text{ at period } p \\ 0 : \text{otherwise} \end{cases}$$

$\alpha,\ \beta,\ \gamma,\ \delta,\ \epsilon$:
the weights that determine the priority of each term in the objective function,
$\alpha \geq 0, \quad \beta \geq 0, \quad \gamma \geq 0, \quad \delta \geq 0, \quad \epsilon \geq 0$

Variables

$$x_{crp} = \begin{cases} 1 : \text{if class } c \text{ is assigned at period } p \text{ to classroom } r \\ 0 : \text{otherwise} \end{cases}$$

$$g_{ci} = \begin{cases} 1 : \text{if class } c \text{ is assigned to classroom of group } i \\ 0 : \text{otherwise} \end{cases}$$

y_{crp} : penalty for class c at period p to assign to the different classroom r from the previous year

$u_{cc'}$: penalty for class c not to assign to the classroom which is assigned the consecutive class c'

$v_{cc'}$: penalty for class c not to assign to the classroom which is assigned the parallel class c'

Formulation

$$\begin{aligned}
\min. \quad & \alpha \sum_{c\in C}\sum_{r\in R_d}\sum_{p\in P} x_{crp} + \beta \sum_{c\in C}\sum_{r\in R}\sum_{p\in P} y_{crp} + \gamma \sum_{c\in C^1}\sum_{c'\in C^2} u_{cc'} + \delta \sum_{c\in C_m}\sum_{c'\in C_t} v_{cc'} \\
& + \epsilon \sum_{c\in C}\sum_{r\in R} (Cap_r - N_c) \sum_{p\in P} x_{crp} && (3.1)
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{r\in R} x_{crp} = O_{cp}, \quad c \in C, \quad p \in P && (3.2) \\
& \sum_{c\in C} x_{crp} \le 1, \quad r \in R, \quad p \in P && (3.3) \\
& \sum_{c\in C} x_{crp} \le A_{rp}, \quad r \in R, \quad p \in P && (3.4) \\
& \sum_{p\in P} x_{crp} \le E_{cr}, \quad c \in C, \quad r \in R && (3.5) \\
& \sum_{c\in C} N_c x_{crp} \le Cap_r, \quad r \in R, \quad p \in P && (3.6) \\
& x_{crp} \ge F_{crp}, \quad c \in C, \quad r \in R, \quad p \in P && (3.7) \\
& -y_{crp} \le (L_{crp} - x_{crp}) \le y_{crp}, \quad s \in S, \quad r \in R, \quad p \in P && (3.8) \\
& -u_{cc'} \le Q_{cc'} \left(x_{crp} - x_{c'rp'}\right) \le u_{cc'}, \\
& \quad c \in C^1, \quad c' \in C^2, \quad r \in R, \quad p \in P_m, \quad p' \in P_t && (3.9) \\
& -v_{cc'} \le T_{cc'} \left(x_{crp} - x_{c',r,p+1}\right) \le v_{cc'}, \\
& \quad c \in C_m, \quad c' \in C_t, \quad r \in R, \quad p \in \{p_m^1, p_t^1\} && (3.10) \\
& B_{ri} \sum_{p\in P} x_{crp} \le g_{ci}, \quad c \in C, \quad r \in R, \quad i \in I \\
& D_{ii'} \left(g_{ci} + g_{c'i'}\right) Q_{cc'} \le 1, \quad c \in C^1, \quad c' \in C^2, \quad i \in I, \quad i' \in I && (3.11) \\
& x_{crp} \in \{0,1\}, \quad c \in C, \quad r \in R, \quad p \in P && (3.12) \\
& y_{crp} \in \{0,1\}, \quad c \in C, \quad r \in R \cup R_d, \quad p \in P && (3.13) \\
& u_{cc'} \in \{0,1\}, \quad c \in C^1, \quad c' \in C^2 && (3.14) \\
& v_{cc'} \in \{0,1\}, \quad c \in C_m, \quad c' \in C_t && (3.15) \\
& g_{ci} \in \{0,1\}, \quad c \in C, \quad i \in I && (3.16)
\end{aligned}$$

In the formulation above, we minimize the objective function (3.1) which is the sum of the number of classes assigned to dummy classrooms, the three penalties, and the difference between the number of students and the classroom capacity. The first penalty is the sum of y_{crp}, defined in (3.8) as the penalty for assigning class c to the same classroom r as in the previous year. The second penalty is the sum of u_{c_1,c_2}, defined in (3.9) as the penalty for assigning classes c_1 and c_2 which are the consecutive classes to the same classroom. The third penalty is the sum of v_{c_1,c_2}, defined in (3.11) as the penalty for assigning classes c_1 and c_2, which are the parallel classes to the same classroom. Constraint (3.2) means that class c should be assigned to single classroom

r at each period p. Constraint (3.3) means that at most single class c should be assigned to classroom r at each period p. Constraint (3.4) means that class c should be assigned to only available classroom r. Constraint (3.5) indicates that class c should be assigned to classroom r, which satisfies the equipment requirements of the professor. Constraint (3.6) means that class c should be assigned to classroom r where the number of students does not exceed the classroom capacity. Constraint (3.7) means that class c should be assigned to classroom r fixed at period p. Constraint (3.8) means that class c should be assigned to the same classroom r as in the previous year as often as possible. Constraint (3.9) means that classes c_1 and c_2, which are the consecutive classes should be assigned to the same classrooms r. Constraint (3.10) means that classes c_1 and c_2, which are parallel classes, should be assigned to the same classrooms r. Constraint (3.11) means that classes c_1 and c_2, which are consecutive classes, should be assigned to classrooms of groups i_1 and i_2, which are reachable. Constraints (3.12) – (3.16) are binary constraints.

Next, we represent the formulation of the classroom assignment problem as a linear programming problem. We describe our technique on formulating the classroom assignment problem as a minimum cost flow problem using Problem 1 (morning classes on Monday and Thursday) as an example (Figure 3.2). We can formulate other problems (Problems 2 – 7) in the same manner.

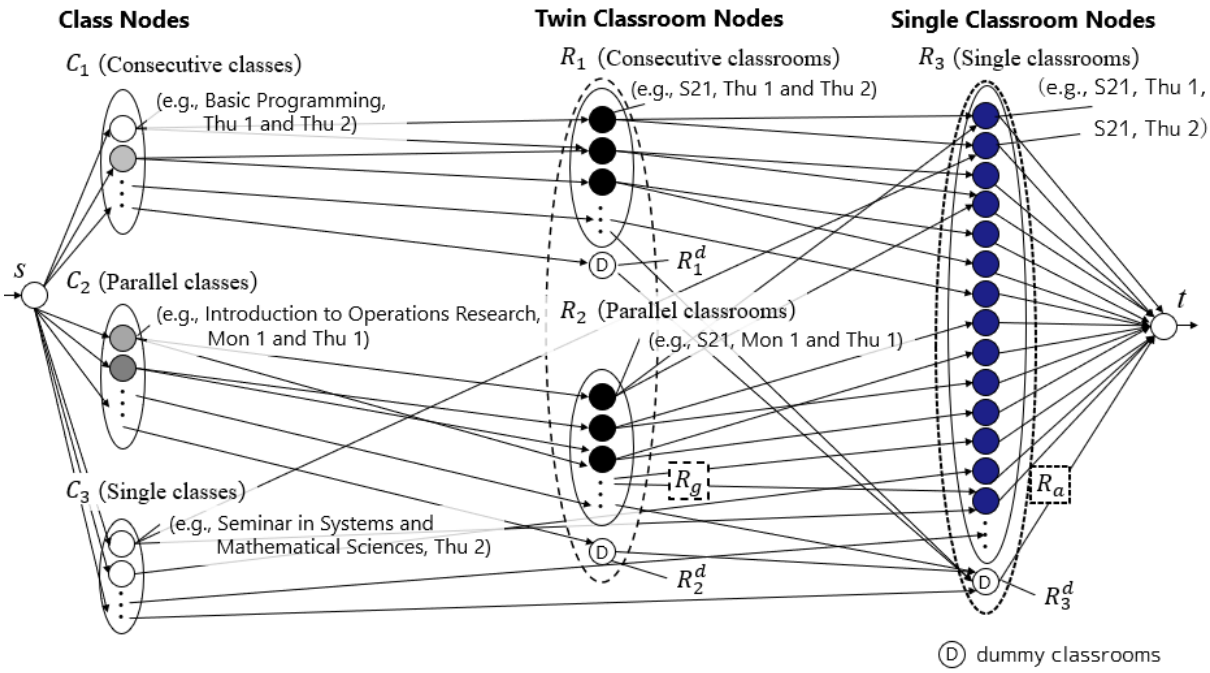

Figure 3.2: Example of Problem 1 network

As illustrated in Figure 3.2, the network has three layers, each corresponding to different types of nodes: class nodes, twin classroom nodes, and single classroom nodes. Each class node in the leftmost layer corresponds to a class, whereas each single classroom node in the rightmost layer corresponds to a classroom. These two types of nodes are essential components for classroom assignments. Additionally, we introduce twin classroom nodes as artificial nodes in the middle layer. Introducing the twin classroom nodes and setting adequate edge costs allows us to flexibly incorporate various soft constraints.

The class nodes in the leftmost layer consist of three sets of nodes, namely, C_1, C_2, and C_3,

which correspond to consecutive, parallel, and single classes, respectively. C_1 and C_2 are nodes to which we must assign the same classroom or nearby classrooms. In this way, if we need to assign nearby classrooms to two classes, we set the classes as a node and introduce a set of nodes. By setting adequate costs for edges between the node and twin classroom nodes, we can assign nearby classrooms to classes. For example, this is applied to consecutive classes, which are offered by the same professor in consecutive periods. This is also applied to parallel classes to avoid student confusion.

The single classroom nodes in the rightmost layer consist of two types of sets: R_3 and R_3^d. A "single classroom" is an available classroom in a period on a day of the week. For example, if classroom S21 is available in the first period on Mondays, we prepare a node (S21, Mon1) as a node in R_3. We also prepare dummy classroom nodes as nodes in R_3^d. R_a consists of all nodes in R_3 and R_3^d.

Meanwhile, a classroom that is available twice a week is called a "twin classroom." We further classify twin classrooms into "consecutive classrooms" and "parallel classrooms," which can be used by consecutive classes and parallel classes, respectively.

- Consecutive classrooms are available in the following periods:
  - First and second periods on any day from Monday to Friday
  - Third and fourth periods on any day from Monday to Friday except Wednesday
- Parallel classrooms are available in the following periods:
  - Same period on Mondays and Thursdays
  - Same period on Tuesdays and Fridays

For the twin classroom nodes in the middle layer, four sets of nodes exist, namely, R_1, R_2, R_1^d, and R_2^d, which correspond to consecutive classrooms, parallel classrooms, dummy classrooms for consecutive classrooms, and dummy classrooms for parallel classes, respectively. If classroom S21 is available in the first and second periods on Thursdays, we prepare a node (S21, (Thu1 and Thu2)) as a node in R_1. Similarly, if classroom S21 is available in the first period both on Mondays and Thursdays, we prepare a node (S21, (Mon1 and Thu1)) as a node in R_2. We also prepare dummy classroom nodes as nodes in R_1^d and R_2^d. R_g denotes all nodes in R_1, R_2, R_1^d, and R_2^d.

In addition to the aforementioned nodes, two special nodes exist: a source node s and sink node t. We connect s and all nodes in C_1 and C_2 with a capacity of 2, and all nodes in C_3 with a capacity of 1. We must assign the nodes in C_1 and C_2 to the same classroom as often as possible owing to constraint (c.5). To achieve this, we must ensure that each class node in C_1 is connected to twin classroom nodes in R_1, and each class node in C_2 is connected to twin classroom nodes in R_2. For example, if the class node "Basic Programming" is scheduled in the first and second periods on Thursdays, we connect the node to all nodes (*, (Thu1 and Thu2)) in R_1, where the wildcard * represents all classrooms that have sufficient capacity for the class.

Each twin classroom node in R_1 and R_2 has two edges corresponding to two single classroom nodes in R_3. For example, node (S21, (Thu1 and Thu2)) in R_1 connects to two nodes (S21, Thu1) and (S21, Thu2) in R_3.

We also connect each class node in C_3 and single classroom nodes in R_3. For example, if the class node "Seminar in Systems and Mathematical Science" is scheduled in the second period

on Thursdays, we connect the node to all nodes (*, (Thu2)) in R_3 that satisfy the capacity requirements. Then, all single classroom nodes in R_3 are connected to t with a capacity of 1.

The flow conservation law holds at all nodes, except for s and t, and flow the total number of classes from s to t. To avoid the unsolvable problem, we must ensure that each class node and the twin classroom nodes are connected to dummy classroom nodes, and all dummy classroom nodes are connected to t. A feasible $s-t$ flow corresponds to a feasible classroom assignment that satisfies all hard constraints (c.1), (c.2), and (c.3).

As mentioned earlier, we must set adequate costs for edges between class nodes and twin classroom nodes to achieve a desirable assignment. Note that flow from a class node to a twin classroom node indicates that a classroom corresponding to the twin classroom node is assigned to the class. Therefore, we set a low (high) cost for edges associated with a desirable (undesirable) assignment. For example, at Nanzan University, we set the lowest cost for edges leading to a node associated with the same classroom as in the previous year because it is the best assignment. In addition, we may set a slightly higher but sufficiently low cost for edges associated with a classroom located close to the same classroom as in the previous year. In contrast, we set a high cost for edges corresponding to an assignment to a classroom that does not fulfill the professor's requirements regarding equipment. Also, we set the highest cost to edges leading to a dummy classroom node to reduce the number of dummy classrooms that are assigned to classes. The flow from a class node to a twin classroom node may be split in two. Even in that case, we set adequate costs for edges to assign the classes to nearby classrooms as possible.

We define the cost associated with edges as follows. For the following edges, we reduce the cost of the edges to make it easier to assign a classroom to a class:

- edges to the same classroom as in the previous year
- edges to classrooms located sufficiently close to the previous year's classroom

Additionally, for the following edges, we increase the cost of the edges in an effort to avoid assigning the classroom to a class:

- edges to classrooms that do not fulfill the professor's equipment requirements
- edges to dummy classrooms

Note that we set different values for the costs of all these edges. Therefore, the flow on all edges with integer capacity is integer values.

In the following, we present the formulation of the classroom assignment problem.

Notation

Index sets

s : the source node

t : the sink node

C_1 : the set of nodes of consecutive classes

C_2 : the set of nodes of parallel classes

C_3 : the set of nodes of single classes

R_1 : the set of nodes of consecutive classrooms

R_2 : the set of nodes of parallel classes

R_3 : the set of nodes of single classrooms

R_1^d, R_2^d, R_3^d : the set of nodes of dummy classrooms

R_g : the set of nodes of consecutive classrooms, parallel classes, and dummy classrooms, $R_1 \cup R_2 \cup R_1^d \cup R_2^d = R_g$

R_a: the set of single classrooms and dummy classrooms, $R_3 \cup R_3^d = R_a$

Parameters

a_{cr} : the capacity of edge (c, r), $c \in C_1$, $r \in R_1$

b_{cr} : the capacity of edge (c, r), $c \in C_2$, $r \in R_2$

m_{cr} : the capacity of edge (c, r), $c \in C_3$, $r \in R_3$

$n_{rr'}$: the capacity of edge (r, r'), $r \in R_1 \cup R_2$, $r' \in R_3$

o_{rt} : the capacity of edge (r, t), $r \in R_a$, $t \in T$

w_{cr}^1 : the cost of edge (c, r), $c \in C_1$, $r \in R_1 \cup R_1^d$

w_{cr}^2 : the cost of edge (c, r), $c \in C_2$, $r \in R_2 \cup R_2^d$

w_{cr}^3 : the cost of edge (c, r), $c \in C_3$, $r \in R_a$

v_{sc} : the flow of edge (s, c), $c \in C_1 \cup C_2 \cup C_3$
2 for all $c \in C_1 \cup C_2$, 1 for all $c \in C_3$

Variables

x_{cr} : the flow of edge (c, r), $c \in C_1$, $r \in R_1 \cup R_1^d$

y_{cr} : the flow of edge (c, r), $c \in C_2$, $r \in R_2 \cup R_2^d$

z_{cr} : the flow of edge (c, r), $c \in C_3$, $r \in R_a$

$u_{rr'}$: the flow of edge (r, r'), $r \in R_g$, $r' \in R_a$

q_{rt} : the flow of edge (r, t), $r \in R_a$, $t \in T$

Formulation

$$\min . \quad \sum_{(c,r),\ c\in C_1,\ r\in R_1\cup R_1^d} w_{cr}^1 x_{cr} + \sum_{(c,r),\ c\in C_2,\ r\in R_2\cup R_2^d} w_{cr}^2 y_{cr} + \sum_{(c,r),\ c\in C_3,\ r\in R_a} w_{cr}^3 z_{cr} \tag{3.17}$$

$$\text{s.t.} \quad x_{cr} \le a_{cr}, \quad c \in C_1, \quad r \in R_1 \tag{3.18}$$

$$y_{cr} \le b_{cr}, \quad c \in C_2, \quad r \in R_2 \tag{3.19}$$

$$z_{cr} \le m_{cr}, \quad c \in C_3, \quad r \in R_3 \tag{3.20}$$

$$u_{rr'} \le n_{rr'}, \quad r \in R_1 \cup R_2, \quad r' \in R_3 \tag{3.21}$$

$$q_{rt} \le o_{rt}, \quad r \in R_a, \quad t \in T \tag{3.22}$$

$$v_{sc} = \sum_{r \in R_1 \cup R_1^d} x_{cr}, \quad c \in C_1 \tag{3.23}$$

$$v_{sc} = \sum_{r \in R_2 \cup R_2^d} y_{cr}, \quad c \in C_2 \tag{3.24}$$

$$v_{sc} = \sum_{r \in R_a} z_{cr}, \quad c \in C_3 \tag{3.25}$$

$$\sum_{(c,\ r),\ c \in C_1} x_{cr} = \sum_{(r,\ r'),\quad r' \in R_a} u_{rr'}, \quad r \in R_1 \cup R_1^d \tag{3.26}$$

$$\sum_{(c,\ r),\ c \in C_2} y_{cr} = \sum_{(r,\ r'),\quad r' \in R_a} u_{rr'}, \quad r \in R_2 \cup R_2^d \tag{3.27}$$

$$\sum_{(c,\ r'),\ c \in C_3} z_{cr'} + \sum_{(r,\ r'),\quad r \in R_g} u_{rr'} = \sum_{(r',\ t),\quad t \in T} q_{r't}, \quad r' \in R_a \tag{3.28}$$

$$x_{cr} \geq 0, \quad c \in C_1, \quad r \in R_1 \cup R_1^d \tag{3.29}$$

$$y_{cr} \geq 0, \quad c \in C_2, \quad r \in R_2 \cup R_2^d \tag{3.30}$$

$$z_{cr} \geq 0, \quad c \in C_3, \quad r \in R_a \tag{3.31}$$

$$u_{rr'} \geq 0, \quad r \in R_g, \quad r' \in R_a \tag{3.32}$$

$$q_{rt} \geq 0, \quad r \in R_a, \quad t \in T \tag{3.33}$$

In the formulation above, we minimize the objective function (3.17) which is the sum of the cost of the flows, whereas constraints (3.18) – (3.22) are the flow capacity constraints for edges. In constraints (3.18) – (3.22), if the number of registered students exceeds the classroom capacity, we set the capacity of the edge to 0. Constraints (3.23) – (3.28) are flow conservation laws, whereas constraints (3.29) – (3.33) are non-negativity constraints.

3.3 Computational Experiments

This section presents the results of computational experiments using the model introduced in Section 3.4.

First, as a trial calculation, we assigned classrooms to classes based on the Nanzan University timetable data from the 2019 academic year. The parameter values and calculation results in the cases examined are presented in Table 3.1.

We initially implemented a system that assigned the same classroom or nearby classrooms to twin classes as often as possible. However, in addition to twin classes, we determined that we also had to assign the same classroom or nearby classrooms to all classes led by the same professor in consecutive periods. This is based on the opinions of the staff responsible for the assignment of classrooms in 2019. Therefore, we improved the system to consider not only twin classes, but also all consecutive classes, and we received feedback from the staff that the assignments were consistent with their intentions.

During the COVID-19 pandemic, the administration of Nanzan University has faced an exceptional situation. In 2020, because all classes were conducted on-line, the staff did not need to assign classrooms. However, Nanzan University offers three types of classes in 2021: on-line classes, hybrid classes that use both on-line and face-to-face settings, and face-to-face classes. For the hybrid classes, the staff have to change the classroom capacity to one-half or one-third

Table 3.1: Size and central processing unit (CPU) time of the classroom assignment problems in the cases of 2019

Terms : Q1	Problems						
	1	2	3	4	5	6	7
$\|C_1\|$	139	137	0	132	111	0	98
$\|C_2\|$	103	100	14	80	79	7	0
$\|C_3\|$	58	130	27	103	160	23	64
#variables	866,246	924,971	743,607	891,581	930,440	737,266	810,404
#constraints	867,936	926,661	744,874	893,271	932,130	738,533	811,671
Times (sec)	441	518	192	404	503	193	342
Set-up times	332	508	191	402	501	192	341
CPU times	9	10	1	3	2	1	1
Total times (problems 1–7)	2,239						
Terms : Q2	Problems						
	1	2	3	4	5	6	7
$\|C_1\|$	127	83	0	112	110	0	68
$\|C_2\|$	73	87	10	77	76	4	0
$\|C_3\|$	43	125	30	67	107	25	46
#variables	835,805	892,405	744,450	851,432	883,963	737,687	782,504
#constraints	837,495	894,095	745,717	853,122	885,653	738,954	783,771
Times (sec)	270	372	161	299	367	165	200
Set-up times	268	366	160	293	365	164	199
CPU times	2	6	1	6	2	1	1
Total times (problems 1–7)	1,587						
Terms : Q3	Problems						
	1	2	3	4	5	6	7
$\|C_1\|$	136	128	0	126	108	0	92
$\|C_2\|$	94	93	11	88	72	6	0
$\|C_3\|$	51	130	34	85	153	28	54
#variables	855,255	918,203	748,253	877,217	920,295	741,068	799,416
#constraints	856,945	919,893	749,520	878,907	921,985	742,335	800,683
Times (sec)	290	424	227	333	433	160	248
Set-up times	287	421	226	330	427	159	247
CPU times	3	3	1	3	6	1	1
Total times (problems 1–7)	2,006						
Terms : Q4	Problems						
	1	2	3	4	5	6	7
$\|C_1\|$	127	102	0	126	118	0	73
$\|C_2\|$	86	96	16	74	67	8	0
$\|C_3\|$	53	124	25	94	143	33	57
#variables	849,754	903,404	742,763	878,900	913,960	746,139	793,914
#constraints	851,444	905,094	744,030	880,590	915,650	747,406	795,181
Times (sec)	287	405	188	343	435	164	316
Set-up times	285	403	187	341	428	163	315
CPU times	2	2	1	2	7	1	1
Total times (problems 1–7)	1,829						
$\|R_1\|$			422				
$\|R_2\|$			422				
$\|R_3\|$			844				
$\|R_1^d\|$			1				
$\|R_2^d\|$			1				
$\|R_3^d\|$			1				

and assign classrooms under more complex conditions than before. At the request of the staff, we have improved the system so that they can change the classroom capacity.

We assigned classrooms to classes based on the data from Q1 of 2021. Table 3.2 presents the

calculation results in the cases examined. If a class is pre-determined to be on-line or to use a particular classroom (e.g., a PC room), the class is excluded from the set of classes.

Table 3.2: Size and central processing unit (CPU) time of the classroom assignment problems in the cases of 2021

Terms : Q1	Problems						
	1	2	3	4	5	6	7
$\|C_1\|$	129	95	0	114	90	0	71
$\|C_2\|$	46	82	8	50	65	6	0
$\|C_3\|$	62	143	20	115	187	20	57
(c.4)	50	95	4	59	99	7	43
(c.5)	38	58	0	36	65	0	16
(c.6)	0	0	0	0	0	0	0
(c.7)	10	16	0	16	0	0	6
#variables	727,589	791,632	629,490	764,681	817,394	628,708	683,020
#constraints	729,151	793,194	630,661	766,243	818,956	629,879	684,191
Times (sec)	257	389	132	325	509	147	197
Set-up times	256	388	131	324	508	146	196
CPU times	1	1	1	1	1	1	1
Total times (problems 1–7)	1,766						

As indicated in Tables 3.1 and 3.2, we found optimal solutions for each network flow problem quickly. In Table 3.2, (c.4) is the number of classes assigned to the same classroom as in the previous year. Meanwhile, (c.5) is the number of twin classes not assigned to the same classroom. (c.6) represents the number of classes assigned to classrooms that do not satisfy the equipment requirements of professors, and (c.7) represents the number of classes assigned to dummy classrooms.

We could assign classrooms to most classes, and only a few classes were assigned to dummy classrooms. In addition, we were able to assign classrooms to all classes that satisfy the professors' equipment requirements. Meanwhile, in Q1 of 2021, the hybrid classes used less than half the classroom capacity of the previous year. Therefore, soft constraint (c.4) was not satisfied. As shown in Table 3.2, many classes are assigned to different classrooms from the previous year. Furthermore, as representative professors often change, their equipment requirements may change; hence, the classroom in the previous year may not satisfy their requirements. In an exceptional situation like in 2021, we need to set the cost of edges so that the impact of soft constraint (c.4) is small because we may not be able to satisfy soft constraints (c.5) and (c.6) owing to soft constraint (c.4). However, at Nanzan University, professors basically request the same classroom as in the previous year. Therefore, soft constraint (c.4) is necessary for the ordinary academic year.

We used MIPCL version 2.5.4 (win64) as our solver with the model implemented in Python 3.7.4. The assignment was performed on a computer running Microsoft Windows 10, with an Intel® Core™ i7-8550U CPU and 16GB of RAM.

3.4 Implementation of the Classroom Assignment Support System

This section introduces a classroom assignment support system that we redeveloped using the model introduced in Section 3.2. We implemented the interactive classroom assignment

support system using Microsoft Access VBA (Figure 3.3). The staff can operate the system by clicking the buttons on the screen.

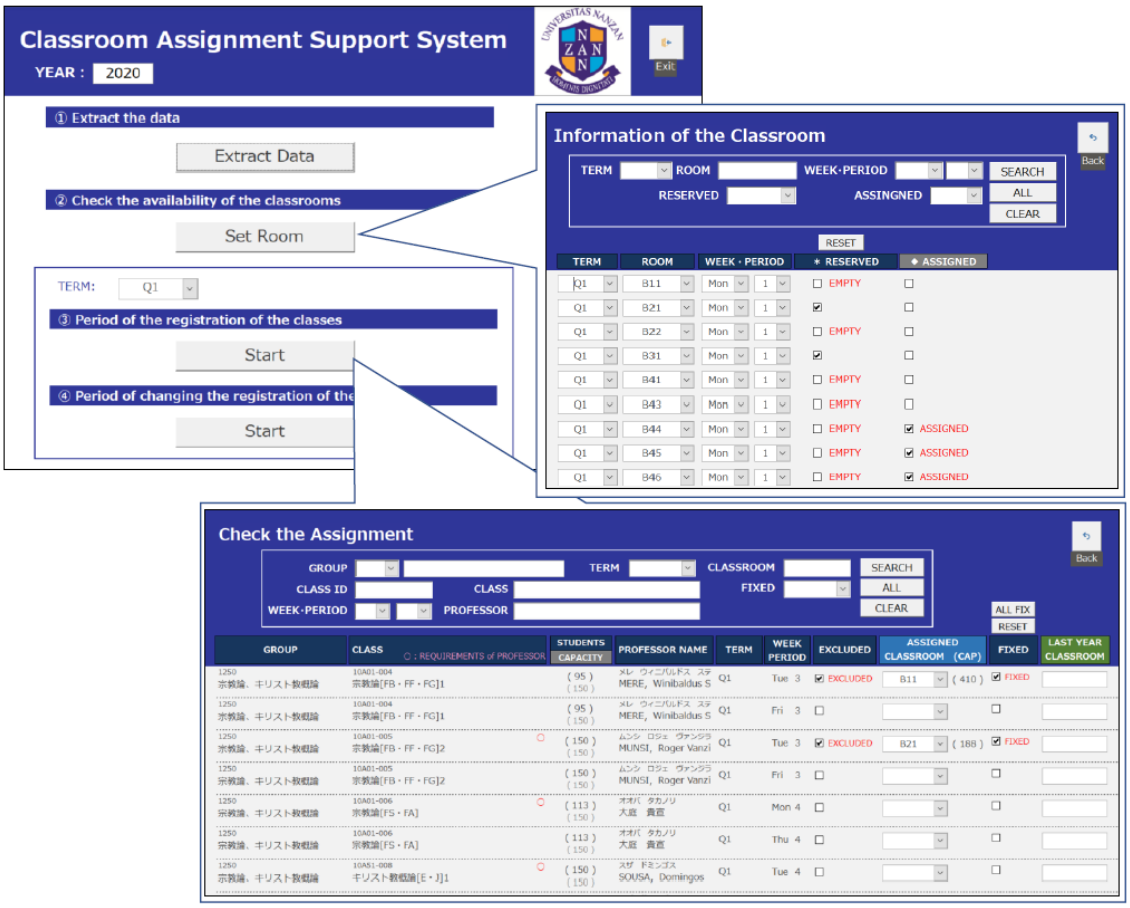

Figure 3.3: The user interface of the classroom assignment support system

Clicking the "Extract Data" button connects to the on-campus database, and data regarding the timetable, classrooms, and professors are acquired from approximately 20 data tables in the database presented in Figure 3.4.

year	lecture_cd	kaiko_keitai_id	day_of_week_id	period_id	room_cd	employee_cd
2020	10A01-001	61	12	14	G26	102874
2020	10A01-001	61	15	14	G26	102874
2020	10A01-002	61	11	13	MB1	000167
2020	10A01-002	61	14	13	MB1	000167
2020	10A01-003	61	11	13	Q101	103281
2020	10A01-003	61	14	13	Q101	103281
2020	10A01-004	51	12	13	MB12	101180
2020	10A01-004	51	15	13	MB12	101180
2020	10A01-005	51	12	13	B21	101925
2020	10A01-005	51	15	13	B21	101925
2020	10A01-006	51	11	14	S47	101347
2020	10A01-006	51	14	14	S47	101347
2020	10A01-007	81	12	14	M2	104315
2020	10A01-007	81	15	14	M2	104315

Figure 3.4: Example of part of the database

Each field in the table is described as follows:

- lecture_cd: class code

- kaiko_keitai_id: ID of the term (e.g., “51” and “61” correspond to Q1 and Q2, respectively)
- day_of_week_id: ID of the day of the week (e.g., “11” and “12” correspond to Monday and Tuesday, respectively)
- period_id: ID of the period (e.g., “13” and “14” correspond to the third and fourth periods, respectively)
- room_cd: assigned classroom’s name
- employee_cd: professor’s code

Next, by clicking the “Set Room” button, we can set whether each classroom can be made available. Then, by clicking the “Start” button, the system assigns classrooms to classes. When clicking the “Start” button, we can also select a few problems from Problems 1 to 7. The upper “Start” button is for the time period to register for classes, whereas the lower “Start” button is for the time period to change registered classes. In the case of the lower “Start” button (in time period 2), constraint (c.4) is as follows, whereas the others are the same as in time period 1:

(c.4)’ A class is assigned to the same classroom as in “time period 1” as often as possible.

If classes are assigned to dummy classrooms, the staff revise the conditions, reassign the classroom, or may manually assign vacant classrooms to the classes.

We designed the system so that staff can perform the assignment by changing parameters, such as the offered classes and professor requirements. In addition, they can fix a part of the obtained assignment and reassign classrooms with the revised conditions.

Chapter 4

Optimal Relocation Plan for Improving Management of Bookshelves

This chapter explains the book relocation in NUL, where the shortage of bookshelves is the most serious problem.

4.1 Background of the Book Relocation Problem

University libraries have improved the quality of the stored resources and continued to increase their number. Moreover, the university education and research have developed based on them [15]. Recently, the mixture of different types of information resources has been promoted at university libraries: traditional resources (paper), digital resources, and stored resources in other institutions including foreign countries [29].

Several university libraries in Japan have also used information systems, and they have been improving the management and operation of books and services to library users by using the systems. However, because many university libraries continue to accept and accumulate books by buying or being donated, they face the shortage of bookshelves [15, 16]. Therefore, the libraries secure vacant sections of bookshelves by moving a part of the books to off-campus storage, adding new bookshelves, or disposing the books.

In the libraries, the books are classified, and the bookshelves are divided into the sections of each book classification number to place books on the bookshelves quickly. As shown in Figure 4.1, each classified book is located from the left end of the shelf. Moreover, the right-hand side of the shelf is kept vacant for the margin of the future accepted books. In the daily

Figure 4.1: An example of actual vacant shelves

operational level, the libraries relocate the books frequently to keep the bookshelf margin of each classification appropriately.

In NUL, we place about 700,000 books in the order of classification numbers on the bookshelves. As explained above, the space of one section consists of the shelves for the existing books and the vacant shelves for the future accepted books of the classification number. We must place newly accepted books on the vacant shelves in their classification number sections. These vacant shelves are determined by library staff based on the number of books of each classification number accepted in the past years. However, because the forecast of the number of the future accepted books is not accurate, the staff need to adjust the vacant shelves frequently. If we cannot place new books on the bookshelves, the vacant shelves of the classification number must be increased by relocating the books and reducing the number of vacant shelves for other classification numbers. However, libraries must maintain accessibility to books for the library users without any problem. In the literature, researchers have analyzed the seriousness of book misplacement and developed a model considering the trade-off in which the library users want to minimize their effort in finding books and the library staff want to minimize its cost of keeping the shelves in order relative to a specified service level [4]. Their study also shows that if books are in perfect order of classification numbers, the library users will usually access books without difficulty. Therefore, to keep access of books for the library users, we have to relocate the books while maintaining the placement order of books based on the classification number.

In NUL, the library staff plan the relocation of the books every year. So far, they are making the plan of the adjustment by hand considering the conditions as explained above, and they spend much time every time. Based on the plan, they move the books during the summer vacation, when there are relatively few library users. Sometimes, the plan includes the move of many books because each year's plan is not so commendable, and the imbalance is accumulating. In this case, more library staff move the books, which is time-consuming and takes much hard work.

NUL has rearranged the bookshelves and removed the books in recent years; they moved about 270,000 books to the off-campus storage in 2017. Consequently, the occupancy of bookshelves in the library improved from 78% (March 2015) to 62% (March 2020). However, if we continue to accept the books every year, the occupancy of bookshelves will exceed 70% in a few years on the second and fourth floors, which are dedicated for the students in the library. The occupancy of bookshelves on all floors is expected to reach 75% by 2036. If the occupancy of bookshelves exceeds 70%, accepting new books is generally difficult [13].

For the rearrangement, the staff need to consider the following two factors:

(i) Keep enough bookshelf margin for each classified book group

Due to the limited capacity of bookshelves and the imbalance of the number of books bought in the future between the classified book groups, the staff need to move the classified book group that is increasing rapidly to the shelves with more vacant shelves.

(ii) The distance of the move of the classified book groups

The staff need to move the books by hand, which is an enormous burden for the library management. They make a plan of the relocation to minimize the distance of the move; however, it takes time and effort.

Although the book relocation is a problem that should be solved in many libraries, research suggesting the optimal solution method is scarce. Hitotsubashi University Library uses the Toshokan Hikkoshi Rakuraku Kit they developed [14] for book relocation. The application is reported to be useful; however, this application does not use OR to make the plan of the relocation. It only provides an assignment using a manually created plan by the staff. It still takes time to make the plan and the move distance is not minimal.

Thus, we developed a model to relocate the books more efficiently. Our goals are to make the plan automatically using the techniques and decrease the move of the books to reduce the burden of the library staff.

4.2 Model of the Book Relocation Problem

This section introduces the formulation of the book relocation problem as a mixed-integer programming problem.

NUL categorizes books into N book groups, denoted by G, and gives a classification number to each group. The bookshelves in NUL are divided into sections, each of which consists of one or more consecutive shelves. Each section is assigned to one of the book groups to appear in the bookshelves according to their classification numbers. Due to this manner, G is regarded as an ordered set $G = \{1, 2, \ldots, N\}$, which corresponds to the classification numbers.

At present, each section has not been filled with books yet; some vacant shelves are remaining for books to be arranged in the section in the future. We suppose that the books in a book group $i \in G$ already occupy a space of width B_i in its section. Moreover, statistics by NUL, revealed that book group i consumes more b_i width every year (Figure 4.2). Hence, book group i will need a space of width $B_i + nb_i$ in n years.

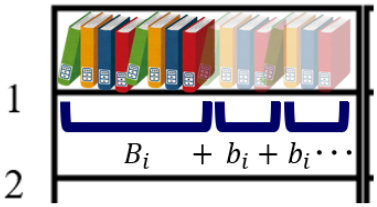

Figure 4.2: An example of a space of width occupied by books in a book group

If the vacant shelves in a section are not enough, we have to widen the section (i.e., add one or more shelves to the section), which may require relocation of many books to keep the order of G. Rearranging books is exhausting; therefore, we need a good plan for dividing bookshelves into sections with adequate shelves for each book group.

In addition, we need to consider that NUL has several types of bookshelves. In this paper, we deal with the case of two groups of bookshelves: bookshelf groups 1 and 2. They are denoted by $S = \{1, 2\}$. For each $j \in S$, bookshelf group j consists of $K^{(j)}$ bookshelves, which are numbered from 1 to $K^{(j)}$, and each of which is an $L^{(j)}$-level bookshelf. The shelves of a bookshelf in bookshelf group $j \in S$ are also numbered from the highest one, and hence the $L^{(j)}$th shelf means the lowest shelf of the bookshelf. Regardless of the bookshelf groups, all the shelves have the same width, normalized to be 1 (Figure 4.3). Hence, parameters B_i and b_i for $i \in G$ are also

converted.

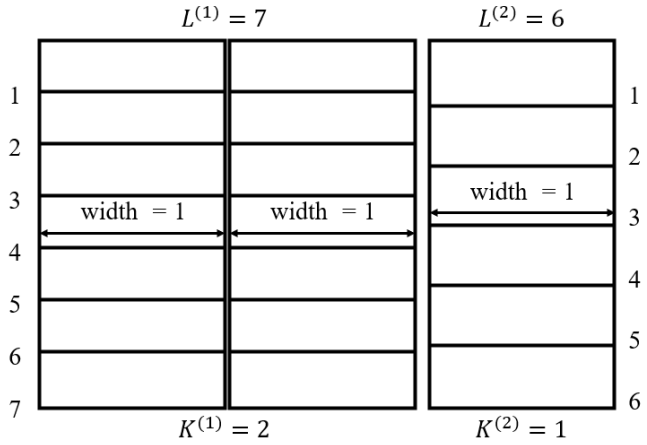

Figure 4.3: An example of bookshelf groups

At present, bookshelf groups S are consistent with book groups G in the following sense: The first N' book groups $G^{(1)} = \{1, 2, \ldots, N'\}$ have their sections in bookshelf group 1, and the last $N - N'$ book groups $G^{(2)} = \{N'+1, N'+2, \ldots, N\}$ in bookshelf group 2. Such a consistency must hold after relocation; that is, N'' exists, such that book groups i with $i \leq N''$ and with $i > N''$ are assigned to a section in bookshelf groups 1 and 2, respectively.

Because of these settings, the positions of the sections are represented as follows: For each bookshelf group j and each book group i in $G^{(j)}$, we use a pair of integers (k, l) to show that the section for book group i starts with the lth shelf of the kth bookshelf in bookshelf group j. The current position of the section for each book group $i \in G$ is assumed to be (k_i^0, l_i^0). Note that, for a book group $i \in G^{(j)}$, we have $k_i^0 \in \{1, 2, \ldots, K^{(j)}\}$ and $l_i^0 \in \{1, 2, \ldots, L^{(j)}\}$. Moreover, if book groups i and $i+1$ both have their sections in the same bookshelf group j, the total width of the section for i can be calculated by $(k_{i+1}^0 L^{(j)} + l_{i+1}^0) - (k_i^0 L^{(j)} + l_i^0) = (k_{i+1}^0 - k_i^0)L^{(j)} + l_{i+1}^0 - l_i^0$.

Our problem is basically determining new positions of sections. It is allowed that, after the relocation, some book groups in $G^{(1)}$ are assigned to sections in bookshelves in bookshelf group 2, and vice versa. Then, we prepare pairs of integer variables $(k_i^{(1)}, l_i^{(1)})$ and $(k_i^{(2)}, l_i^{(2)})$ with

$$k_i^{(j)} \in \{1, 2, \ldots, K^{(j)} + 1\}, \quad i \in G \cup \{N+1\}, \quad j \in S,$$
$$l_i^{(j)} \in \{1, 2, \ldots, L^{(j)}\}, \quad i \in G \cup \{N+1\}, \quad j \in S,$$

for each book group $i \in G = G^{(1)} \cup G^{(2)}$ and $i = N+1$, and we define variables $w_i^{(1)}$ and $w_i^{(2)}$ as

$$w_i^{(1)} = (k_{i+1}^{(1)} - k_i^{(1)})L^{(1)} + l_{i+1}^{(1)} - l_i^{(1)}$$

and

$$w_i^{(2)} = (k_{i+1}^{(2)} - k_i^{(2)})L^{(j)} + l_{i+1}^{(2)} - l_i^{(2)}.$$

In addition, we use the following binary variables:

$$x_i^{(j)} = \begin{cases} 1 & \text{if book group } i \text{ has its new section in bookshelf group } j \\ 0 & \text{otherwise} \end{cases} \quad i \in G, \quad j \in S$$

together with constraints:

$$w_i^{(j)} \leq K^{(j)} L^{(j)} x_i^{(j)}, \quad i \in G, \quad j \in S,$$
$$x_i^{(1)} + x_i^{(2)} = 1, \quad i \in G.$$

Then, the width of the new section for book group i is given by either $w_i^{(1)}$ or $w_i^{(2)}$. Note that $w_i^{(1)} = 0$ require that $k_{i+1}^{(1)} = k_i^{(1)}$ and $l_{i+1}^{(1)} = l_i^{(1)}$. Hence, by setting $k_{N+1}^{(1)} = K^{(1)} + 1$ and $l_{N+1}^{(1)} = 1$, we have (c1) $k_i^{(1)} = K^{(1)} + 1$ and $l_i^{(1)} = 1$ if $x_i^{(1)} = 0$; that is, i is not assigned to a section in bookshelf group 1. Similarly, by introducing the following technical constraints:

$$k_i^{(2)} \leq 1 + (K^{(2)} - 1) x_i^{(2)}, \quad i \in G,$$
$$l_i^{(2)} \leq 1 + (L^{(2)} - 1) x_i^{(2)}, \quad i \in G,$$

we have, as a consequence of $x_i^{(2)} = 0$, (c2) $k_i^{(2)} = 1$ and $l_i^{(2)} = 1$ and moreover, $k_{i+1}^{(2)} = 1$ and $l_{i+1}^{(2)} = 1$ if i is not assigned to a section in bookshelf group 2.

We now discuss the objective function of our model. Our main aim is to divide bookshelves into sections so that new books still have a margin in each section in n years, which will avoid a drastic relocation. At the same time, we want to minimize the difference between the current positions of the sections and their new positions because a small rearrangement is also tiring. For simplicity, we relate the tiredness to the sum of distances to carry books from the current positions to the newly assigned position. The rearrangement task is assumed to be composed of the following three steps: (1) take books down on the floor, (2) carry the books to their new bookshelf (if needed), and then (3) put the books on their new shelf; see Figure 4.4.

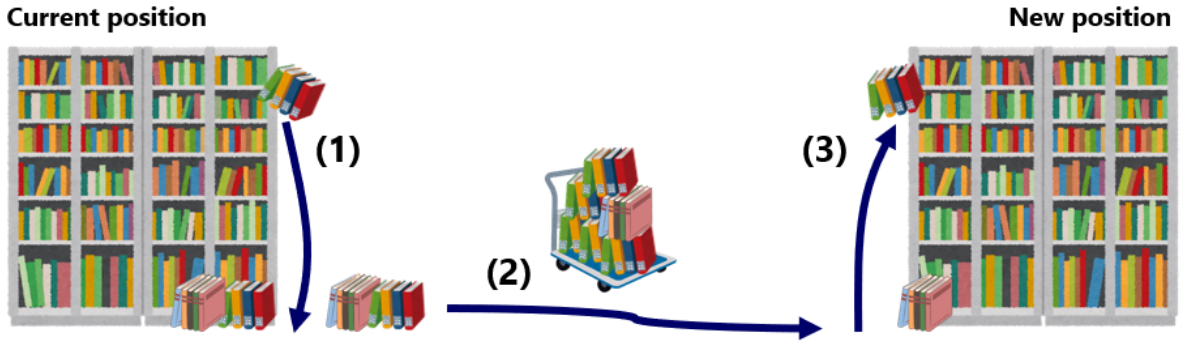

Figure 4.4: The distance to carry books from the current positions to the newly assigned position

To measure the distances in this task, we introduce integer variables t_i, t_i' and t_i'' for each book group $i \in G$, although we omit to explain their constraints here. The variable t_i denotes the distance between the current bookshelf and the new one, which might be 0. Meanwhile, the variable t_i' is basically equal to the difference between the current shelf and the new one, which is not directly included in the objective function; however, it is used to control the following binary variable:

$$y_i = \begin{cases} 1 & \text{if book group } i \text{ changes the position of its section} \\ 0 & \text{otherwise} \end{cases} \qquad i \in G$$

By using this variable y_i, the height to put books down on the floor is given by $y_i(L^{(1)} - l_i^0)$ for $i \in G^{(1)}$ and $y_i(L^{(2)} - l_i^0)$ for $i \in G^{(2)}$. The variable t_i'' denotes the height of the shelf for the books.

Finally, we incorporate the number of books to carry, which is preferred to be small, into the objective function by employing a multiplying factor $f(B_i)$ for each book group $i \in G$. Then, our problem is formulated as follows:

Notation

Index sets

N : the number of categorized book groups

G : the set of book groups, $\quad G = \{1, 2, ..., N\}$

S : the set of bookshelf groups, $\quad S = \{1, 2\}$

Parameters

B_i : the space of width currently occupied by book group i

$f(B_i)$: the number of books in book group i to carry

b_i : the space of width consumed by book group i a year

n : the number of years

$K^{(j)}$: the number of bookshelves of bookshelf group j

$L^{(j)}$: the number of levels of bookshelf group j

k_i^0 : the current bookshelf position to start placement for book group i in bookshelf group j

l_i^0 : the current shelf position to start placement for book group i in bookshelf group j

Variables

$$x_i^{(j)} = \begin{cases} 1 : \text{if book group } i \text{ has its new section in bookshelf group } j \\ 0 : \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 : \text{if book group } i \text{ changes the position of its section} \\ 0 : \text{otherwise} \end{cases}$$

$k_i^{(j)}$: the bookshelf position to start newly placement for book group i in bookshelf group j

$l_i^{(j)}$: the shelf position to start newly placement for book group i in bookshelf group j

$w_i^{(j)}$: the space of width of the new section for book group i in bookshelf group j

t_i : the distance between the current bookshelf and the new bookshelf

t_i' : the difference between the current shelf and the new shelf

t_i'' : the height of the shelf to put books on

Formulation

$$\min. \quad \sum_{i\in G^{(1)}} f(B_i)\{t_i + y_i(L^{(1)} - l_i^0) + t_i'')\} + \sum_{i\in G^{(2)}} f(B_i)\{t_i + y_i(L^{(2)} - l_i^0) + t_i'')\} \tag{4.1}$$

$$\text{s.t.} \quad w_i^{(j)} = (k_{i+1}^{(j)} - k_i^{(j)})L^{(j)} + l_{i+1}^{(j)} - l_i^{(j)}, \quad i \in G, \quad j \in S \tag{4.2}$$

$$w_i^{(j)} \le K^{(j)} L^{(j)} x_i^{(j)}, \quad i \in G, \quad j \in S \tag{4.3}$$

$$B_i + nb_i \le w_i^{(1)} + w_i^{(2)}, \quad i \in G \tag{4.4}$$

$$\sum_{i\in G}(B_i + nb_i) \le K^{(1)}L^{(1)} + K^{(2)}L^{(2)} \tag{4.5}$$

$$k_i^{(j)} \le k_{i+1}^{(j)}, \quad i \in G, \quad j \in S \tag{4.6}$$

$$L^{(1)}(k_i^{(1)} - k_{i+1}^{(1)}) + x_{i+1}^{(1)} \le l_{i+1}^{(1)} - l_i^{(1)}, \quad i \in G \tag{4.7}$$

$$L^{(2)}(k_i^{(2)} - k_{i+1}^{(2)}) + x_i^{(2)} \le l_{i+1}^{(2)} - l_i^{(2)}, \quad i \in G \tag{4.8}$$

$$x_i^{(1)} + x_i^{(2)} = 1, \quad i \in G \tag{4.9}$$

$$x_i^{(1)} \ge x_{i+1}^{(1)}, \quad i \in G \tag{4.10}$$

$$x_i^{(2)} \le x_{i+1}^{(2)}, \quad i \in G \tag{4.11}$$

$$k_i^0 - k_i^{(1)} \le t_i, \quad i \in G^{(1)} \tag{4.12}$$

$$k_i^{(2)} - x_i^{(1)} + k_i^{(1)} - k_i^0 \le t_i, \quad i \in G^{(1)} \tag{4.13}$$

$$k_i^0 - k_i^{(2)} + K^{(1)} + 1 - k_i^{(1)} \le t_i, \quad i \in G^{(2)} \tag{4.14}$$

$$k_i^{(2)} - k_i^0 \le t_i, \quad i \in G^{(2)} \tag{4.15}$$

$$l_i^0 - l_i^{(j)} \le t_i', \quad i \in G^{(j)}, \quad j \in S \tag{4.16}$$

$$l_i^{(j)} - l_i^0 \le t_i', \quad i \in G^{(j)}, \quad j \in S \tag{4.17}$$

$$\frac{t_i}{2(K^{(1)} + K^{(2)})} + \frac{t_i'}{2L^{(j)}} \le y_i, \quad i \in G^{(j)}, \quad j \in S \tag{4.18}$$

$$y_i \le t_i + t_i', \quad i \in G \tag{4.19}$$

$$L^{(j)}x_i^{(j)} - l_i^{(j)} \le t_i'' + L^{(j)}(1 - y_i), \quad i \in G^{(j)}, \quad j \in S \tag{4.20}$$

$$k_i^{(2)} \le 1 + (K^{(2)} - 1)x_i^{(2)}, \quad i \in G \tag{4.21}$$

$$l_i^{(2)} \le 1 + (L^{(2)} - 1)x_i^{(2)}, \quad i \in G \tag{4.22}$$

$$k_i^{(j)} \in \{1, 2, \ldots, K^{(j)} + 1\}, \quad i \in G, \quad j \in S \tag{4.23}$$

$$k_{N+1}^{(j)} = K^{(j)} + 1, \quad j \in S \tag{4.24}$$

$$l_i^{(j)} \in \{1, 2, \ldots, L^{(j)}\}, \quad i \in G, \quad j \in S \tag{4.25}$$

$$l_{N+1}^{(j)} = 1, \quad j \in S \tag{4.26}$$

$$x_i^{(j)} \in \{0, 1\}, \quad i \in G, \quad j \in S \tag{4.27}$$

$$y_i \in \{0, 1\}, \quad i \in G \tag{4.28}$$

$$w_i^{(j)} \in \mathbb{Z}_+, \quad i \in G, \quad j \in S \tag{4.29}$$

$$t_i, \quad t_i', \quad t_i'' \in \mathbb{Z}_+, \quad i \in G \tag{4.30}$$

The definitions of the constraints, which have not been provided yet, are as follows. Constraint (4.4): Book group i still have enough space in its section in n years. Constraint (4.5): All book groups must be arranged on the bookshelves in bookshelf groups 1 and 2 in n years. Constraint (4.6): The bookshelf for book group i has to be the same as or anterior to that for book group $i+1$. Constraints (4.7) and (4.8) (together with constraint (4.6)): If book groups i and $i+1$ are assigned to the same bookshelf, then their sections have to start with distinct shelves. Constraints (4.10) (resp., (4.11)): If a book group $i+1$ (resp., i) is assigned to a section in bookshelf group 1 (resp., 2), then book group i (resp., $i+1$) is also assigned to a section in bookshelf group 1 (resp., 2). Constraints (4.12) and (4.13): For a book group $i \in G^{(1)}$, the distance between its current bookshelf and its new bookshelf is equal to t_i with the minimum value satisfying these two constraints. Note that, by (c2) condition, $k_i^{(2)} = x_i^{(1)} = 1$ if book group i is assigned to a section in the bookshelf group 1; otherwise $k_i^{(2)} - x_i^{(1)} = k_i^{(2)}$ and, by (c1) condition, $k_i^{(1)} = K^{(1)} + 1$. Constraints (4.14) and (4.15): For a book group $i \in G^{(2)}$, the distance between its current bookshelf and its new bookshelf is equal to t_i with the minimum value satisfying these two constraints. Note that, by (c1) condition, $k_i^{(1)} = K^{(1)} + 1$ if book group i is assigned to a section in the bookshelf group 2. Constraints (4.16) and (4.17): If book group i changes its shelf, t_i' has to be positive. Constraint (4.18) and (4.19): If a book group changes its section, binary y_i has to be positive (i.e., 1); otherwise y_i has to be 0. Constraint (4.20): If book group i changes its section, t_i'' has to be at least the height of the (uppermost) shelf to put book group i on; otherwise t_i'' can be 0.

4.3 Computational Experiments

This section presents the results of computational experiments using the model introduced in Section 4.2.

We obtained an optimal relocation of the books of actual data of a part of the second floor of NUL.

The books are about Social Science with classification numbers from 300 to 330; they are categorized into 182 book groups. Table 4.1 shows the minimum, maximum, average, and sum of their B_i and b_i. The bookshelves are on the second floor of NUL (Figure 4.5).

Table 4.1: Actual data in Nanzan University Library

	Minimum	Maximum	Average	Sum
B_i	0.025	30.6	3.26	593.6
b_i	0	0.947	0.09	16.4

The bookshelves are highly occupied. Each bookshelf has seven levels. The bookshelf groups 1 and 2 consist of 52 and 70 bookshelves, respectively. Hence, we have 854 shelves in total. Therefore, the bookshelves have a margin for at most $(854 - 593.6)/16.4 = 15.8$ years. We set $n = 8$, moderately.

We used MIPCL version 2.5.4 (win64) as our solver with the model implemented in Python 3.7.4. The computation was done on a computer running Microsoft Windows 10 with an Intel®

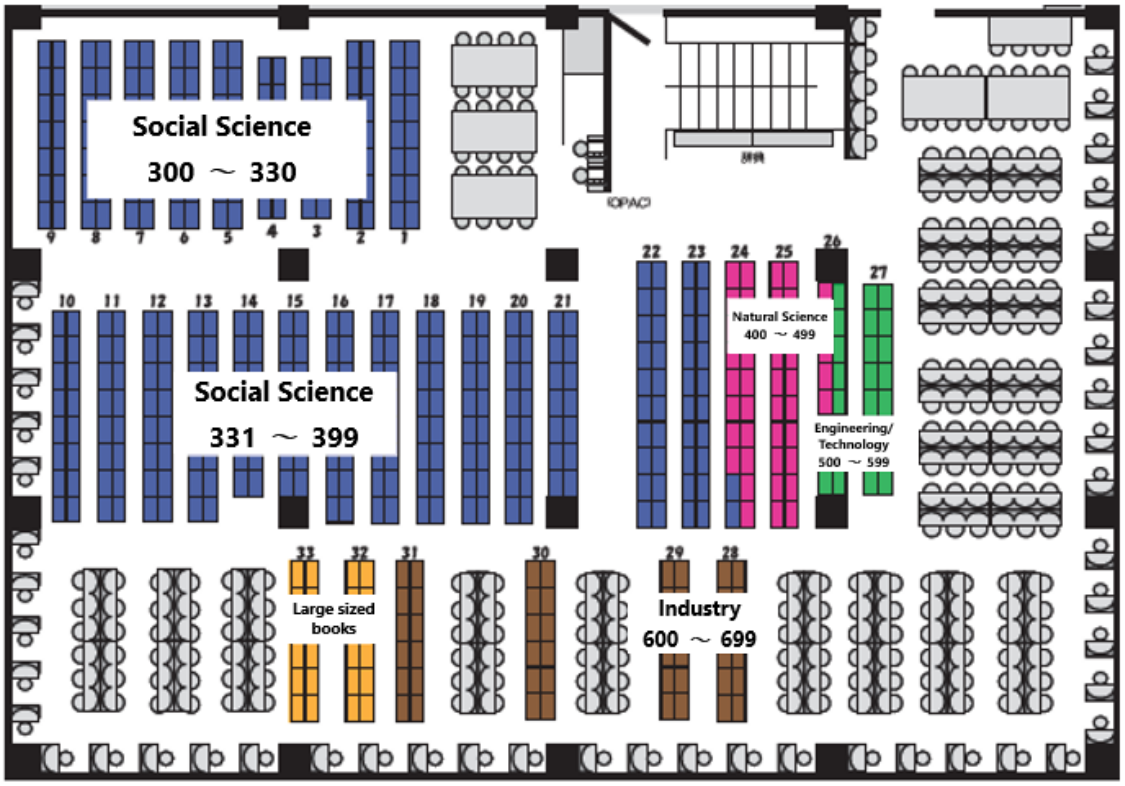

Figure 4.5: Bookshelves on the second floor of Nanzan University Library

Core^{TM} i7-8550U CPU and 16GB of RAM.

It took about 54 seconds to obtain the optimal solution whose value is about $3,811$. The optimal solution relocates 173 book groups, that is, 95.6% percent of the book groups have to be rearranged. The problem for $n = 9$ was infeasible.

Therefore, we relaxed the original integer variables $l_i^{(j)}$, w_i^j, t'_i, and t''_i as a continuous variable. It took about 34 seconds to obtain the optimal solution whose value is about $1,311$. The optimal solution relocates 37 book groups (i.e., 20% percent of the book groups have to be rearranged).

Additionally, we can solve the problem for $n = 11$ with the continuous variables. It took about 40 seconds to obtain the optimal solution whose value is about $3,972$. The optimal solution relocates 176 book groups, that is, 97% percent of the book groups have to be rearranged. If we accept that a section can start at the middle of a shelf, we can maintain the vacant sections for 8 years with fewer moving books. Also, we can maintain the vacant sections for a longer time. This optimal solution is considered to be feasible in practice and more acceptable.

Chapter 5

Conclusion

In this research, we constructed the models for solving the problems at Nanzan University using OR to reduce the burden on the staff. We formulate the TA shift scheduling problem as a 0-1 integer programming problem, the classroom assignment problem as a 0-1 integer and linear programming problems, and the books relocation problem as a mixed-integer programming problem using the actual data of Nanzan University. Then, using the models, we implemented interactive assignment support systems: the shift scheduling support system for TAs and the classroom assignment support system.

The TA shift scheduling support system and the classroom assignment support system can flexibly respond to different conditions and provide an assignment quickly and reduce the workload of the staff. For the TA shift scheduling problem, we succeeded in reducing the working time to about 1/25 and improving the schedule preparation time by using the shift scheduling support system for TAs, compared to manual work. For the classroom assignment problem, even the staff who is not accustomed to assigning classes to classrooms can quickly obtain an initial schedule that satisfies all the conditions and requirements and performs assignments without difficulty by using the classroom assignment support system. We also found from the computational experiments the necessity to improve the system so that the staff can freely change the cost of the edges in the future.

For the book relocation problem, we obtained the optimal relocation of the books of a part of the second floor of NUL by using our constructed model. The results are acceptable for practical use, and the CPU time to solve the problem is within the practical range. We will consider the problem of the whole library by dividing the library into several parts and check the effectiveness of the model and by the feedback from the library staff, we will improve the model. In the future, we will implement a book relocation support system.

We will continue to improve the systems, and we will receive feedback from the staff concerning improving the systems' effectiveness and user-friendliness. As we mentioned in Chapter 1, many universities have similar problems with assignments such as those introduced in this research. Hence, we can contribute to improving the operational efficiency in such universities if they introduce the systems. Our implemented systems consist of general conditions applied to almost every university and conditions specific to Nanzan University. Therefore, the system can be flexibly introduced to other universities only by modifying functions to acquire data from the on-campus database or customizing the system to exclude the conditions specific to Nanzan University.

Acknowledgment

First, I would like to express sincere appreciation to Professor Atsuo Suzuki whose comments and suggestions were of inestimable value for my study. I am also deeply grateful to Professors Masao Fukushima, Mihiro Sasaki, and Shungo Koichi for their support and guidance. Without their support, I would not be able to complete this dissertation. I am also grateful to Professors Hidetoshi Miura and Koichi Nakade who gave me constructive comments and warm encouragement.

Second, I would like to thank Mr. Norihiko Omiya and Mr. Hitoshi Ikeuchi for supporting my research activities. And, I am grateful to Mr. Kenji Kondo, Mr. Takeshi Kanda, Ms. Marie Suzuki, Ms. Kumiko Kondo, and staff of Nanzan University for permission to use various data for this dissertation. I also highly appreciate the feedback offered by them.

Finally, I express my gratitude to my family for supporting me with warm encouragement.

Bibliography

[1] Adrianto, D.: "Comparison using Particle Swarm Optimization and Genetic Algorithm for Timetable Scheduling," *Journal of Computer Science*, Vol. 10, No. 2, pp. 341–346 (2014)

[2] Babaei, H., Karimpour, J. and Hadidi, A.: "A Survey of Approaches for University Course Timetabling Problem," *Journal of Computers & Industrial Engineering*, Vol. 86, pp. 43–59 (2015)

[3] Camilo, T., Jairo R.M., Carlos, L.Q., Angélica, S., and Mónica, C.: "University Course Scheduling Classroom Assignment," *Journal of Ingeniería y Universidad*, Vol. 18, No. 1, pp. 59–75 (2014)

[4] Cooper, Michael D. and Wolthausen, J.: "Misplacement of Books on Library Shelves: A Mathematical Model," *Journal of Library Quarterly*, Vol. 47, No. 1, pp. 43–57 (1977)

[5] Daskalaki, S., Birbas, T., and Housos, E.: "An Integer Programming Formulation for a Case Study in University Timetabling," *European Journal of Operational Research*, Vol. 153, pp. 117–135 (2004)

[6] Furusawa, M. and Sasaki, M.: "大学におけるクォーター制に対応した教室割り当て問題 – 南山大学を例として –," *Academia. Sciences and Engineering : Journal of Nanzan Academic Society*, Vol. 19, pp. 39–47 (2019) (in Japanese)

[7] Horio, M.: "Application to a University Course Timetabling Problem by a General Project Scheduler," *Journal of Liberal Arts, Nagoya University of Arts and Sciences*, No. 4, pp. 61–74 (2008) (in Japanese)

[8] Horio, M.: "Development of a Practical System to Generate University Timetables," *Journal of Japan Industrial Management Association*, Vol. 62, No. 1, pp. 1–11 (2011) (in Japanese)

[9] Ikegami, A., Niwa, A., and Ohkura, M.: "我が国におけるナース・スケジューリング問題," *Communications of the Operations Research Society of Japan*, Vol. 41, No. 8, pp. 436–442 (1996) (in Japanese)

[10] Ito, M., Ohnishi, A., Suzuki, A., Imamura, A., and Ito, T.: "The Resident Scheduling Problem – A Case Study at Aichi Medical University Hospital –," *Journal of Japan Industrial Management Association*, Vol. 68, No. 4E, pp. 259–272 (2018)

[11] Ito, M., Suzuki, A., and Fujiwara, Y.: "A Prototype Operating Room Scheduling System – A Case Study at Aichi Medical University Hospital –," *Journal of Japan Industrial Management Association*, Vol. 67, No. 2E, pp. 202–214 (2016)

[12] Michael, W. C. and Craig, A. T.: "When Is the Classroom Assignment Problem Hard?," *Institute for Operations Research and the Management Sciences*, Vol. 40, No. 1 (suppl.), pp. S28–S39 (1992)

[13] Ministry of Education, Culture, Sports, Science and Technology-Japan, https://www.mext.go.jp/b_menu/shingi/gijyutu/gijyutu4/toushin/attach/1337935.htm, Japanese web site (accessed 2021-01-10)

[14] Nagumo, S. and Terashima, K.: “「図書館引っ越しらくらくキット」を用いた資料再配置の実践報告,” *Annual Report of the Hitotsubashi University Library Research Development Office*, No. 7, pp. 7-04-1–7-04-8 (2019) (in Japanese)

[15] Nakamoto, M.: “早稲田大学図書館 – 十字路に立つ大学図書館 –,” *Journal of Information Processing and Management*, Vol. 45, No.6, pp. 428–430 (2002) (in Japanese)

[16] Nampo, K.: “明治大学図書館が抱える課題,” *Bulletin of Meiji University Library*, No. 24, pp. I–III (2002) (in Japanese)

[17] Ogawa, T. and Hotta, K.: “Providing Support in Solving Course Timetabling Problem at Bunkyo University,” *Shonan Forum : Journal of Shonan Research Institute Bunkyo University*, No. 20, pp. 21–46 (2016) (in Japanese)

[18] Ohnishi, A. and Suzuki, A.: “TA 勤務スケジューリング支援システム – 南山大学 S 棟 学生サポート窓口の事例 –,” *Proceedings of the Forty-six Annual Meeting at Chubu chapter of the Operations Research of Japan*, pp. 2–5 (2018) (in Japanese)

[19] Ohnishi, A. and Suzuki, A.: “A Prototype Subject-classroom Assignment Support System – A Case Study at Nanzan University –,” *Proceedings of the International Symposium on Scheduling*, pp. 172–177 (2019)

[20] Ohnishi, A., Suzuki, A., and Fujiwara, Y.: “麻酔科医の当番・当直勤務スケジューリング支援システムの試作,” *Proceedings of the Communications of the Operations Research Society of Japan 2015 Spring Research Presentation*,” pp. 88–89 (2015)

[21] Shiozawa, T. and Okabe, K.: “Development of the Complicated University Class Time Table Making Program,” *IPSJ Transactions on Computers and Education*, No. 101, pp. 1–4 (2001) (in Japanese)

[22] Smalley, Hannah K. and Keskinocak, P.: “Automated Medical Resident Rotation and Shift Scheduling to Ensure Quality Resident Education and Patient Care,” *Health Care Management Science*, Vol. 30, pp. 1–23 (2014)

[23] Smalley, Hannah K. and Keskinocak, P.: “Physician Scheduling for Continuity: An Application in Pediatric Intensive Care,” *Journal of Interfaces*, Vol. 45, No. 2, pp. 133–148 (2015)

[24] Suzuki, A. and Fujiwara, Y.: “手術室のスケジューリング支援システムについて,” *Communications of the Operations Research Society of Japan*, Vo. 58, No. 9, pp. 515–523 (2013)

[25] Suzuki, A., Sawaki, K., and Hasegawa, T.: “An OR/MS approach to managing Nanzan Gakuen (Nanzan Educational Complex): From the strategic to the daily operational level,” *Journal of Interfaces*, Vol .36, pp. 43–54 (2006)

[26] Tokunaga, T., Tanaka, Y., Kobayashi, T., Kutsumizu, Y., and Ikegami, A.: “Development of a Staff Scheduling Model for Part-time Employees and a Support System,” *Journal of Information Processing Society of Japan. Transactions on mathematical modeling and its applications*, Vol. 8, No. 2, pp. 57–65 (2015) (in Japanese)

[27] Topaloglu, S. and Ozkarahan, I.: “A Constraint Programming-based Solution Approach for Medical Resident Scheduling Problems,” *Computers & Operations Research*, Vol. 38, No. 1, pp. 246-255 (2011)

[28] Yamamoto, K. and Suzuki, A.: “南山大学における入試監督者自動割当システムの作成,” *Communications of the Operations Research Society of Japan*, Vol. 54, No. 6, pp. 335–341 (2009) (in Japanese)

[29] Yashiro, T.: "Bunkyo University Shonan Library – Its Role in the Activities of Education and Research –," *Shonan Forum : Journal of Shonan Research Institute Bunkyo University*, No. 12, pp. 69–79 (2008) (in Japanese)

[30] *AirSHIFT*, Recruit Co., Ltd.,
https://airregi.jp/shift/, Japanese web site (accessed 2021-06-09).

[31] *Hataluck*, Knowledge Merchants Works Inc.,
https://hataluck.jp/lp/shift/, Japanese web site (accessed 2021-06-09).

[32] *Shiftmation*, Axicerse, Inc.,
https://www.shiftmation.com/, Japanese web site (accessed 2021-06-09).

[33] *Timetabling of University*, Chuo Computer Service. Inc.,
https://www.ccs1981.jp/daigakujikanwari/timetable/,
Japanese web site (accessed 2020-08-21).

[34] *Timetabling System for the University*, Global Systems Co., Ltd.,
https://www.gss-sys.co.jp/service/university/jikanwari/,
Japanese web site (accessed 2020-08-21).

[1] "Microsoft", "Windows", "Excel", "Accessl", and "VBA" are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries

[2] "Intel Core" is trademark of Intel Corporation or its subsidiaries in the United States and other countries.