Solving Social Problems Using Integer Programming

石倉, 弘貴

Kyushu University

Doctoral Thesis

Solving Social Problems Using Integer Programming

Author:
Hiroki Ishikura

Supervisor:
Prof. Katsuki Fujisawa

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Mathematics

in the

Graduate School of Mathematics

January 9, 2024

KYUSHU UNIVERSITY

Abstract

Graduate School of Mathematics

Doctor of Mathematics

Solving Social Problems
Using Integer Programming

by Hiroki Ishikura

Recent advances in computing, cameras, and sensors have significantly improved their performance and expanded the range of applications for mathematical optimization and deep learning. This technological evolution is also reflected in the industrial sector, where the acquisition and utilization of diverse data is becoming more important. In addition, more and more applications and services utilizing these technologies are emerging. This paper deals with two studies aimed at solving real-world problems using mathematical optimization. The first is optimizing the wiring design for optical fibers with a film-type system. When optical fibers are routed on film, various conditions impose constraints on the fibers to be routed. It is challenging to find a way to minimize the number of film fibers used while keeping these restrictions. Therefore, we formulated this problem as a mixed integer programming problem to find a wiring method that minimizes the number of films used. Numerical experiments confirmed that the proposed method uses fewer films than the rule-based wiring method. The second is mobility optimization in an automated warehouse. Automated warehouses contain multiple transport machines. In the automated warehouse used by Rohto Pharmaceutical Co., the subject of this study, multiple transporters are used to transport goods. It is not easy to use these transporters efficiently because each transporter's behavior affects the state of the others. Therefore, we modeled the movement of all transporters and cargo using a time expansion network and devised a problem to optimize the behavior of the transporters as an integer programming problem. Using this problem, we confirmed that the total time required to transport the cargo is reduced compared to the rule-based transport method.

Acknowledgement

I would like to express my deepest gratitude to Professor Katsuki Fujisawa for his heartfelt assistance and guidance in my doctoral dissertation. Professor Fujisawa's dedicated guidance and insight allowed me to delve deeper into my research and achieve results. His advice and encouragement were essential to my development as a researcher. Thank you very much. Next, I would like to extend my deepest gratitude to my seniors, Doctors Hata and Tateiwa, as well as other laboratory members. Thanks to all of you, I was blessed with an environment that facilitated my growth through valuable discussions and stimulated the creation of new ideas. I would also like to express my appreciation to Ms. Ikebe, Ms. Sakaii, and Ms. Iwashita for their generous support in administrative work related to my research. Thanks to their assistance, I could focus on my research. Finally, I want to thank my family and my girlfriend, Kanon, for their unwavering support. Thanks to them, I could complete my doctoral dissertation without giving up. Thank you so much.

Contents

Chapter 1

Introduction

In recent years, the performance of computers, cameras, and other sensors has improved significantly, and the scope of applications for mathematical optimization and deep learning has expanded. In response to this trend, the industrial world is also moving toward acquiring and utilizing various types of data. An example is the inspection of defective products in the manufacturing industry. Until now, defective products have been inspected manually. However, the inspection of defective products was very costly because of the high workload placed on workers, who were often replaced after only a short time cycle. Today, the company uses cameras to capture images of parts and products and automatically determines defective products using machine learning and deep learning. Mathematical optimization is also being used to formulate delivery plans. For example, in the task of LP gas delivery, mathematical optimization is used to select the optimal delivery destination based on the prediction of each customer's remaining gas volume and to formulate a route to visit the selected delivery destination efficiently. Thanks to a series of tasks, it is now possible to automatically select appropriate delivery destinations based on the amount of remaining gas and even to automatically formulate routes, which previously had to be thought up by delivery staff. The data acquired using such diverse IoT devices can be used to improve operational efficiency and reduce the workload of workers.

A social system that maximizes the use of these technologies is called a Cyber Physical System (CPS). In a society where CPSs are realized, we can expect to build more efficient social systems and improve the quality of our lives by modeling and databasing real-world phenomena, analyzing them in cyberspace, acquiring new knowledge, and feeding them back into the real world. The smart factory is the realization of this system on the platform of a factory. The smart factory acquires IoT devices such as cameras and beacons, machines used in the manufacturing process, and data created by human factors such as production planning. It analyzes digitized production activities to realize production planning to increase throughput and improve worker safety in the factory. Numerical optimization is also a necessary technology for realizing smart factories. In order to determine the optimal production schedule based on the acquired data and to optimize the flow of people, products, and other items in the factory, these problems are formulated as optimization problems.

Thus, the demand from the real world to solve social problems using mathematical optimization and

deep learning is increasing yearly. This paper focuses on the optical fiber routing optimization problem using mixed integer programming and mobility optimization in automated warehouses using integer programming. In Chapter 2, we aim to automatically determine the wiring method of optical fibers, which the workers themselves have considered, and to minimize the number of film components used in the wiring process. In Chapter 3, we consider optimizing the mobility of loads in an automated warehouse installed in a factory. Research on automated warehouses is one of the project's research areas for the realization of smart factories.

Chapter 2

Optimization of film-type optical fiber wiring design using mixed-integer programming problem

2.1 Background

Optical fiber technology is among the technologies that support modern communication. It is a communication technology that passes light through a fiber consisting of two overlapping materials with different refractive indices. Due to the nature of light as a medium, optical fiber technology offers faster communication speeds and minor transmission loss than conventional coaxial cables. Therefore, it is used in various communications, such as in submarine cables for international communication and in enterprises' high-speed communication lines. This is possible because optical fiber technology enables long-distance communications. Optical fiber is also used for inter-node communication in server design because it is necessary to increase the communication speed between nodes when communicating a large amount of data or performing large-scale calculations. Additionally, the demand for optical fiber has been increasing in recent years as the amount of data communication continues to increase with the spread of smartphones, cloud computing, and the Internet of Things.

There are two optical fiber wiring methods: spatial and film. In spatial wiring, only the top and bottom ends of an optical fiber are fixed, and wiring is performed without fixing the interconnections between them. In this method, there is no need to design the wiring section between the upper and lower ends, and wiring can be done manually. However, a drawback is the risk of transmission loss because optical fibers' bending shape cannot be fixed. On the other hand, in film wiring, the optical fiber is fixed to a rectangular film using adhesive during the wiring process. In this method, the fibers' bending shape can be maintained, and the risk of transmission loss is minimal. It also has the advantage of high flexibility due to its sheet form. However, it is necessary to satisfy several design requirements for fixing optical fibers, such as bending geometry and equipment restrictions. Therefore, there is a limit to the number of optical fibers that can be wired per layer of film, and wiring design must be done in advance.

Furthermore, the number of films required for wiring must be as small as possible to reduce the cost of films and other materials. In addition, the time required for wiring should not be excessively long.

We previously collaborated with Sumitomo Electric Industries, Ltd. (here-after referred to as Sumitomo Electric) on work on film optical fiber interconnects, in which optical fibers are routed from the film's top to the bottom (one-direction layer-based optical fiber arrangement, 1d-LOFA). This study is the latest version of that work. This study proposes an exact solution method using integer programming problems. Integer programming problems have been used to solve various real-world problems, such as routing [3, 13, 30], scheduling [19, 21, 22], and portfolio problems [4, 23]. In this study, we exploited the expressive power of integer programming problems to formulate 1d-LOFA. We also devise a heuristic method to efficiently solve problems of a scale that exact solution methods cannot handle.

Related studies include wiring problems in integrated circuits and piping design problems in shipbuilding. The differences between the problems addressed in this study and the problems mentioned above are the dimension of the wiring space, the presence or absence of obstacles in the wiring space, and whether lines are allowed to cross each other. For wiring problems in integrated circuits, there are approaches such as the maze wiring method and genetic algorithms [18]. Regarding the place-and-route problem, which includes the placement of circuit elements, a component of the wiring problem, methods based on neural networks and simulated annealing have been proposed [17, 8]. For the piping design problem in shipbuilding, approaches have been proposed that divide the problem into multiple mixed-integer programming problems [12] and those that apply genetic algorithms [26]. Angilella et al. also solved the network design problem using integer linear programming [1].

The remainder of this paper is organized as follows. Section 2.2 describes the design requirements for film-based interconnection design and defines the 1d-LOFA. Section 2.3 defines the present routing problem and proposes an exact solution method and a heuristic method, including a formulation as a mixed-integer programming problem. In Sect. 2.4, we report experiments using sample data prepared based on the data Sumitomo Electric provided; we discuss the results comparatively.

2.2 Problem description

Section 2.2.1 briefly introduces optical fibers and explains the points to be considered when routing them. Section 2.2.2 introduces the optical fiber routing problem.

2.2.1 Design requirements

This section discusses the design requirements for film optical fibers. An optical fiber consists of a core with a high refractive index, a cladding with a low refractive index, and a coating covering them (Fig. 2.1). The core and cladding are two-layered glass structures, and light incident on the core propagates in the optical fiber by total reflection at the boundary surface with the cladding. When an optical fiber is bent

steeply, the angle of light incidence on the boundary surface becomes significant at the bent part, resulting in a partial loss of light instead of total reflection [9, 16].

In film wiring, optical fibers are wired while being fixed to a rectangular film with adhesive. In this method, each fiber's path and distributing the fibers to the film are decided according to maintaining the fibers' bending shape. This method can minimize the risk of transmission loss. To reduce transmission loss, it is first necessary to avoid the sudden bending of optical fiber cables during routing. The following restrictions are set to satisfy this condition.

(a) The path curvature radius must be greater than the specified value. The minimum value is denoted by R and is called the minimum radius of the curvature. It is represented by a positive real number $R > 0$.

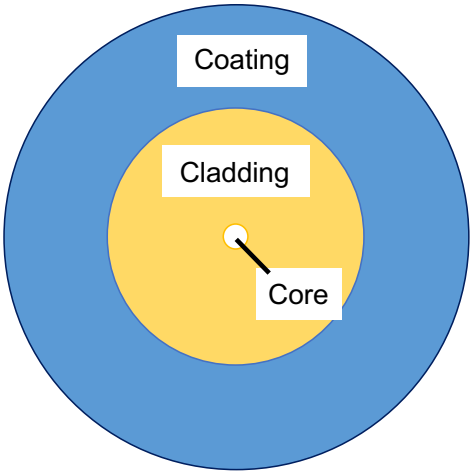

FIGURE 2.1: Structure of an optical fiber

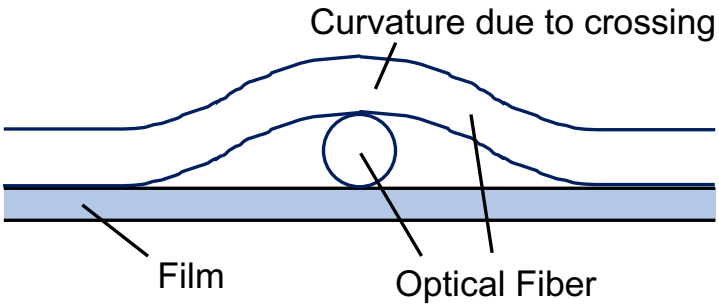

FIGURE 2.2: Intersection of two optical fibers

Optical fiber intersections must also be considered. For example, in a film arrangement, fibers overlap other fibers at intersections (Fig. 2.2). Bending near these intersections can cause transmission loss or breakage. Considering these factors, the following requirements are set for fiber intersections.

(b) At intersections, optical fibers are on straight lines and do not intersect during bending.

(c) No more than three optical fibers shall intersect at a single point.

(d) At intersections, fibers must be orthogonal.

These conditions can prevent fiber breakage. These conditions can also reduce transmission losses at the crossing. When fibers overlap, fibers bend, and transmission loss occurs. Therefore, by limiting fiber crossings to two fibers and making the crossings orthogonal, overlapping fibers can be minimized, and transmission loss can be expected to be suppressed.

Additionally, the following requirements for manufacturing facilities are to be considered.

(e) Two parallel wired fibers must be at least a certain distance apart. Let $d > 0$ denote the minimum distance between two parallel fibers. d is called the minimum adjacency space.

In this study, (a)–(e) are referred to as design requirements.

2.2.2 Problem definition

Let F be the set of optical fibers to be routed and L be the set of films for routing. Define W and H as the width and height of the film, respectively. The point at the upper left of the film is the origin; the x-axis is taken in the width direction, and the y-axis is in the height direction. Each fiber $f \in F$ has pre-defined endpoints $\left(x_0^f, y_0^f\right)$ and $\left(x_1^f, y_1^f\right)$ on the film edge. Fibers are routed from $\left(x_0^f, y_0^f\right)$ to $\left(x_1^f, y_1^f\right)$. Depending on the position of the endpoints, the fiber may need to be bent. Additionally, there are restrictions on how fibers are routed to each other depending on the endpoints' positional relationship. Therefore, the number of films required to wire all fibers may vary greatly depending on the fiber wiring method. The LOFA is the optimization problem to minimize the number of films required for fiber routing.

We define the LOFA handled in this study as the one-directional LOFA (1d-LOFA) based on the direction of the optical fiber to be inserted. The 1d-LOFA is routed only from the top edge of the film to the bottom edge. If necessary, the fiber must be bent during routing to comply with design requirements. In order to comply with the design requirements, some fibers cannot be routed on the same film, and some fibers introduce bending restrictions when routing. Therefore, the fiber routing method significantly impacts the number of films required. Therefore, we devised an algorithm to identify the wiring method using a mixed-integer programming problem for 1d-LOFA.

2.3 Two methods to solve 1d-LOFA

This section describes the methodology for solving 1d-LOFA as a mixed integer programming problem. First, Sect. 2.3.1 introduces 1d-LOFA in detail. Next, Sect. 2.3.2 describes the formulation of the 1d-LOFA. The design requirements are formulated as constraints, and the number of layers is minimized using an objective function. Finally, Sect. 2.3.3 presents a heuristic algorithm for solving 1d-LOFA fast. When solving the formulated problem, the computation may be time-consuming. Therefore, we devised a heuristic algorithm to find the fiber optic wiring fast.

2.3.1 Problem definition of 1d-LOFA

The 1d-LOFA is a wiring optimization problem for wiring fibers only from the top edge of the film to its bottom edge. In this study, to simplify the wiring problem, we restrict the fiber wiring method with the following constraints. These conditions are the primary wiring methods for the 1d-LOFA, as discussed and determined with Sumitomo Electric. Adhering to these conditions will make it easy to find wiring

that meets the design requirements described in Sect. 2.2.1. These restrictions also narrow the solution search range but make the optimization problem easier to handle.

(A) Each fiber is routed parallel to the x and y-axis except for bent portions.

(B) Each fiber can be bent up to two times.

Restriction (A), in other words, prohibits wiring that goes straight in a diagonal direction to the top or bottom edge of the film. This restriction means that when fibers cross, they always are orthogonal except for the bent portion. Therefore, if design requirement (b) is satisfied, design requirement (d) is naturally satisfied. Furthermore, since the fibers are routed along the x, and y-axes and intersections only occur at the straight sections of the fibers, three or more fibers never intersect at a single point unless they share x and y coordinates. In other words, design requirement (c) can also be satisfied.

Optical fiber f is routed from the pre-defined top end to the bottom end. If the x coordinates of the top and bottom ends are different, the fiber must be bent to observe the restriction (A). However, when fibers are bent, transmission loss will likely occur at the bent portion. In other words, bending a fiber many times will adversely affect the transmission of information. The restriction (B) can reduce overly complicated paths and unnecessary intersections caused by fiber bending.

Due to these restrictions, three bending methods depend on the position of the fiber endpoints (Fig. 2.3). Let the x coordinate of the top end of the fiber be x_0^f and the x coordinate of the bottom end be x_1^f. The fiber bends at $|x_1^f - x_0^f| > 2R$, $0 < |x_1^f - x_0^f| \leq 2R$, and $|x_1^f - x_0^f| = 0$. When $|x_1^f - x_0^f| > 2R$, the fiber consists of two sections parallel to the y-axis, two bending sections, and one section parallel to the x-axis. When $0 < |x_1^f - x_0^f| \leq 2R$, the fiber consists of two sections parallel to the y-axis and two bending sections. When $|x_1^f - x_0^f| = 0$, the fiber consists of one section parallel to the x axis. The red and blue lines in Fig. 2.3 represent the bending sections.

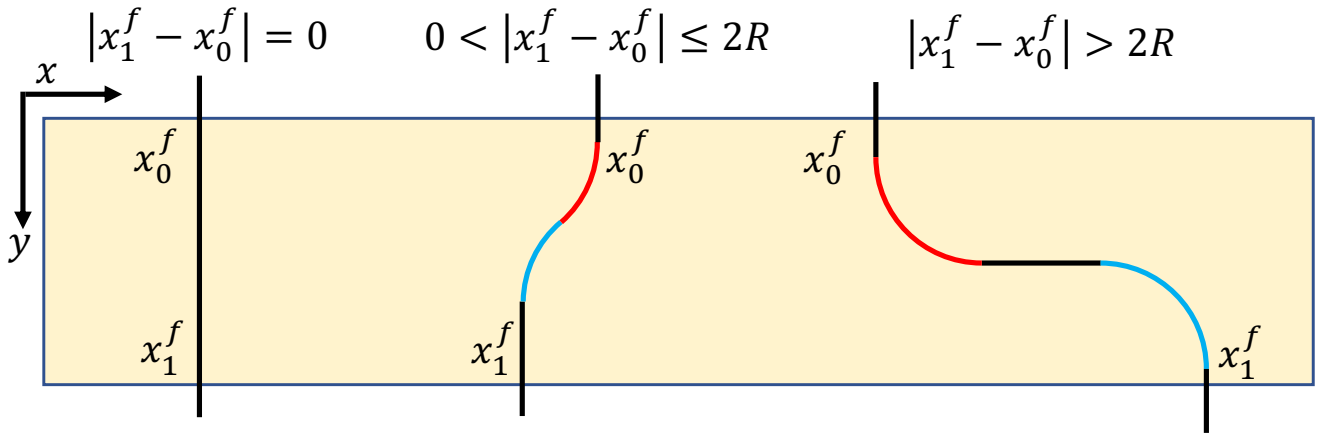

FIGURE 2.3: Examples of optical fiber arrangements

Under these conditions, fiber f is routed as follows (Fig. 2.4). First, f is extended straight from the top end of the film. Next, bend f to an appropriate shape and extend f straight to the bottom end of the film.

When bending f, bend it to form an arc with a minimum bending radius of R. The central angle of the fan-shaped radius R with this arc is called the bending angle and is defined as $\theta_f \in [0, \frac{\pi}{2}]$. Furthermore, to consider the wiring conditions, y_f is defined as the bending height to match the bending portion of the fiber. In the case of $|x_1^f - x_0^f| > 2R$, y_f is the y coordinate of the section parallel to the x-axis. On the other hand, in the case of $0 < |x_1^f - x_0^f| \leq 2R$, y_f is the y coordinate of the point where the two bend sections are connected. In the case of $|x_1^f - x_0^f| = 0$, the value of y_f is treated as 0 because there is no bending part in the fiber.

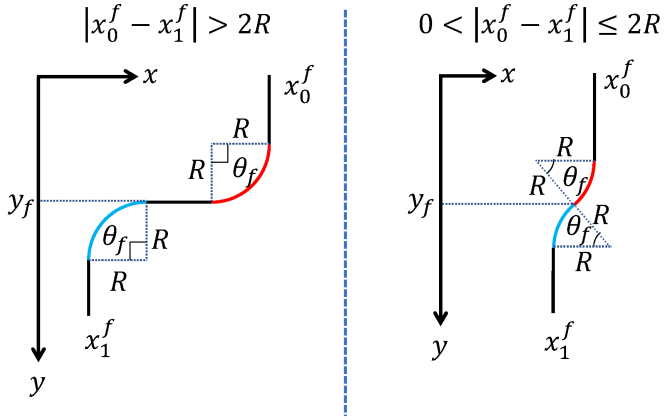

FIGURE 2.4: Examples of fiber curves

2.3.2 Exact solution method

For the 1d-LOFA restricted by (A) and (B), we give the formulation as a mixed-integer programming problem. When formulating the 1d-LOFA, it is necessary to express the design requirements (b) and (e) as constraint expressions. Since these design requirements refer to pairs of optical fibers, their form depends on their respective positions in the fiber pair. Specifically, the more or lower height that can be bent in each fiber pair is determined by the higher or lower x coordinates of each start and end point.

This section describes the constraints set to satisfy two types of design requirements. Constraints (b) and (e) have eight and four constraint conditions, respectively. In this paper, each condition is denoted by $B = \{b_i | i \in [0, 7]\}, E = \{e_i | i \in [0, 3]\}$, and the corresponding set of fiber pairs is denoted by $C_{b_i}(i \in [0, 7]), C_{e_i}(i \in [0, 3])$. The applicable conditions for a given pair of fibers are determined by the relationship between the fibers' starting and ending points.

The condition for design requirement (b) is that the two fibers f_1, f_2 should be routed to not overlap at their bend positions. Possible conditions for this are as follows.

(b_0) cannot be wired in the same layer

(b_1) $y_{f_2} \leq y_{f_1} - R \sin \theta_{f_2}$

(b_2) $y_{f_2} \geq y_{f_1} + R \sin \theta_{f_2}$

(b_3) $y_{f_2} \geq y_{f_1}$

(b_4) $y_{f_2} \leq y_{f_1} - R\sin\theta_{f_1}$

(b_5) $y_{f_2} \geq y_{f_1} + R\sin\theta_{f_1}$

(b_6) $y_{f_2} \leq y_{f_1}$

(b_7) can be freely wired

The following explains the constraints in (b0) and (b1) based on a concrete example. Here, we use two fibers, f1 and f2, which are $|x_0^f - x_1^f| > 2R$. Due to the symmetry of the wiring method, we assume $x_0^{f_1} \leq x_1^{f_1}$ to exclude patterns with the same bending method.

The b_0 indicates that two different fibers cannot be routed to the same film; Fig. 2.5 shows an example. We consider the case where the top end of fiber f_2 is $x_0^{f_2} > x_1^{f_1} + g_{f_1}$, and the bottom is $x_1^{f_1} - g_{f_1} < x_1^{f_2} < x_1^{f_1}$, where $g_{f_1} = R(1-\cos\theta_{f_1})$ represents the bent section's length on the x-axis. No matter where these fibers f_1, f_2 are bent in height, the other fiber always overlaps one fiber's bent portion. Therefore, these two fibers cannot be routed to the same film.

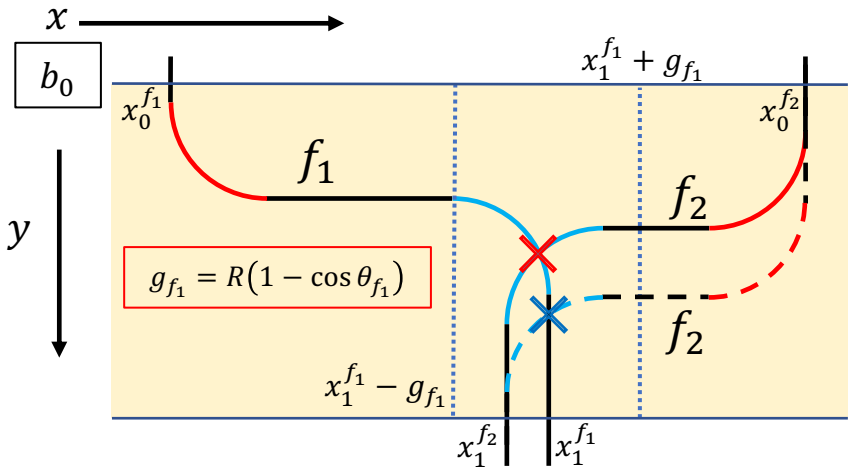

FIGURE 2.5: Example of optical fibers that cannot be routed to the same film

An example of the routing of b_1 to b_6 is shown in Fig. 2.6, where the coordinates of the two fibers' endpoints constrain the location of the fiber bend. For example, if the top end of fiber f_2 is at $x_0^{f_1} < x_0^{f_2} < x_0^{f_1} + g_{f_1}$ and the bottom end is at $x_0^{f_1} + g_{f_1} < x_1^{f_2} < x_1^{f_1} - g_{f_1}$, fiber f_2 must be bent at a position higher than $R\sin\theta_{f_2}$ for the bending height of f_1. If the bend is lower than that, the red bend in f_1 and f_2 or the blue bend in f_2 intersects with the other fiber. Therefore, if these fibers are routed to the same film, the b_1 constraint must be satisfied. We consider b_2 through b_6 similarly, with each constraint occurring when routing to the same film. The relationships between all fiber pairs and constraints are described in Appendix A.1.2.

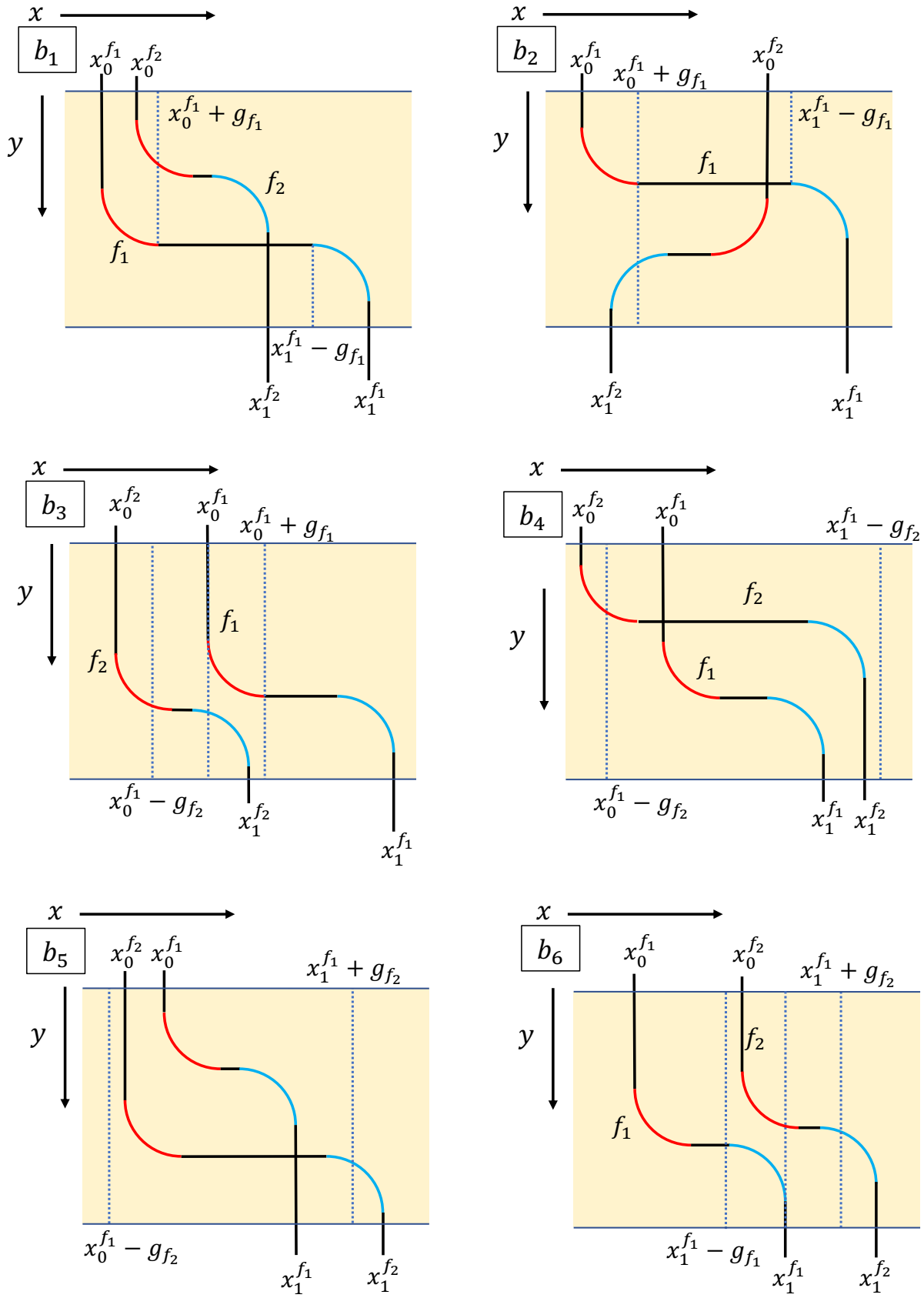

FIGURE 2.6: Examples of wiring from b_1 to b_6. g_f is defined as $R(1 - \cos\theta_f)$, where R is the minimum radius of the curvature and θ_f is the bending angle of fiber f

The condition in design requirement (e) means that when routing two fibers f_1, f_2, a space must be opened above a specific spacing. The conditions considered for this are as follows.

(e_0) cannot be wired in the same layer

(e_1) $y_{f_2} \geq y_{f_1} + d \times \alpha(f_2)$

(e_2) $y_{f_2} \leq y_{f_1} - d \times \alpha(f_2)$

(e_3) can be freely wired

The function α within constraints (e_1) and (e_2) is defined as follows;

$$\alpha(f) = \begin{cases} 1 & \text{Optical fiber } f \in F \text{ is routed with bending} \\ 0 & \text{otherwise} \end{cases}.$$

When bending fibers, these conditions allow the spacing between fibers to be greater than or equal to the minimum adjacent spacing d. The relationship between all fiber pairs and constraints is described in Appendix A.1.2. Additionally, for all combinations of fibers to be routed on the same film, the corresponding constraints of (b_0) – (b_7) and (e_0) – (e_3) must be satisfied.

We formulated the 1d-LOFA as a mixed-integer programming problem based on the above constraints. The following summarizes the sets, constants, variables, and a function used in the formulation.

1. Sets
    - F : Optical fiber set to be routed
    - $L = \{0, \ldots, n-1\}$: The set of films to be wired, where n is the maximum number of films available
    - $B = \{b_0, \ldots, b_7\}$: The set that manages the constraints of design requirement (b)
    - $E = \{e_0, \ldots, e_3\}$: The set that manages the constraints of design requirement (e)
    - $C_i \subset F \times F (i \in B \cup E)$: The set of fiber pairs corresponding to the constraint i
    - $I^B_{f_1,f_2} \subset B$: Constraints in design requirement (b) that arise when routing two fibers $f_1, f_2 \in F$ on the same film
    - $I^E_{f_1,f_2} \subset E$: Constraints in design requirement (e) that arise when routing two fibers $f_1, f_2 \in F$ on the same film
2. Constants
    - $R \in \mathbb{R}^+_0$: The minimum radius of the curvature

- $H \in \mathbb{R}^+$: The film height
- $d \in \mathbb{R}^+$: The minimum adjacency space
- $\theta_f \in [0, \frac{\pi}{2}]$: The bending angle of fiber f

3. Variables

- $x_{l,f} = \begin{cases} 1 & \text{Optical fiber } f \in F \text{ is wired to film } l \in L \\ 0 & \text{otherwise} \end{cases}$
- $u_l = \begin{cases} 1 & \text{Film } l \in L \text{ is used} \\ 0 & \text{otherwise} \end{cases}$
- $z^i_{f_1,f_2} = \begin{cases} 1 & \text{Fiber pairs } f_1, f_2 \in F \text{ satisfy design requirement } i \in B \cup E \\ 0 & \text{otherwise} \end{cases}$
- $y_f \in [0, H]$: A continuous variable that determines the bending height of fiber $f \in F$

4. Function

- $\alpha(f) = \begin{cases} 1 & \text{Fiber } f \text{ needs to be bent.} \\ 0 & \text{otherwise} \end{cases}$

Problem 1. *1d-LOFA*

$$\underset{u,x,y,z}{\textit{minimize}} \quad \sum_{l\in L} u_l \tag{2.1}$$

$$\textit{subject to} \quad u_l \le u_{l-1} \qquad {}^{\forall} l \in L \setminus \{0\} \tag{2.2}$$

$$\sum_{f\in F} x_{l,f} \le |F| \times u_l \qquad {}^{\forall} l \in L \tag{2.3}$$

$$\sum_{l\in L} x_{l,f} = 1 \qquad {}^{\forall} f \in F \tag{2.4}$$

$$x_{l,f_1} + x_{l,f_2} \le 1 \qquad {}^{\forall}(f_1,f_2) \in C_{b_0} \cup C_{e_0},\ {}^{\forall} l \in L \tag{2.5}$$

$$y_{f_2} - y_{f_1} + R\sin\theta_{f_2} \le \left(1 - z^{b_1}_{f_1,f_2}\right) \times M \qquad {}^{\forall}(f_1,f_2) \in C_{b_1} \tag{2.6}$$

$$y_{f_1} - y_{f_2} + R\sin\theta_{f_2} \le \left(1 - z^{b_2}_{f_1,f_2}\right) \times M \qquad {}^{\forall}(f_1,f_2) \in C_{b_2} \tag{2.7}$$

$$y_{f_1} - y_{f_2} \le \left(1 - z^{b_3}_{f_1,f_2}\right) \times M \qquad {}^{\forall}(f_1,f_2) \in C_{b_3} \tag{2.8}$$

$$y_{f_2} - y_{f_1} + R\sin\theta_{f_1} \le \left(1 - z^{b_4}_{f_1,f_2}\right) \times M \qquad {}^{\forall}(f_1,f_2) \in C_{b_4} \tag{2.9}$$

$$y_{f_1} - y_{f_2} + R\sin\theta_{f_1} \le \left(1 - z^{b_5}_{f_1,f_2}\right) \times M \qquad {}^{\forall}(f_1,f_2) \in C_{b_5} \tag{2.10}$$

$$y_{f_2} - y_{f_1} \le \left(1 - z^{b_6}_{f_1,f_2}\right) \times M \qquad {}^{\forall}(f_1,f_2) \in C_{b_6} \tag{2.11}$$

$$y_{f_1} - y_{f_2} + d \times \alpha(f_2) \le \left(1 - z^{e_1}_{f_1,f_2}\right) \times M' \qquad {}^{\forall}(f_1,f_2) \in C_{e_1} \tag{2.12}$$

$$y_{f_2} - y_{f_1} - d \times \alpha(f_2) \le \left(1 - z^{e_2}_{f_1,f_2}\right) \times M' \qquad {}^{\forall}(f_1,f_2) \in C_{e_2} \tag{2.13}$$

$$x_{l,f_1} + x_{l,f_2} - 1 \le \sum_{i\in I^B_{f_1,f_2}} z^i_{f_1,f_2} \qquad {}^{\forall} f_1,f_2 \in F, f_1 \ne f_2,\ {}^{\forall} l \in L \tag{2.14}$$

$$x_{l,f_1} + x_{l,f_2} - 1 \le \sum_{i\in I^E_{f_1,f_2}} z^i_{f_1,f_2} \qquad {}^{\forall} f_1,f_2 \in F, f_1 \ne f_2,\ {}^{\forall} l \in L \tag{2.15}$$

$$M = R + H + 1 \tag{2.16}$$

$$M' = H + d + 1 \tag{2.17}$$

The objective function and each constraint equation are as follows;

- (1) denotes the objective function. In this research, we aim to minimize the number of films used.
- (2) guarantees that the film with the smallest number is used first.
- (3) guarantees that the film to which the fiber is routed has already been used.
- (4) guarantees that all fibers are routed to one of the films.
- (5) is a constraint that controls pairs of fibers that cannot be routed to the same film.
- (6) through (11) are constraints to satisfy design requirement (b).
- (12) and (13) are constraints to satisfy design requirement (e).
- (14) and (15) are the relationship between variables x and z.
- (16) and (17) represent big-M.

2.3.3 Heuristic method

Next, we describe a heuristic algorithm for Problem 1. This problem often fails to find a feasible solution within the desired computation time because the computation time increases when the number of optical fibers is large or the film height H is small. Therefore, we propose a heuristic algorithm to find a feasible solution quickly.

The heuristic algorithm uses the solution to the following problem as a lower bound on the number of layers used.

Problem 2. *Film number minimization problem*

$$
\begin{aligned}
\underset{u,x,y,z}{\textit{minimize}} \quad & \sum_{l \in L} u_l & & \\
\textit{subject to} \quad & u_l \leq u_{l-1} & & {}^{\forall} l \in L \setminus \{0\} \\
& x_{l,f_1} + x_{l,f_2} \leq 1 & & {}^{\forall}(f_1, f_2) \in C_{b_0} \cup C_{e_0},\ {}^{\forall} l \in L \\
& x_{l.f} \leq u_l & & {}^{\forall} f \in F_{C_{b_0} \cup C_{e_0}},\ {}^{\forall} l \in L \\
& x_{l,f} \in \{0,1\} & & {}^{\forall} f \in F_{C_{b_0} \cup C_{e_0}},\ {}^{\forall} l \in L \\
& u_l \in \{0,1\} & & {}^{\forall} l \in L
\end{aligned}
$$

Problem 2 is a version of Problem 1 in which the target of the optimization problem is restricted to optical fibers ($F_{C_{b_0} \cup C_{e_0}}$) belonging to $C_{b_0} \cup C_{e_0}$, and the constraints on the fibers are restricted to b_0, e_0. Therefore, the optimal solution to Problem 2 is the lower bound of Problem 1. This problem assigns the fibers to the film, like "Number Place," where the numbers are arranged according to the conditions [2]. In this section, the optimal objective function value of Problem 2 is N_{LB}. Further, solving Problem 2 yields the tentative assignment to films of optical fibers belonging to $C_{b_0} \cup C_{e_0}$. For each film, $0, \ldots, N_{LB} - 1$, denote the set of fibers assigned to $F_l (l = 0, \ldots, N_{LB} - 1)$.

Using F_l obtained by solving Problem 2, we consider each film's optimal optical fiber wiring method. This paper uses this routing method to maximize the number of fiber wires assigned to film l. This problem can be expressed as Problem 3.

Problem 3. *Optical fiber routing maximization problem*

$$
\begin{aligned}
\underset{x,y,z}{\textit{maximize}} \quad & \sum_{f \in F_l'} x_f & & \\
\textit{subject to} \quad & x_{f_1} + x_{f_2} \leq 1 & & {}^{\forall}(f_1, f_2) \in C'_{l,b_0} \cup C'_{l,e_0} \\
& y_{f_2} - y_{f_1} + R\sin\theta_{f_2} \leq \left(1 - z^{b_1}_{f_1,f_2}\right) \times M & & {}^{\forall}(f_1, f_2) \in C'_{l,b_1} \\
& y_{f_1} - y_{f_2} + R\sin\theta_{f_2} \leq \left(1 - z^{b_2}_{f_1,f_2}\right) \times M & & {}^{\forall}(f_1, f_2) \in C'_{l,b_2} \\
& y_{f_1} - y_{f_2} \leq \left(1 - z^{b_3}_{f_1,f_2}\right) \times M & & {}^{\forall}(f_1, f_2) \in C'_{l,b_3} \\
& y_{f_2} - y_{f_1} + R\sin\theta_{f_1} \leq \left(1 - z^{b_4}_{f_1,f_2}\right) \times M & & {}^{\forall}(f_1, f_2) \in C'_{l,b_4} \\
& y_{f_1} - y_{f_2} + R\sin\theta_{f_1} \leq \left(1 - z^{b_5}_{f_1,f_2}\right) \times M & & {}^{\forall}(f_1, f_2) \in C'_{l,b_5} \\
& y_{f_2} - y_{f_1} \leq \left(1 - z^{b_6}_{f_1,f_2}\right) \times M & & {}^{\forall}(f_1, f_2) \in C'_{l,b_6} \\
& x_{f_1} + x_{f_2} - 1 \leq \sum_{i \in I^B_{f_1,f_2}} z^i_{f_1,f_2} & & {}^{\forall}f_1, f_2 \in F'_l, f_1 \neq f_2 \\
& y_{f_1} - y_{f_2} + d \times \alpha(f_2) \leq \left(1 - z^{e_1}_{f_1,f_2}\right) \times M' & & {}^{\forall}(f_1, f_2) \in C'_{l,e_1} \\
& y_{f_2} - y_{f_1} - d \times \alpha(f_2) \leq \left(1 - z^{e_2}_{f_1,f_2}\right) \times M' & & {}^{\forall}(f_1, f_2) \in C'_{l,e_2} \\
& x_{l,f_1} + x_{l,f_2} - 1 \leq \sum_{i \in I^E_{f_1,f_2}} z^i_{f_1,f_2} & & {}^{\forall}f_1, f_2 \in F'_l, f_1 \neq f_2 \\
& x_f \in \{0, 1\} & & {}^{\forall}f \in F'_l \\
& y_f \in [0, H] & & {}^{\forall}f \in F'_l \\
& z^i_{f_1,f_2} \in \{0, 1\} & & {}^{\forall}f_1, f_2 \in F'_l, f_1 \neq f_2, \ {}^{\forall}i \in B \cup E \\
& M = R + H + 1 & & \\
& M' = H + d + 1 & &
\end{aligned}
$$

The fiber routing decision is made as follows. First, we solve Problem 2 only for the fibers belonging to $C_{b_0} \cup C_{e_0}$ among the fibers to be routed. Based on the result, tentatively assign ($F_l (l \in L)$) from among the optical fibers belonging to $C_{b_0} \cup C_{e_0}$. Next, solve Problem 3 sequentially, starting with the film with $l = 0$ to assign the fiber to the film. The target optical fibers for Problem 3 are F_0 fibers and the fibers not in all the fiber pairs in $C_{b_0} \cup C_{e_0}$. We solve this to determine the fiber assignment and bend position for $l = 0$. Let F_0^{NG} be the fiber that could not be routed, and use it as input when solving the routing problem for the film with $l = 1$. In other words, when solving Problem 3 for a film with $l = 1$, the optimization problem's target fibers are F_1 and F_0^{NG}. This process is repeated until all the fibers are wired. This process is shown in Algorithm 1.

Algorithm 1 Heuristic method

Require: F : Optical fibers set , $C_i \subset F \times F(^{\forall} i \in B \cup E)$: Sets of fiber pairs corresponding to the constraint i

Solve Problem 2 for $F_{C_{b_0} \cup C_{e_0}}$ to obtain the tentative assignment $F_i (i = 0, \ldots, N_{LB} - 1)$.
$l = 0$
$F'_0 \leftarrow F_0 \cup (F \setminus F_{C_{b_0} \cup C_{e_0}})$
while True **do**
    Solve Problem 3 with F'_l as input to obtain F_l^{NG} for the film l
    $F'_{l+1} \leftarrow F_{l+1} \cup F_l^{NG}$
    $l \leftarrow l + 1$
    if All fibers are wired **then**
        break
    end if
end while

2.4 Numerical Experiments

2.4.1 Problem Setting

We used sample data from the data used in our joint research with Sumitomo Electric. An example is shown in Fig. 2.7. At the top and bottom ends of a wiring area with width $W = 423$[mm] and height $H = 160$[mm], 12 optical fibers are lined up in groups of 12 fibers each. At the top end, the fibers in line are designated A1, ..., A12, B1, ..., B12, ..., L1, ..., L12 from the left. At the lower end, the fibers are arranged in the order of A1, ..., L1, A2, ..., L2, ..., A12, ..., L12 from the left. The distance between each group is 30[mm], and the distance between each fiber is 0.25[mm].

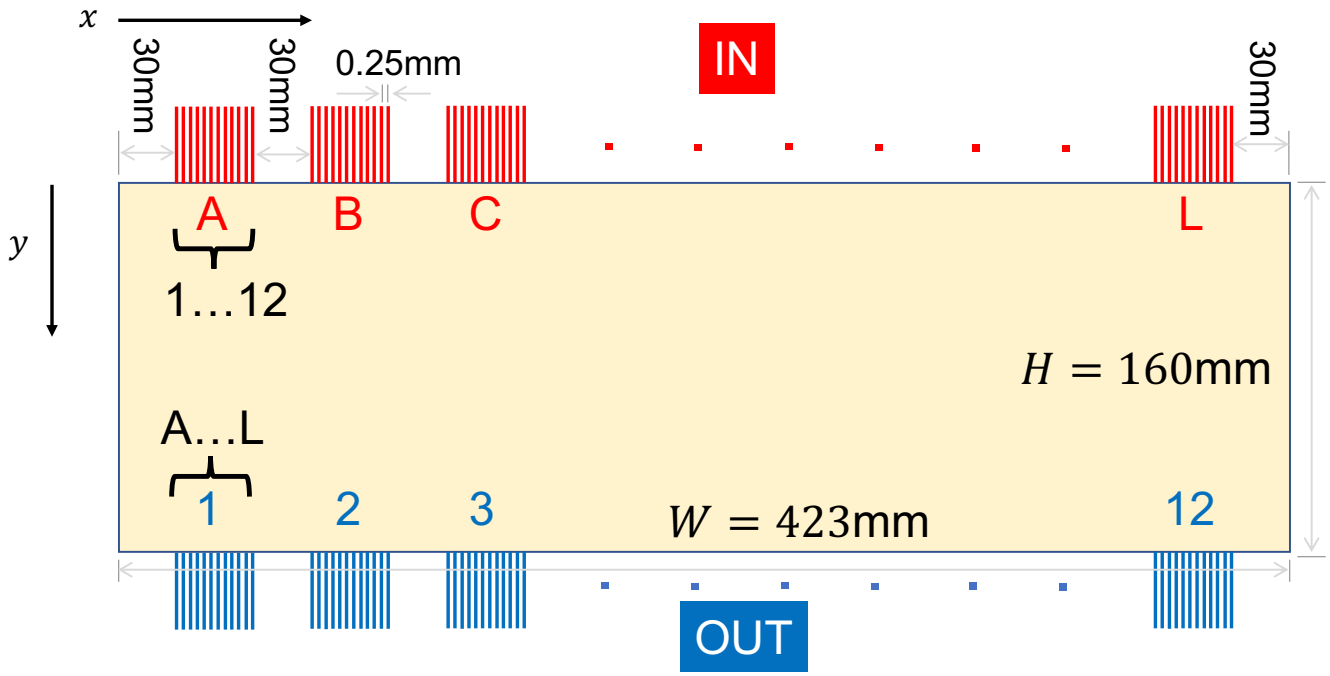

FIGURE 2.7: An example of fibers and a film

The rule-based wiring method used in the experiment is introduced here. Optical fiber wiring using film is a new method, and the rule-based method should be simple due to its expandability. Four rule-based methods to make it easy for workers to perform wiring tasks have been prepared for comparison.

R1: Randomly select and route fibers. The bend position should be at the bottom of the film whenever possible when bending fibers.

R2: Randomly select and route fibers. The bend position should be at the top of the film whenever possible when bending fibers.

R3: Wire in the order of A1, ... , A12, B1, ..., B12, ..., L1, ..., L12. The bend position should be at the bottom of the film whenever possible when bending fibers.

R4: Wire in the order of A1, ... , A12, B1, ..., B12, ..., L1, ..., L12. The bend position should be at the top of the film whenever possible when bending fibers.

Additionally, when executing the exact solution method described in Sect. 2.3.2, it is necessary to determine the number of usable films (n) in advance. In this experiment, Problem 1 is calculated from $n = 1$, and if a solution cannot be found, the value of n is increased by one, and the calculation is repeated. This loop is repeated until a solution is found and the wiring method is obtained. In the heuristic method, when solving Problem 2, the calculation is repeated with increasing values of n from $n = 1$ until a solution is found.

This paper evaluated the proposed method's performance based on sample data and settings generated from previous studies. The various settings were then modified to explore the conditions under which the number of films used could be efficiently reduced.

2.4.2 Experiments setting

This section first describes the settings related to the experiments reported in this paper. The settings used in the experiments are shown in Table 2.1. "exp1" is the basic setup. "exp2" and later are modifications of the wiring environment that depart from "exp1."

TABLE 2.1: Experiments setting. Where R is the minimum radius of curvature, s is the spacing of the optical fibers within the group, H is the height of films, and d is the minimum radius of curvature of the optical fibers

	R [mm]	s [mm]	H [mm]	d[mm]
exp1	15	0.25	160	0.1
exp2	50	0.25	160	0.1
exp3	**10**	0.25	160	0.1
exp4	15	0.5	160	0.1
exp5	15	**0.1**	160	0.1
exp6	15	0.25	80	0.1
exp7	15	0.25	**320**	0.1
exp8	15	0.25	160	0.2
exp9	15	0.25	160	**0.01**

Additionally, we set a time limit for solving the optimization problem with the proposed method. This limit prevents the computation from taking an enormous amount of time to find the optimal solution, preventing the completion of the computation. We set the time limit based on the time required to find a tentative solution. In this problem set-up, a tentative solution close to the optimal solution could be obtained in less than one hour for most problems. Therefore, the time limit for solving one optimization problem was one hour.

The computer environment for each experiment and the optimization solver used to solve the optimization problem are shown in Table 2.2. Each experiment was run five times. The resulting values are the average of the five runs. Twenty central processing unit (CPU) cores were used to solve the optimization problems using the exact solution and heuristic methods. Computation time was measured for all experiments as the time taken from the start of the run to the determination of the wiring method.

2.4.3 Experimental results

Basic Setting

TABLE 2.2: Computational Environment

CPU	Intel(R) Xeon(R) Silver 4214R CPU
Clock rate	2.40GHz
Memory	192GiB
Solver	Gurobi 10.0.1

The results of "exp1" are shown in Fig. 2.8. R1 to R4 required 9 or more sheets of film. On the other hand, the exact solution method, which utilizes an optimization problem, reduced the number of films used to two. The heuristic method, devised based on the exact solution method, reduced the number of films to three. In terms of computation time, each of the rule-based methods took only about 15 seconds. The exact solution and heuristic methods using optimization problems were able to find the wiring method in about 30 minutes.

Based on these results, the proposed method utilizing the optimization problem can find a way to route optical fibers using fewer films than the rule-based method. However, differences in the number of films used occurred between the exact solution and heuristic methods. These differences are thought to be because the more options there are for wiring methods when routing fibers, the easier it is to reduce the number of films used. The rule-based method determines how to route each optical fiber to which film one by one. In the heuristic method, fibers that cannot be routed to the same film are assigned to a film first, and then the fiber routing is considered for each film. In the exact solution method, all optical fibers are considered in terms of how to assign them to films and how to wire them. In other words, the number of films used is affected by the complexity of how optical fibers are assigned to films.

On the other hand, the computation time required to find the optical fiber routing method tends to be longer for heuristic and exact solution methods that use optimization problems than for rule-based methods. However, the actual time taken was about 30 minutes. Wiring optical fibers requires determining the wiring method in advance rather than considering it in real time during manufacturing. Therefore, a certain amount of calculation time can be secured for practical use. Hence, it can be said that the exact solution and heuristic methods, which can determine the wiring method in about 30 minutes, can be put to practical use without problems.

The computation time for optimization problems varies greatly depending on the problem set. For example, with the "exp1" setting, the calculation can be completed in about 30 minutes. However, the calculation time can vary significantly when conditions such as bend radius and film height change. In the next section, we consider more suitable conditions for optical fiber wiring based on changes in the number of films used and the calculation time by modifying the wiring conditions.

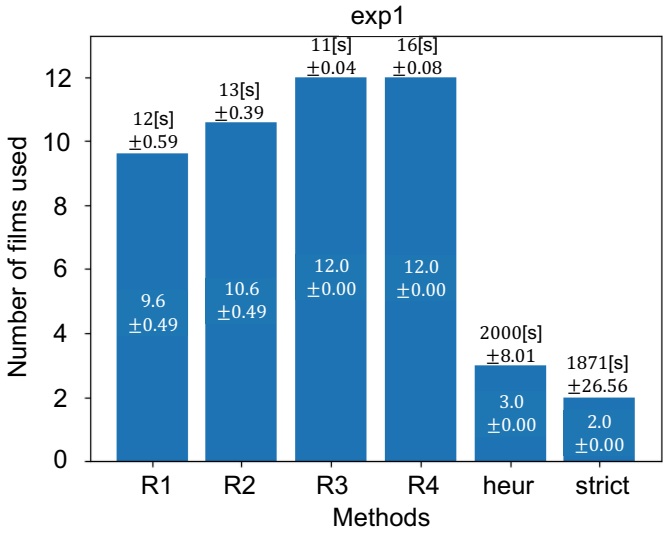

FIGURE 2.8: Result of "exp1." "heur" represents the heuristic method (Sect. 2.3.3), and "exact" represents the exact solution method (Sect. 2.3.2). The numbers above the bars are the average time taken for the calculation. The numbers in the bars are the average number of films used. The standard deviation over five runs is also listed below each value

Effective wiring design studies

We examined whether changing the related settings makes more efficient wiring possible. In Table 2.1, "exp2" to "exp9" are the wiring conditions in which one of "bend radius," "fiber spacing within a group," "film height," or "minimum adjacent distance" is changed to depart from the primary setting, "exp1." We discuss better LOFA settings based on the number of films used and the calculation time.

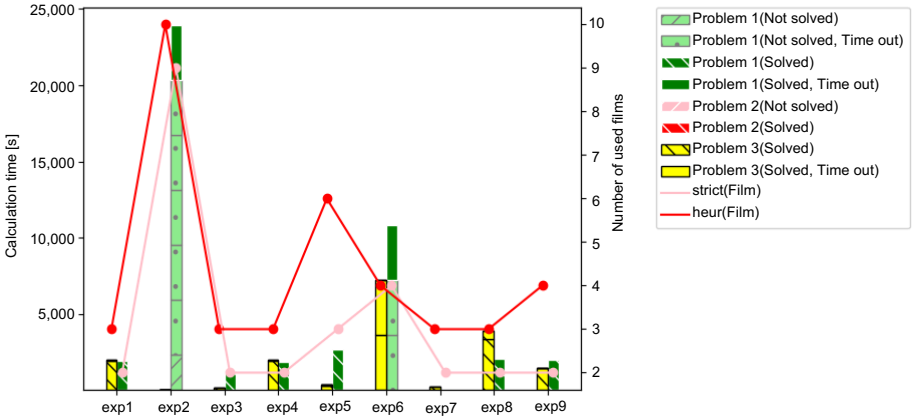

FIGURE 2.9: Comparison of the number of films used and computation time. Bar graphs represent the number of films used. The bar on the left side of each experiment represents the heuristic method results, and the bar on the right side represents the exact solution method results. The axis uses the values on the left side. Line graphs show the time taken to find the solution. The axis of line graphs uses the numbers on the right

The number of films obtained using the exact solution and heuristic methods in each experimental setting is shown in the line graph in Fig. 2.9. The time used for the calculations is shown in the bar graph in Fig. 2.9. The color and shape of the graphs change with the type of problem solved and the post-calculation status. The post-calculation status was determined by whether the solver found a solution to

TABLE 2.3: The film setup used in each experiment and the film's cost. H is the height of the film, and W is the width of the film. "cost" was calculated as the ratio of the size of each film when the area of the film in "exp1" was set to 1

	Film		
	H	W	Cost
exp1	160	423	1.00
exp2	160	423	1.00
exp3	160	423	1.00
exp4	160	**456**	**1.08**
exp5	160	403.2	0.95
exp6	80	423	0.50
exp7	**320**	423	**2.00**
exp8	160	423	1.00
exp9	160	423	1.00

the problem and whether the calculation was completed within the time limit. Among all experiment settings, "exp3," "exp4," "exp7," and "exp8" obtained results equivalent to "exp1," the basic setting. Furthermore, the time taken to find the solution is compared. The results show that "exp3" and "exp7" obtained solutions faster than "exp1" among the four settings that obtained results equivalent to "exp1." "exp4" took almost as long as "exp1" to obtain a solution, and "exp8" took longer than "exp1." Some settings require more than one hour to find a solution when using the exact solution method.

These differences are discussed in terms of the different experimental setups. "exp3" reduces the size of the bending radius R. This alteration makes the bending area smaller in the wiring and increases the number of places where fibers can cross each other. In other words, this increases the number of optical fibers that can be wired to a single film. "exp7" increases the optical fibers wired in a single film by increasing the film height H. On the other hand, in "exp8," the minimum adjacent spacing d is larger than in "exp1." This alteration means fewer optical fibers wired in a single film. In addition, "exp4" increases the fiber spacing within a group, making it harder for bends to overlap with neighboring fibers. However, "exp4" did not show any significant difference from the result of "exp1." This result is probably because the minimum bend radius is so large compared to the size of the adjacent fiber spacing that the number of optical fibers routed on a single film is not significantly different from "exp1." Based on these results, we consider that designing films with more optical fibers that can be routed on a single film reduces the time required to find the routing.

Next, let us consider the feasibility of these settings. "exp4" and "exp7" affect the size of the film. The specific values are shown in the "Film" column of Table 2.3. In this paper, we assume that the cost associated with film production is proportional to the area of the film. The cost of each set when the cost

of "exp1" is set to one is listed in the "Cost" section of Table 2.3. These values are directly related to the size of the film used, i.e., the cost related to the fabrication of the film used. Thus, "exp4" and "exp7" are wiring designs that have higher film production costs than "exp1." On the other hand, the film-related settings for "exp3" are the same as for "exp1." The film production cost is the same as for "exp1." In other words, the "exp3" setting is a design that can reduce the wiring cost, having the same film production cost as "exp1."

From the above, we think it is important to set the number of optical fibers routed on a single film to be significant to consider a better LOFA. For this purpose, it is considered to be a good idea to change the height of the film, the spacing of fibers in a group, and the fibers' bending radius. However, since the film's height and the fibers' spacing in a group affect the cost of fabricating the film, it is necessary to consider the balance with the cost of wiring. Additionally, when the fiber's bending radius is reduced, the risk of transmission loss is adversely affected. Therefore, when determining the wiring design, using the smallest value of the allowable bending radius is desirable.

Computation time characteristics differ significantly between Problem 1, 3, and Problem 2. Problem 1 and 3 are mixed integer programming problems, while Problem 2 is an integer programming problem. Problem 2 was solved within approximately five seconds in all experimental conditions. Problems 1 and 3 required more time to solve each problem than Problem 2, which had different characteristics. For Problem 1, the computation time it differed greatly depending on the problem setup and the number of available films (n). For $n = 1$, the problem was found to be infeasible in a very short time, less than 0.1 seconds. After $n = 2$, finding the optimal solution for some problems was possible. When it was possible to find the optimal solution, the time required to find the solution varied depending on the conditions of the experiment. It took about 80 seconds in the shortest (exp7) and the longest, about 2,000 seconds (exp8, 9). When n was more significant than two, most cases where the problem could not be solved were those where no solution could be found within an hour (exp2, 6). In Problem 3, there were no cases where the problem was not solved. Problem 3 cannot be solved when not a single fiber can be routed to a film. However, at least one fiber can always be routed to one film. Therefore, we consider that there was no case in which Problem 3 could not be solved.

Also, let us consider the total time taken for the calculation. For "exp2" and "exp6," obtaining even a tentative solution in one hour was impossible when n was small. In these cases, the heuristic method can find a solution faster than the exact solution method. Therefore, it is necessary to appropriately select the method to be used depending on the conditions of the problem. For example, in this experiment, when the number of fibers that can be wired to a single film is greater than the number of fibers in "exp1," it is thought that the exact solution method can also find a wiring method in a certain amount of time. On the other hand, if the film size becomes smaller than the "exp1" setting, for example, and the number of fibers that can be wired to a film becomes smaller, it becomes difficult to solve the problem using the exact solution method. Therefore, heuristic methods are an effective tool in such situations.

Chapter 3

Scheduling system for automated storage and retrieval system with multiple machines using a time-expanded network

3.1 Background

In recent years, automated storage and retrieval systems (AS/RSs) have become popular in the industrial world owing to developments in control technology. AS/RSs are equipped with storage/retrieval (S/R) machines, which can store and retrieve *loads* without human labor. Efficient operation of AS/RSs significantly improves warehouses' operational efficiency and shortens the lead time required for production.

An AS/RS generally consists of racks arranged on either side of one or more parallel aisles. Loads are stored and managed in a place called a *cell*, and a rack consists of several cells. The AS/RS also stores and retrieves loads and automatically transports loads using S/R machines. The AS/RS has input/output (I/O) stations, where loads are taken in and out. There are two types of requests: storage requests, which take and store loads in the AS/RS, and retrieval requests, which take loads in the AS/RS outside. When a user inputs a retrieval request, the corresponding load is selected from the AS/RS and retrieved to an I/O station by a specific S/R machine. When a user inputs a storage request, the AS/RS allocates a storage cell, and an S/R machine receives the load at an I/O station and transports the loads into the storage cell. Using S/R machines drastically reduces human resources required for receiving and storing compared with conventional warehouses. Moreover, by controlling S/R machines appropriately and shortening loading and unloading times, the AS/RS can transport loads efficiently.

Research on improving the efficiency of AS/RSs has been widely conducted. Roodbergen and Vis [24] classified studies on AS/RSs according to their objectives and led a comprehensive survey. Boysen and Stephan [5] classified AS/RSs according to their layouts, request characteristics, and objective functions and investigated the scheduling problem for a single S/R machine in an AS/RS.

In many studies, an S/R machine departs from an I/O station, visits cells, and returns to an I/O station. This movement is often called a cycle. The problem of scheduling requests in an AS/RS is often

similar to the traveling salesman problem (TSP) because the S/R machine moves between I/O stations and multiple cells [5]. Gharehgozli et al. [11] modeled the problem as an asymmetric TSP to minimize the total travel time of a single S/R machine, given a set of storage and retrieval requests, where I/O stations are located at both ends of an aisle. They proposed an algorithm for determining the optimal solution in polynomial time under several assumptions. When an S/R machine can transport multiple loads at a time, the combination of loads to be transported and the optimal route can be simultaneously found by formulation as TSP [29, 7].

There are different AS/RSs from the conventional AS/RSs in which the S/R machine moves directly between racks and I/O stations. Hu et al. [15] proposed a split-platform AS/RS (SP-AS/RS) that uses multiple S/R machines to transport loads. An SP-AS/RS has vertically moving S/R machines and horizontally moving S/R machines for efficiently transporting additionally heavy loads, such as marine containers. In an SP-AS/RS, the movement range is limited for each S/R machine. Vertically moving S/R machines have access to I/O stations, whereas horizontally moving S/R machines have access to cells. That is, the load moves between the cell and I/O station using multiple S/R machines. Carlo and Vis [6] devised an algorithm that allows multiple S/R machines to move vertically and share a path to execute a given request without collision, eliminating the SP-AS/RS bottleneck.

In our study, we aim to minimize the operation time required to complete storage and retrieval requests in more complex AS/RS as a joint study with Rohto Pharmaceutical Co. (Rohto). We targeted an AS/RS installed in one factory located in Japan but modeled it as a generic warehouse to be deployed in other factories owned by Rohto. Rohto does not use additionally heavy loads, thereby eliminating vertical and horizontal restrictions on the movement direction of the S/R machine and enabling a more flexible AS/RS design. In addition, the consideration of optimization only for the vertical scheduling of S/R machines was sufficient for the split-platform AS/RS. However, in Rohto's AS/RS, particular S/R machines do not become bottlenecks owing to the size of loads to be transported, AS/RS structure, and performance of S/R machines. Therefore, by optimizing the scheduling of all S/R machines simultaneously, we can now reduce the time required to complete all storage and retrieval requests in an AS/RS in which each S/R machine works independently and multiple S/R machines carry loads. We called AS/RSs with these characteristics multi-control AS/RSs (MC-AS/RSs). Rohto uses cranes and sorting transfer vehicles (STVs) as S/R machines. Figure 3.1 shows an example of an MC-AS/RS.

Among MC-AS/RSs, we focused on optimizing the scheduling for the MC-AS/RS, which is a unit-load AS/RS. That is, the AS/RS handles each load as a unit called a *pallet* and moves and stores loads according to requests. In addition, the MC-AS/RS has buffer stations, which are temporary storage areas used by one S/R machine to pass pallets to another S/R machine. As all buffer stations have the capacity, controlling each S/R machine requires paying attention to whether pallets can be placed on buffer stations.

This paper proposes a method for the MC-AS/RS to transport loads efficiently. In our target AS/RS, requests are automatically assigned to S/R machines according to the input order. Therefore, we present

a method for scheduling storage and retrieval requests and creating a request order to be given to S/R machines. The MC-AS/RS simultaneously controls multiple S/R machines to transport loads, making its structure and control methods complex; thus, the sequence and timing of conveyance and AS/RS state must be considered. As the number of requests and S/R machines increases, the number of combinations increases, making devising efficient operations difficult. We modeled the MC-AS/RS using a time-expanded network (TEN) to consider these combinations. TEN is a dynamic graph with time information converted into a static graph. In this study, TEN represents the travel paths of requests and S/R machines in the AS/RS. We formulate the operational efficiency of the MC-AS/RS as a problem of minimizing the sum of requested execution times on the TEN. We generate the *requests order* for practical use based on solving this problem. In the MC-AS/RS targeted in this study, the area in which each S/R machine can operate is limited. Furthermore, only one S/R machine is operating in an area. Therefore, collisions between S/R machines are not considered. However, the MC-AS/RS can cause deadlocks at buffer stations while carrying loads. If deadlocks occur, loads cannot be transported. The deadlocks in this paper are described in detail in Sect. 3.3.3. Deadlocks are avoided by generating the buffer stations' situations.

The proposed method was tested using a simulator created based on AS/RS, owned by Rohto and installed at one of its factories. With this AS/RS, obtaining the data necessary to optimize the scheduling of the S/R machines for a day's work at the start of operations is impossible. For example, some storage requests involve storing loads transported by trucks from outside locations or warehouses. Storage requests for these loads are generated when trucks arrive at the AS/RS and preparations for storage are completed. However, predicting accurately when the trucks will arrive at the AS/RS is impossible. For some other reasons, due to manual work in the pharmaceutical process, the loading and unloading of loads depend on the actions of the workers. In other words, the time when requests occur is determined dynamically as the workers perform the work. For these reasons, optimizing the schedule of S/R machines at once is not easy. Therefore, we first devised a method to optimize the short-term schedule of S/R machines. This paper aims to finish executing the generated requests as soon as possible. This objective was set at the demand of Rohto. The experiment used a simulator to check whether the generated requests were executed as quickly as possible.

The remainder of this paper is organized as follows. Section 3.2 overviews the AS/RS and problem addressed in this paper, and Sect. 3.3 describes our proposed method. The performance of the proposed method is verified in Sect. 3.4.

3.2 Problem description

Since 2019, we have conducted our research using the model case of a factory in Mie Prefecture, Japan, owned by Rohto. It has an MC-AS/RS installed in the factory that manufactures cosmetics and pharmaceuticals. The MC-AS/RS stores and retrieves raw materials, semi-finished products, and finished products

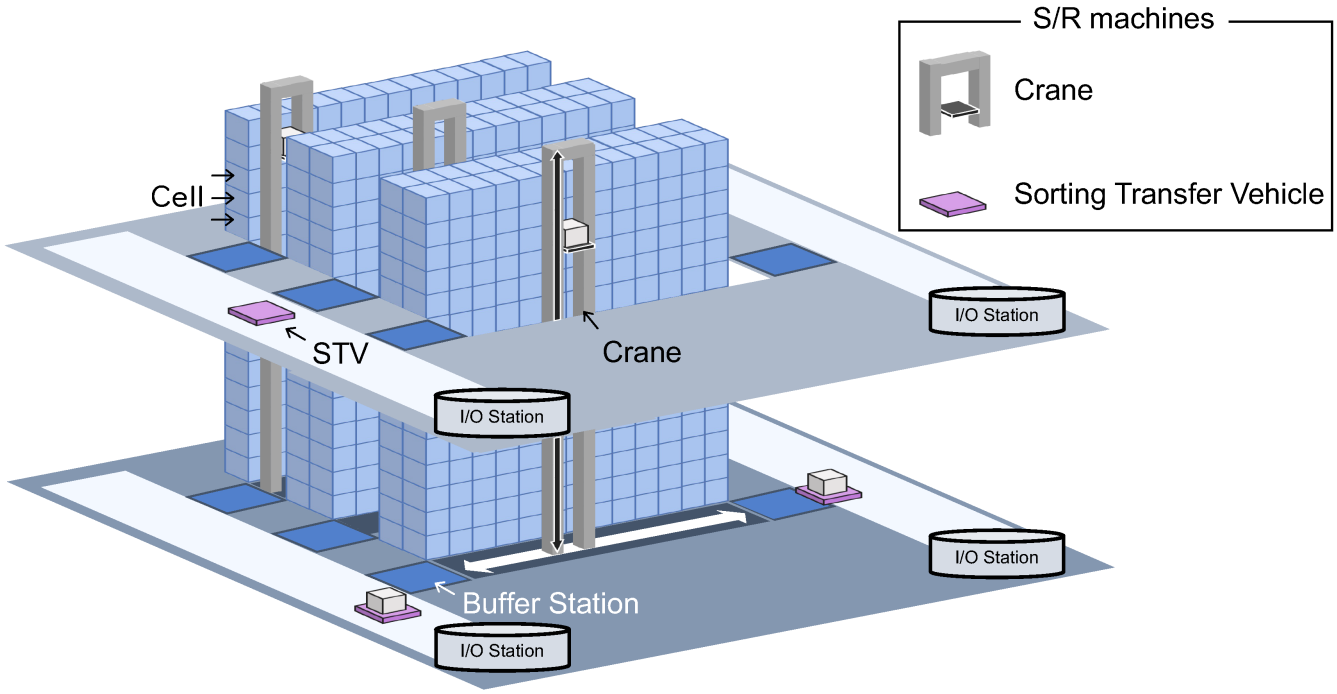

FIGURE 3.1: Overview of multi-control AS/RS

required for manufacturing. This section describes the overall setup of our target AS/RS (target-AS/RS) and problem encountered when creating a request order. The target-AS/RS is an MC-AS/RS based on the AS/RS used in the model case. Section 3.4.1 describes detailed settings for the number of target-AS/RS racks and S/R machine performance.

3.2.1 MC-AS/RS settings

An MC-AS/RS, including the target-AS/RS, has the following features in its structure and setting:

- Each S/R machine operates independently.
- Loads are transported using multiple S/R machines.
- There are temporary storage places for loads called buffer stations. These are used when an S/R machine transfers a load to another S/R machine.
- Buffer stations have capacity. That is, the number of pallets that can simultaneously exist in a buffer station is limited.
- The target-AS/RS is a unit-load AS/RS. In the target-AS/RS, a pallet is used as a unit.

3.2.2 Target-AS/RS settings

The target-AS/RS has the following assumptions as an AS/RS. These are based on the settings adopted in the model case:

- Only one pallet can be stored in a cell.
- Each S/R machine exists only one in its operating range. That is, the collision between S/R machines is not considered.
- An S/R machine can accept a request at any location and stand by without loading/unloading.

These settings include many settings in the existing researches (e.g. [27, 5, 11]) and we focus on AS/RSs that have multiple S/R machines and use multiple S/R machines to carry pallets.

3.2.3 S/R machines settings

We describe the settings of S/R machines installed in the target-AS/RS. These are based on the S/R machines installed in the model case:

- The capacity of an S/R machine is one. That is, one S/R machine can carry only one pallet.
- Some S/R machines do not have direct access to the racks in the AS/RS.
- A pallet is stored in a buffer station by one S/R machine to pass the pallet to another S/R machine. That is, buffer stations must be used if multiple S/R machines transport a pallet.
- S/R machines work to execute requests according to the order of the request order.

The target-AS/RS has two types of S/R machines, cranes and STVs. In the target-AS/RS, cranes can access cells in racks and buffer stations, and STVs can transport pallets between I/O stations and buffer stations. For example, when a storage request is executed, an STV first transports the pallet from the I/O station to the buffer station. Then, the crane moves the pallet from the buffer station to the storage cell, completing the request. We consider that this method applies to AS/RSs, where S/R machines operate independently.

3.2.4 Storage and retrieval request settings

This section describes how to set up storage and retrieval requests. In the target-AS/RS, S/R machines carry pallets according to requests. Requests contain the following information:

- A pallet transfer source.
- A pallet transfer destination.
- Occurrence time of a request.

The S/R machines used by requests are determined by the source and destination information. If the request is a storage request, the destination cell is assumed to be vacant in advance. If the request is a

retrieval request, a pallet is assumed to be stored in the retrieval source cell in advance. That is, storage and retrieval requests for the same pallet do not co-occur, and the pallet of the storage request is not stored in the cell vacated by the execution of the retrieval request. Therefore, considering dependencies between requests in terms of their execution order is unnecessary.

3.3 Proposed method

In this section, we describe our proposed methodology. The proposed methodology is primarily divided into a *generator*, *optimizer*, and *scheduler*. First, the generator takes requests, S/R machines, and the AS/RS structures as input and models the target-AS/RS as a TEN [10]. Next, the optimizer formulates and solves the problem of improving the efficiency of the AS/RS using information from requests, S/R machines, and TEN. This paper aims to minimize the total execution time of requests to improve AS/RS efficiency. Finally, the scheduler generates the request order for the S/R machines based on the solution computed by the optimizer.

If S/R machines carry pallets only according to the request order, they may be unable to carry pallets and process requests. This situation is called a *deadlock*. Section 3.3.3 describes examples of deadlocks. Deadlocks occur when buffer stations have a limited capacity. Therefore, the scheduler also generates the transition of each buffer station from the optimizer's solution. We confirmed that a deadlock does not occur via simulation using the request order and the transition of buffer stations.

Figure 3.2 overviews our proposed method. The proposed method acquires information from running the AS/RS, generates transport instructions using a graph, and verifies that deadlock does not occur. By giving a request order to the AS/RS, the operation of the AS/RS can be optimized. In this study, we address the problem of performing this cycle once. We consider that repeating this cycle improves the operational efficiency of AS/RSs.

Section 3.3.1 describes creating a TEN for the AS/RS to be executed in our generator. Then, Sect. 3.3.2 describes the formulation of the optimization problem on the TEN performed by the optimizer, and Sect. 3.3.3 describes the deadlock and basic idea of how to avoid it. Finally, Sect. 3.3.4 describes the algorithm that our scheduler executes.

3.3.1 Construction of an AS/RS time-expand network

This section describes how our generator uses information from an AS/RS, storage and retrieval requests and S/R machines to model AS/RS. A graph $G = (N, A)$ is created based on the information of requests, S/R machines, and the AS/RS. R is a set of requests. Node $n \in N$ represents a specific location in the AS/RS corresponding to I/O stations, buffer stations, and cells. In particular, only the locations necessary to transport loads in requests are nodes in the graph. Every request uses the corresponding I/O station, buffer station, and cell once. If the initial location of each S/R machine is not included in N, that location

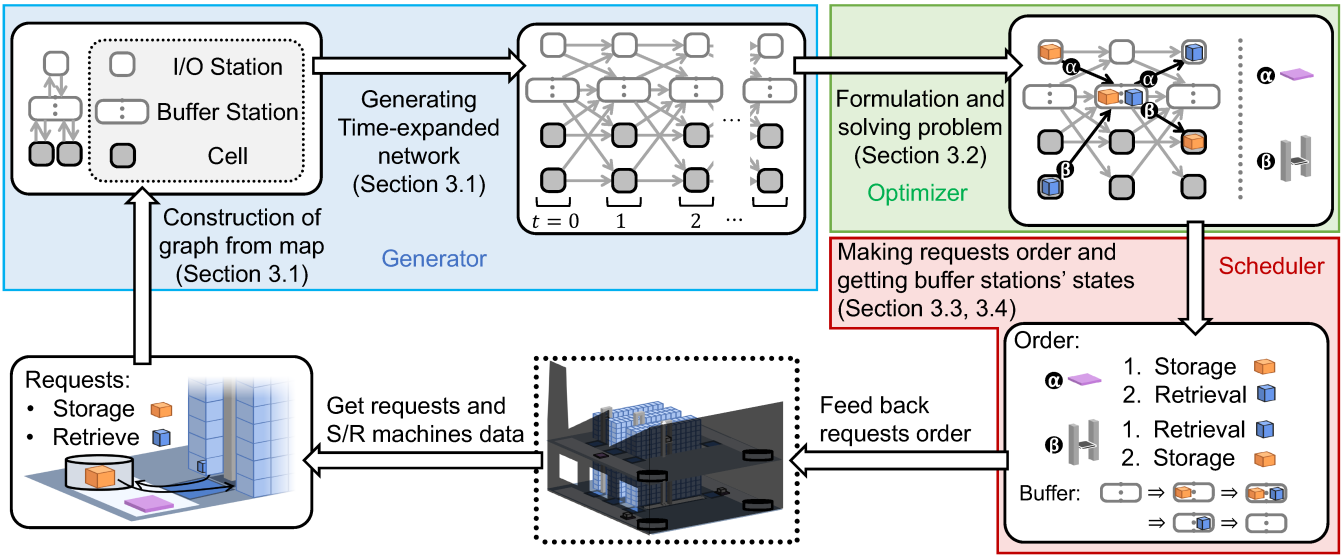

FIGURE 3.2: Flowchart of the proposed method. A cycle starts from the factory and proceeds clockwise

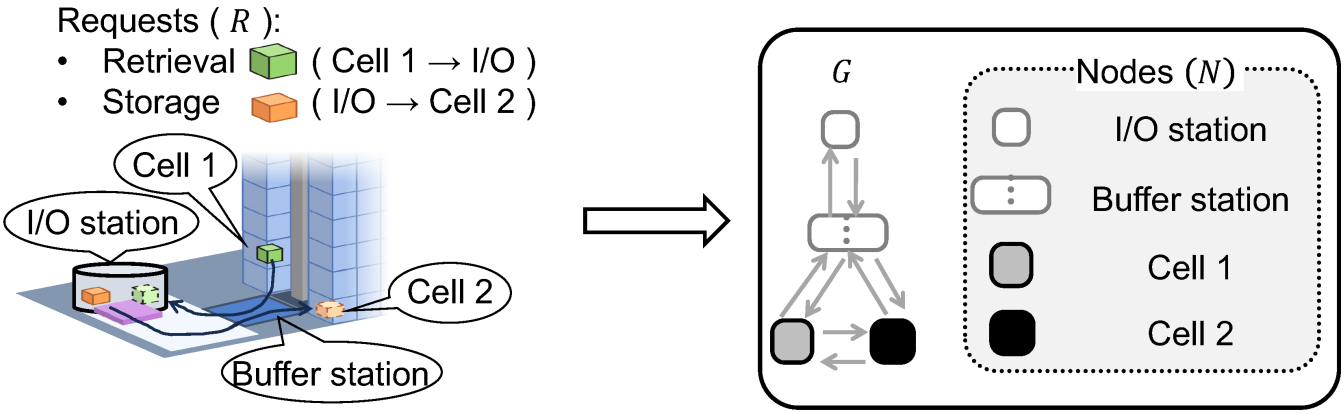

FIGURE 3.3: Example of graph G constructed from two requests

is also added to N as a node. An arc $a \in A$ from $n \in N$ to $n' \in N$ represents how a single S/R machine can move the pallet between the two nodes; restricting N to the set of the start and end points of all requests in R and initial points of all S/R machines does not affect the subsequent optimization process.

Figure 3.3 shows an example of G. Assume that there are one retrieval request and one storage request in R. In this case, the graph nodes are the places where the two requests pass through. In other words, four nodes are created: I/O station, buffer station, cell 1, and cell 2. Then, if the initial position of the transporter does not exist among the nodes created so far, it is added to graph G as a node. Arcs are added between two points where the transporter can move directly.

We construct a TEN $G^{\mathcal{T}} = (N^{\mathcal{T}}, A^{\mathcal{T}})$ based on graph G. The symbols used in TEN are defined as follows:

- $\Delta \in \mathbb{R}_+$: A time granularity.
- $U \in \mathbb{N}_+$: An optimization period.

- $\mathcal{T}$: A discretization of $[0, U]$, representing a set of time points. The number of elements of $\mathcal{T}$ is $\lceil U/\Delta \rceil$.
- $N^{\mathcal{T}}$: A set of nodes. $N^{\mathcal{T}}$ can be written as a disjoint union: $N^{\mathcal{T}} = N_G^{\mathcal{T}} \cup L$.
  - $N_G^{\mathcal{T}}$: A set of nodes N expanded by $\mathcal{T}$.
  - L: A set of supersources and supersinks.
- $A^{\mathcal{T}}$: A set of arcs. $A^{\mathcal{T}}$ can be written as a disjoint union: $A^{\mathcal{T}} = A_G^{\mathcal{T}} \cup H^{\mathcal{T}} \cup A^L$.
  - $A_G^{\mathcal{T}}$: A set of arcs A expanded by $\mathcal{T}$.
  - $H^{\mathcal{T}}$: A set of staying arcs.
  - A^L: A set of arcs about supersource and supersink.
  - $\delta_+(n) \subset A^{\mathcal{T}}$: A set of arcs whose head is n.
  - $\delta_-(n) \subset A^{\mathcal{T}}$: A set of arcs whose tail is n.

Let $\Delta \in \mathbb{R}_+$ denote the unit time for discretizing time on TEN. If the optimization period is $U \in \mathbb{N}_+$, U can be divided into $\lceil U/\Delta \rceil$ points using Δ. For example, when $U = 600$ s and $\Delta = 10$ s, the TEN contains $\#\mathcal{T} = \lceil U/\Delta \rceil = 60$ periods. We define $\mathcal{T} = \{0, 1, \ldots, \lceil U/\Delta \rceil - 1\}$ as the discretized time set of these time points. Δ and U should be determined appropriately depending on the AS/RS and requests.

We extend graph G using the time-point set $\mathcal{T}$. Figure 3.4 shows an example. First, node set N in graph G is extended with $\mathcal{T}$. A node $n(t) \in N_G^{\mathcal{T}}$ denotes a node $n \in N$ at time $t \in \mathcal{T}$. Then, all arcs are extended with $\mathcal{T}$. The travel time from the start point to the endpoint is defined for each arc of G. Consider the expansion of the arc using this travel time and $N_G^{\mathcal{T}}$. For example, an arc $a = (n, m) \in A$ expands as follows. First, note that arc a has two different travel times.

(1) Time required for S/R machines to move from $n \in N$ to $m \in N$.

(2) Time required by S/R machines to move from $n \in N$ to $m \in N$ + time required to load and unload pallets.

Let $t_{n,m}$ be the discretization of (1) and $t'_{n,m}$ the discretization of (2). For $t_{n,m} \neq t'_{n,m}$, $A_G^{\mathcal{T}}$ has both arcs $(n(t), m(t + t_{n,m}))$ and $(n(t), m(t + t'_{n,m}))$, where $\forall t, t + t_{n,m}, t + t'_{n,m} \in \mathcal{T}$. Request flows can only pass through $(n(t), m(t + t'_{n,m}))$, while S/R machine flows can pass through $(n(t), m(t + t_{n,m}))$ and $(n(t), m(t + t'_{n,m}))$. For $t_{n,m} = t'_{n,m}$, $A_G^{\mathcal{T}}$ has arcs $(n(t), m(t + t_{n,m}))$, where $\forall t, t + t_{n,m} \in \mathcal{T}$. We also add staying arcs $H^{\mathcal{T}}$, which indicate that the pallets and S/R machines are stationary and not moving. A staying arc moves from nodes $n(t) \in N_G^{\mathcal{T}}$ to $n(t + 1) \in N_G^{\mathcal{T}}$. These are the arcs shown in Fig. 3.4 (b) that connect from one of the nodes in Fig. 3.4 (a) to the same node.

In addition, supernodes are created for request and S/R machine flows. First, we create a supersource (l_r or l_μ) for each request (r) and S/R machine (μ). Then, an arc is added from each supersource to each

initial position of the requests and S/R machines. For example, Fig. 3.4 (b) has these arcs connected to nodes in the AS/RS from outside the AS/RS. Next, supersinks (l'_r) are set for each request r. For example, let n_r be the destination of a request r. We can then check whether r has completed its transportation by checking whether the flow of r flowed from node $n_r(t)$ to the supersink at time $t \in \mathcal{T}$. Requests that cannot be completed by the end time pass through an arc leading from the node at the end time to the supersink. Moreover, the supersink (l'_μ) of S/R machine μ guarantees that the flow will finish flowing by connecting the arc from the last time of all nodes that μ can pass through. These arcs connect from the graph of the AS/RS to a node outside the AS/RS in Fig. 3.4 (b).

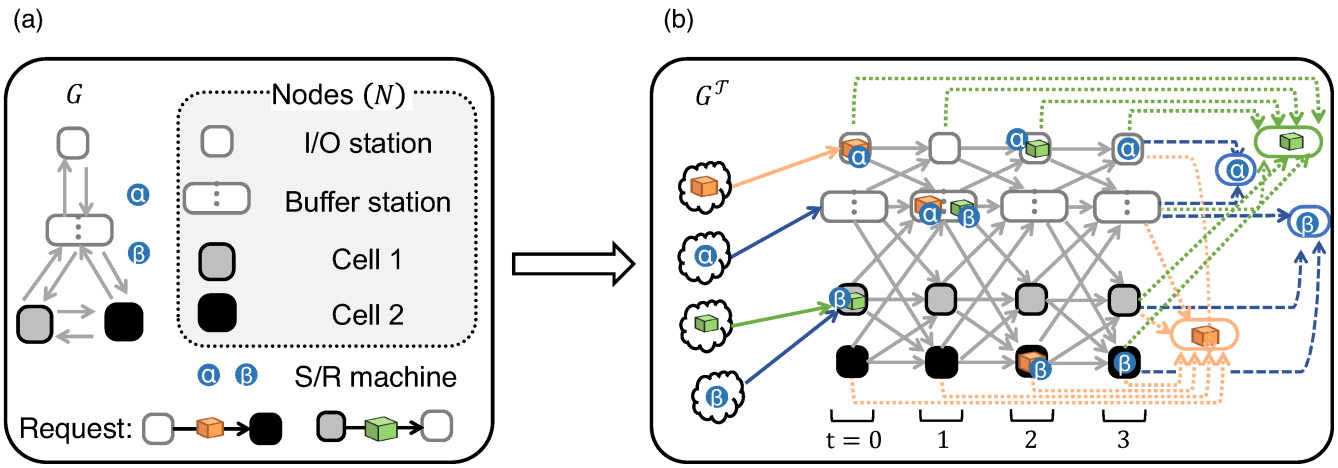

FIGURE 3.4: Example of the time-expanded network $G^{\mathcal{T}}$. Graph G in (a) is expanded by $\mathcal{T} = \{0, 1, 2, 3\}$ (see (b))

3.3.2 Formulation of the improvement of an AS/RS efficiency

Given a TEN $G^{\mathcal{T}}$, set of requests R, and set of S/R machines M, the optimization problem to make the AS/RS efficient can be formulated as follows. First, let $x_{r,a}$ be a binary variable indicating whether the flow of request $r \in R$ passes through arc $a \in A^{\mathcal{T}}$. Then, let $y_{\mu,a}$ be a binary variable for the flow of S/R machine $\mu \in M$.

$$\min_{x,y} \quad \sum_{r \in R} \left\{ \left(\sum_{t \in \mathcal{T}} t \times x_{r,(n_r(t),l'_r)} \right) + \left(1 - \sum_{t \in \mathcal{T}} x_{r,(n_r(t),l'_r)} \right) \times U - \tau_r \right\} \tag{3.1}$$

The objective function (3.1) minimizes the sum of execution times of all requests, where τ_r denotes the time when request r occurs. When the request is completed, $\sum_{t \in \mathcal{T}} x_{r,(n_r(t),l'_r)} = 1$ is obtained. Therefore, $\sum_{t \in \mathcal{T}} t \times x_{r,(n_r(t),l'_r)} - \tau_r$ is added to the objective function as the execution time of a request r. If the request is not completed, $\sum_{t \in \mathcal{T}} x_{r,(n_r(t),l'_r)} = 0$ is obtained, and $U - \tau_r$ is added to the objective function as a penalty.

Constraints are

$$\sum_{r\in R} x_{r,a} \leq 1 \qquad (^{\forall} a \in A_G^{\mathcal{T}} \cup A^L) \tag{3.2}$$

$$\sum_{\mu\in M} y_{\mu,a} \leq 1 \qquad (^{\forall} a \in A^{\mathcal{T}}) \tag{3.3}$$

$$\sum_{a\in\delta_+(n)} x_{r,a} - \sum_{a\in\delta_-(n)} x_{r,a} = \begin{cases} -1 & (n = l_r) \\ 1 & (n = l'_r) \\ 0 & (Otherwise) \end{cases} \quad (^{\forall} r \in R, {}^{\forall} n \in N^{\mathcal{T}}) \tag{3.4}$$

$$\sum_{a\in\delta_+(n)} y_{\mu,a} - \sum_{a\in\delta_-(n)} y_{\mu,a} = \begin{cases} -1 & (n = l_\mu) \\ 1 & (n = l'_\mu) \\ 0 & (Otherwise) \end{cases} \quad (^{\forall} \mu \in M, {}^{\forall} n \in N^{\mathcal{T}}) \tag{3.5}$$

$$\sum_{r\in R} x_{r,a} \leq \sum_{\mu\in M} y_{\mu,a} \qquad (^{\forall} a \in A_G^{\mathcal{T}}) \tag{3.6}$$

$$\sum_{r\in R} \sum_{a\in\delta_+(b(t))} x_{r,a} \leq w(b) \qquad (^{\forall} b(t) \in B^{\mathcal{T}}) \tag{3.7}$$

$$x_{r,a} = 0 \qquad (^{\forall} r \in R, {}^{\forall} a \in A^{\mathcal{T}} \setminus A_r^{\mathcal{T}}) \tag{3.8}$$

$$y_{\mu,a} = 0 \qquad (^{\forall} \mu \in M, {}^{\forall} a \in A^{\mathcal{T}} \setminus A_\mu^{\mathcal{T}}) \tag{3.9}$$

$$x_{r,a} \in \{0,1\} \qquad (^{\forall} r \in R, {}^{\forall} a \in A^{\mathcal{T}}) \tag{3.10}$$

$$y_{\mu,a} \in \{0,1\} \qquad (^{\forall} \mu \in M, {}^{\forall} a \in A^{\mathcal{T}}) \tag{3.11}$$

Constraints (3.2) and (3.3) represent the capacity constraints of arcs on the TEN. Constraints (3.4) and (3.5) denote flow conservation. l_r and l'_r denote the supersource and supersink of a request $r \in R$, respectively. In addition, l_μ and l'_μ denote the supersource and supersink of an S/R machine $\mu \in M$, respectively. These ensure that all flows flow over the TEN from the corresponding supersource to the supersink. Constraint (3.6) indicates that the corresponding request and S/R machine flows simultaneously move on the same arc when transporting a pallet. This condition arises from the limitation of having to use the S/R machine to transport pallets. Buffer stations (B) have limits on the number of pallets that can be placed simultaneously. We define this as the capacity $w(b)$ of a buffer station $b \in B$. Here, $B^{\mathcal{T}}$ represents the node set of the buffer stations on the TEN, and $b(t) \in B^{\mathcal{T}}$ represents the buffer station b at time t. The left-hand side of constraint (3.7) represents the number of pallets in a buffer station b at time t. Therefore, constraint (3.7) allows us to restrict the number of pallets at buffer stations to be less than or equal to $w(b)$ at any time t. Constraint (3.8) restricts the arcs through which the request flow

can pass. When a request moves through the warehouse, loading and unloading by the S/R machine occurs. Therefore, there are arcs in the TEN that cannot be passed through (Sect. 3.3.1). $A_r^{\mathcal{T}}$ represents the arcs in the TEN through which request r can pass. In addition, each S/R machine has a limit on the range in which it can operate. Let $A_\mu^{\mathcal{T}}$ be the set of arcs that the S/R machine μ can move. S/R machine $\mu \in M$ cannot go through arcs other than $A_\mu^{\mathcal{T}}$ (constraint (3.9)). Constraints (3.10) and (3.11) indicate that variables x and y are binary, respectively.

3.3.3 Creating operations for S/R machines without deadlocks

Here, we present how our scheduler controls S/R machines from the solution of the optimization problem described above. By referring to the solution to the optimization problem, we can obtain the timing required to execute each request. However, the travel times of S/R machines are rounded for discretization. Therefore, if we completely reproduce the start time of requests according to the obtained solution, the S/R machines wait needlessly until the start time of requests. This is more pronounced as the discretization interval increases.

To avoid this inefficiency, we consider the scheduler using the *order of requests to be executed* obtained from the optimization result instead of the request start time. The basic rule of the scheduler is that each S/R machine μ executes requests independently without pausing according to the request order O_μ. However, as explained in the following paragraph, this simple scheduler can deadlock S/R machines. Therefore, in Sect. 3.3.4, we introduce the technique to avoid deadlock.

Figure 3.5 shows an example of deadlock occurring using the simple scheduler. The middle node represents a buffer station whose capacity is two. This buffer station can access two S/R machines: α and β. Case (1) shows the history of pallets in and out of the buffer station according to the optimization result. The status of the buffer station changes from left to right with time. First, S/R machine β places pallet B on the buffer station and picks up pallet A. Second, S/R machine α places pallet C on the buffer station and picks up pallet B.

Based on the optimization result of case (1), the requests order in which S/R machines execute is as follows.

- O_α: (Storage C) $\rightarrow$ (Retrieval B)
- O_β: (Retrieval B) $\rightarrow$ (Storage A) $\rightarrow$ (Storage C)

Case (2) shows how a deadlock occurs in a situation in which S/R machines α and β operate independently according to their request orders O_α and O_β, respectively. In this case, S/R machine α executes Storage C before Retrieval B of S/R machine β. S/R machine β wants to execute Retrieval B but cannot perform this operation because the buffer station is already full. Furthermore, S/R machine α cannot execute Retrieval B until S/R machine β places pallet B on the buffer station, i.e., a deadlock occurs.

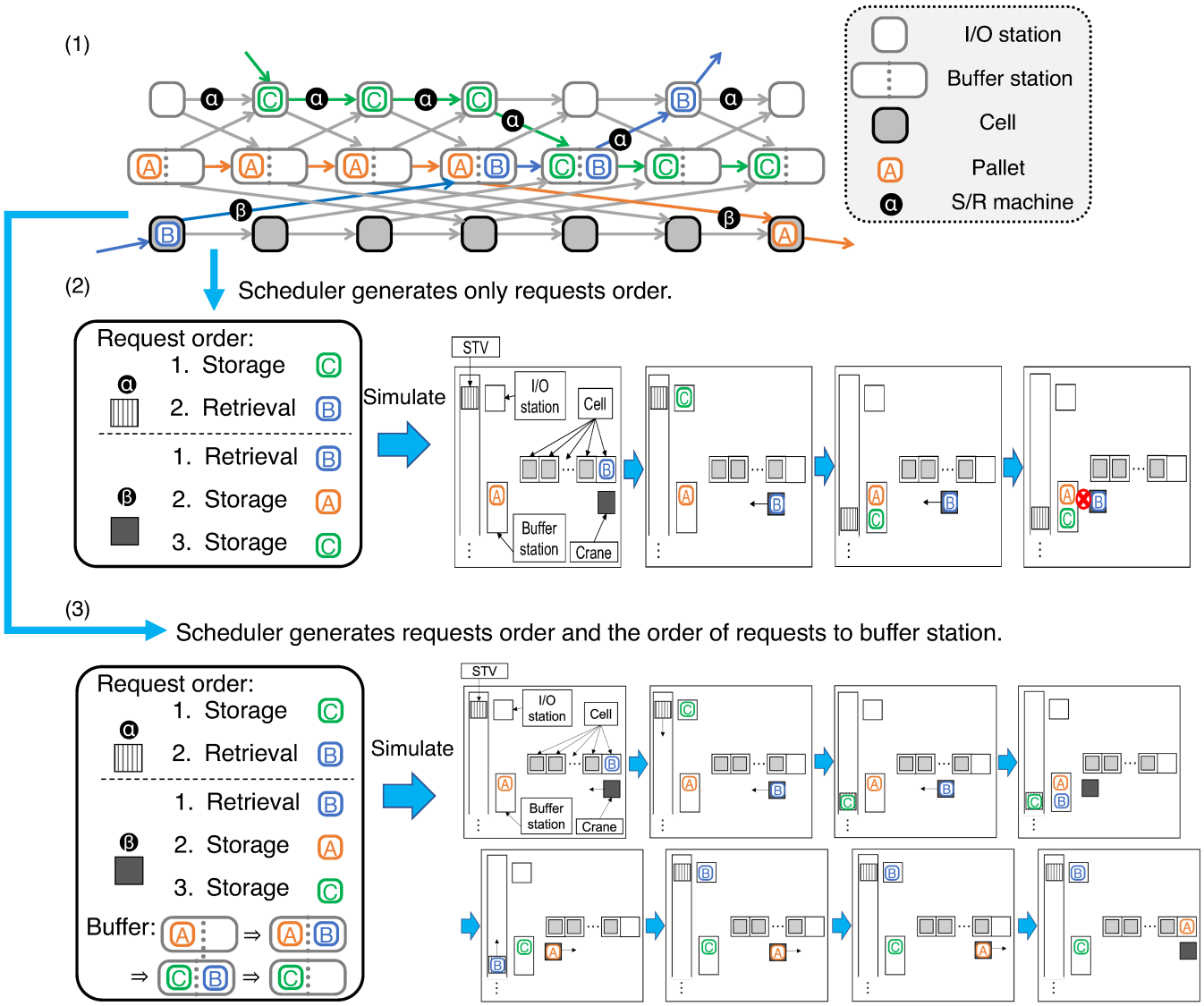

FIGURE 3.5: Case (1) is an example of processing a request based on the problem's solution formulated in Sect. 3.3.2. In this example, deadlock does not occur. Case (2) is an example in which deadlock occurs because of the order of arrival and departure at the buffer station. Case (3) is an example of avoiding deadlock by maintaining arrival and departure times at the buffer station

A manner of avoiding deadlocks as in Case (2) in Fig. 3.5 is to make S/R machines maintain the request order of Case (1) in and out of buffer stations. In this case, avoiding deadlocks is possible by preventing S/R machine α from executing Storage C until S/R machine β executes Retrieval B and Storage A, even if α can execute Storage C. In the following subsection, we generalize this observation.

3.3.4 Scheduling technique to avoiding deadlocks

We call the input and output requests for a buffer station *put* and *pick-up* requests, respectively, in this subsection. To avoid deadlocks, we should properly control the execution order of put and pick-up requests on buffer stations. However, the order of requests to buffer stations is not uniquely determined from the optimization results. For example, at a buffer station b, multiple $x_{r,a}$ are allowed to be 1 for

$a \in \delta_+(b(t))$, which represents multiple S/R machines placing pallets on b at time window t. The same applies to pick-up requests. Therefore, we can reorder these requests in the same time window. Let $r_1, \ldots, r_k$ be the put and pick-up requests in time window t for a buffer station. Owing to the physical constraints on pallets and S/R machines, the order of the requests in the same time window must satisfy the following constraints.

(i) If there exists a put request r_i and pick-up request r_j for the same pallet, then $i < j$. The placing of a pallet onto the buffer station must precede the pick-up of the same pallet.

(ii) If there exists a put request r_i and pick-up request r_j by the same S/R machine, then $i < j$. This ensures the integrity of the S/R machine's movement.

(iii) $s + \sum_{j=1}^{i} h(r_j) \leq w$ must hold for any $i \leq k$, where s is the total number of arcs whose head is the buffer station in that time window, h is a function that returns 1 if the input is a put request (else -1), and w is the capacity of the buffer station.

$s + \sum_{j=1}^{i} h(r_j)$ represents the number of pallets in the buffer station immediately after executing requests $r_1, \ldots, r_i$. Hence, the third condition requires that the number of pallets in the buffer station is always less than or equal to the capacity. In general, there are several possible orderings of requests $r_1, \ldots, r_k$ that satisfy these conditions. We describe the procedure for obtaining one of them as follows.

(1) If there are any S/R machines without put requests, we first list the pick-up requests by these S/R machines, only if there is no put request for the same pallet. Let k_1 be the number of requests listed here, and the requests order becomes $r_1, \ldots, r_{k_1}$.

(2) Next, all put requests are arranged in $r_{k_1+1}, \ldots, r_{k_1+k_2}$, where k_2 is the number of these requests.

(3) Finally, the remaining requests are arranged in $r_{k_1+k_2+1}, \ldots, r_k$.

The request order obtained by this procedure satisfies conditions (i) and (ii) because the pick-up requests associated with conditions (i) and (ii) are listed last at step (3). In addition, we can confirm that the request order satisfies condition (iii) because $s + k_2 \leq w$ holds owing to the constraints (3.7) of the optimization problem, and $s + \sum_{j=1}^{i} h(r_j) \leq s + k_2$ holds for any i from the steps (1) and (2).

We summarize the control of the S/R machine by our scheduler. We extract the request order O_μ for each S/R machine μ from the optimization results. We also determine the order of put and pick-up execution O_b for buffer station b in the above procedure. The scheduler assigns requests to each S/R machine according to the O_μ independently. If the request is a put or pick-up request to a buffer station b, the scheduler suspends the S/R machine's execution of the request, if necessary, to maintain the execution order O_b.

3.4 Numerical experiments

In this section, we show the performance of our scheduling methods by simulating the target-AS/RS. In this section, we show the performance of our scheduling methods by simulating the target-AS/RS. As described in Sect. 3.1, the experiments in this paper were designed to optimize the short-term schedule of S/R machines. The evaluation metric used was the sum of the execution times of all requests based on Rohto's demand. The execution time of one request is the time required from when the request occurs until the target pallet is transported to the requested destination. The smaller this value, the better the scheduling performance.

Section 3.4.1 describes the details of the target-AS/RS used in this experiment and experimental setup, including the method for generating requests for our experiments. We discuss the preliminary experiments conducted to determine the time granularity used in the proposed method in Sect. 3.4.2. In Sects. 3.4.3 and 3.4.4, we discuss the experiments conducted using the randomly generated request data. Section 3.4.3 used randomly generated information on all requests, and Sect. 3.4.4 conducted two types of experiments with a biased ratio of storage requests to retrieval requests. In addition, the two experiments were conducted without restrictions on the time required to solve the optimization problem. However, when our proposed method is applied to the AS/RS in continuous operation, as in the model case, the calculation time required to solve the problem is limited, as discussed in Sect. 3.3.2. Therefore, we investigated the relationship between the performance of the proposed method and its calculation time length by the experiment in Sect. 3.4.5. Based on the experiment, we discuss the real-time application of the proposed method. Appendix A.5 shows the computer environment used in the experiments.

3.4.1 Common experimental settings

This section describes the standard settings used in the following experiments. First, we describe the details of the target-AS/RS used in the experiments. Second, the request data used were generated as described. Finally, we describes the allocation method used as the baseline for the experiments.

Detailed settings of the target-AS/RS

This section describes the detailed settings for the target-AS/RS. Figure 3.6 shows a schematic of target-AS/RS. The target-AS/RS has a two-story workspace with one I/O station on each floor's left and right sides. A different STV corresponds to each I/O station. The target-AS/RS has six racks with 8 rows and 27 columns of identical cells in each rack. The capacity of a buffer station is four. That is, each buffer station can hold up to four pallets.

The crane moves horizontally at a speed of 2.67 m/s and vertically at 1.34 m/s and requires 10 s to load and unload a pallet. The movement speed of the STV is 2.67m/s, and loading/unloading one pallet requires 10 s. The S/R machine always moves at a constant speed. The crane can move horizontally and vertically independently.

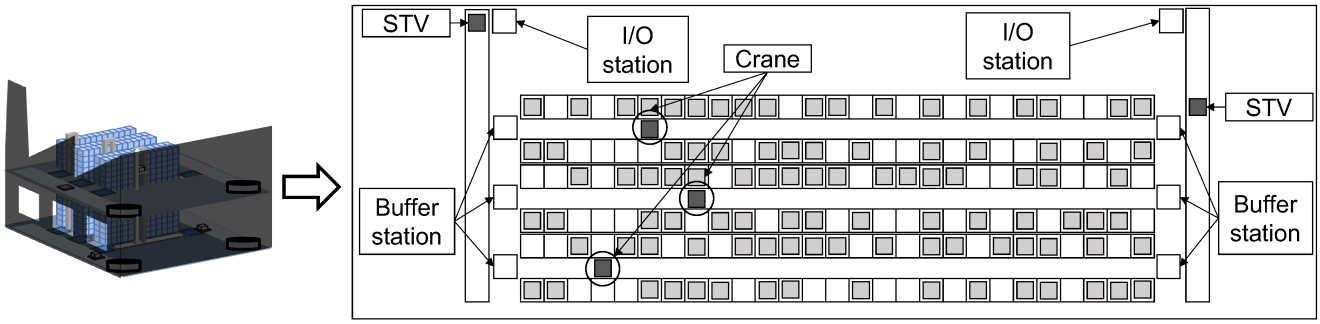

FIGURE 3.6: Overview of the target-AS/RS. We assume that the first and second floors of the target-AS/RS are the same structures

Requests data for our experiments

This section describes the method for generating request data for the experiment. Requests are generated randomly. In this case, storage requests are generated, their pallets are stored in blank cells in the AS/RS, and only pallets already stored are subject to retrieval requests. Note that the same cell is never used with the same data, and the same pallet is never retrieved and stored with the same data, as explained in Sect. 3.2.3. It is also assumed that all requests occurred simultaneously at time 0. In other words, equations ${}^{\forall}r \in R, \tau_r = 0$ hold. The following three types of data were generated for each experiment.

1. R^n_{random}: Randomly generated storage/retrieval requests. The number of requests generated is n.
2. R^n_{in}: Generate requests such that the number of storage requests accounts for 80% of n. The number of requests generated is n.
3. R^n_{out}: Generate requests such that the number of retrieval requests accounts for 80% of n. The number of requests generated is n.

Description of the baseline method used in experiments

In this section, we introduce two rule-based assignment methods that we prepared as baselines. Rule-based allocation generation algorithms have been developed for a long time. Most algorithms only consider one or two I/O stations. Among them, nearest-neighbor (NN) [14, 25] and first-come-first-served (FCFS) [20, 28] are well known. With NN, each S/R machine transports the pallet closest to its current position. The request to be executed is selected from those already generated and available for transport. FCFS searches for available requests according to the order in which they are generated, and S/R machines execute them. Even with these methods, we must also take care of S/R machines' deadlocking. Appendix A.3 describes how to avoid deadlock with the baselines.

3.4.2 Preliminary experiments for determining time granularity

This section describes preliminary experiments conducted to determine the time granularity Δ for the experiments in Sect. 3.4.3, 3.4.4, and 3.4.5. Δ should be set appropriately according to the characteristics

of the AS/RS to be applied. This reduces the error in discretizing the travel time of requests and S/R machines. In this study, two experiments were conducted to obtain an appropriate Δ.

First, we compared the execution status of requests on the TEN with that of the simulation based on the order generated by our scheduler. As the time information is discretized in the TEN, the execution status of requests deviates from it in the target-AS/RS, which can be executed continuously. However, if the discretization is too fine, the problem described in Sect. 3.3.2 becomes great, and the calculation time increases. Therefore, we ran our proposed method with different time granularity Δ, which determine the discretization width.

These experiments were conducted at R^{40}_{random}. The optimization period U was determined by the method described in Appendix A.4. The calculation time was not limited to accurately measuring the performance of the proposed method. Five runs were performed for one dataset.

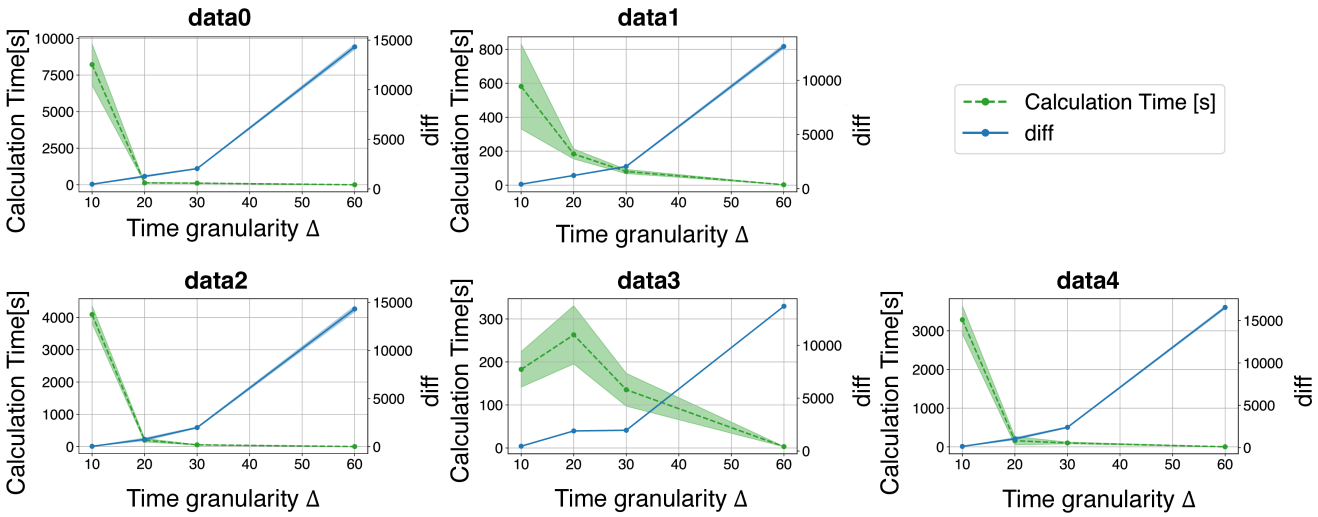

FIGURE 3.7: The relationship between time granularity, calculation time, and discretization errors. The dotted line represents the time required to solve the problem in Sect. 3.3.2 when the time granularity is changed. When changing the time granularity, the solid line is the sum of the error between actual travel time and discretized time. As the time granularity decreases, the calculation time tends to increase. However, the error becomes smaller as the time granularity is reduced

Figure 3.7 shows the results. The dotted lines represent the calculation time for each time granularity. The smaller this value is, the faster the problem formulated in Sect. 3.3.2 can be solved. In addition, a discrepancy was observed between the end time of a request recorded on the TEN and simulation. The solid line represents the sum of the absolute values of the discrepancies at the end time for each request.

The figure shows that the calculation time tends to decrease as Δ increases. This result is considered to be since the larger the time granularity, the fewer the number of discretized time points generated when splitting the problem, and thus the smaller the problem size. Furthermore, the sum of the differences in the end times measured for each request is smaller for smaller values Δ. This result implies that the smaller Δ is, the smaller the difference between the motion on the TEN and motion in the simulation can

be. Therefore, it is considered that the smaller the value Δ, the more accurately the actual movement time can be represented on the TEN. That is, the approximation can be said to be performed with minor errors.

We also compared the total execution time in the simulations for different values of Δ. In this experiment, we can see the similarity degree's effect on the total execution time. Figure 3.8 shows the violin plots of five runs for each dataset. These results show that the total execution time is best when $\Delta = 10$.

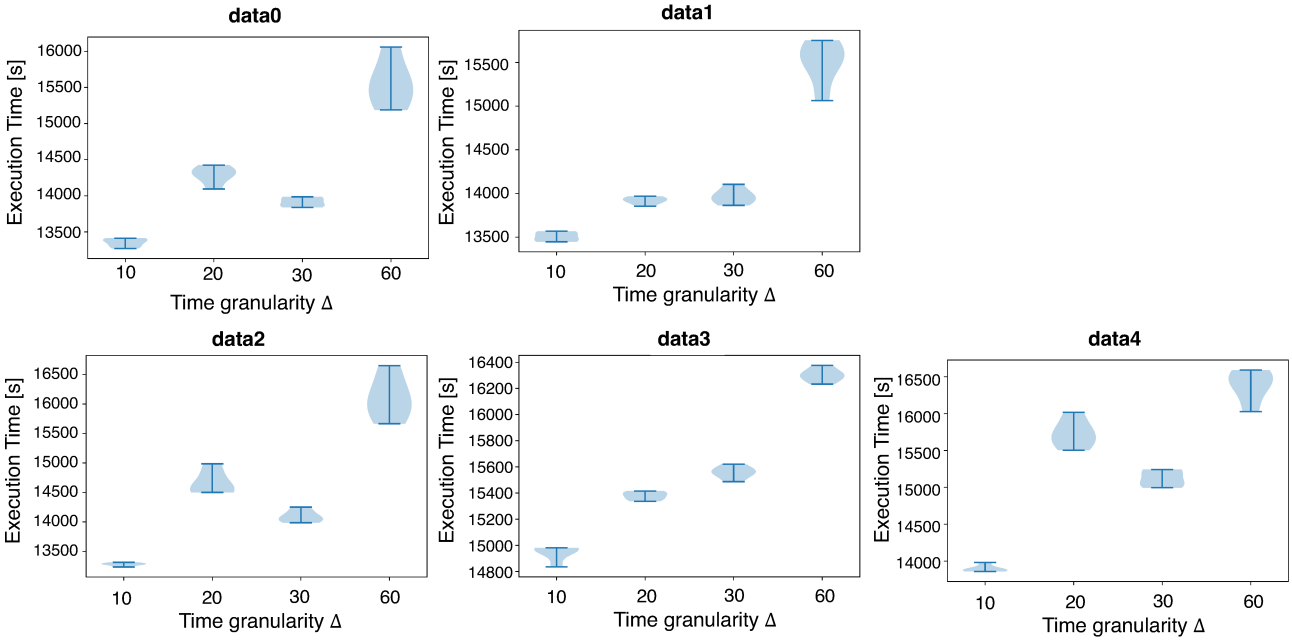

FIGURE 3.8: This figure shows the distribution of evaluation indices measured at each time granularity. The evaluation index represents the sum of the execution times of all requests. Five runs were performed for each time granularity, and the results are shown as a violin plot. The smaller the value of the evaluation index, the more efficiently the AS/RS is controlled. The smaller the time granularity, the greater the likelihood of good control

These results indicate a trade-off between good approximation and calculation time in setting of Δ. A good approximation can be said to increase the possibility of obtaining better results. Therefore, in Sects. 3.4.3, 3.4.4, and 3.4.5, we use small Δs to evaluate the potential of the proposed method, focusing on the approximation accuracy. Appendix A.4 describes the specific determination method. The optimal Δ for each instance was calculated in this manner, and $\Delta = 10$ was obtained for almost all instances. Hence, $\Delta = 10$ was used in the following experiments. In Sect. 3.4.5, we discuss the relationship between calculation time and scheduling performance by analyzing the evolution of the tentative solutions.

3.4.3 Experiments with random inputs

To evaluate the basic performance of the proposed algorithm, we conducted experiments on randomly generated data R^n_{random}. The ns used in the experiments were 10, 20, 30, and 40. We determined the number of requests based on the status of the AS/RS operation in the Rohto. Five datasets were created for each number of jobs. In this section, we describe the method and results of the experiments.

To execute the proposed method, we used R^n_{random} as input for the requests data. First, our generator and optimizer formulated and solved the problem of minimizing the sum of execution times according to Sect. 3.3. Our scheduler then generated the request order and state of buffer stations. Then, we ran the simulation using the request order generated. For the baseline, the simulation was performed based on the rules described in Sect. 3.4.1. That is, the S/R machines were simulated according to the rules, executing the available requests in order. All of these requests were assumed to occur simultaneously. The experiment was performed five times for each dataset, and the average value was calculated. Note two points when conducting this experiment. One is that the performance of the baseline method depends on the order in which the input data are listed. The other is that the time required for the optimizer to solve the problem formulated in Sect. 3.3.2 varies with the order of the requests listed in the data. This difference occurs because the solver used can produce differences in computation time depending on the order of variables, even for the same formulated problem. In this experiment, the order of the input request data affects the order of the variables. In other words, even for the same request set, there may be differences in computation time depending on the order in which the data are listed. Therefore, we randomly changed the order in which the input data were listed for each run.

Table 3.1 lists the results. *Execution Time* is the sum of the time required to execute each request and represents the average of the five runs, with the standard deviation shown in parentheses. The best method for each dataset is shown in bold, and the second-best method is underlined. *Diff* shows how much time the proposed method saves compared with the best method among the other methods. *Calculation Time* is the mean and standard deviation of the time taken to compute the optimization problem in the optimizer. The time used by our generator and our scheduler is very small compared to the time used by the optimizer (Figure 3.9). Therefore, we decided to compare the time used by the optimizer in this paper.

TABLE 3.1: Comparison of the rule-based assignment with the proposed method. Each method was run five times on one dataset, and the average of the five runs is shown. Bolded results represent the best performance results, and underlined results are the second-best performance results. The numbers in parentheses represent the standard deviation of each value. #Rqs. refers to the number of requests

		Execution Time [s]				
#Rqs.	data	FCFS	NN	Ours(Proposed)	diff	Calculation Time [s]
10	A	2,070.8(±4.87)	2,036.0(±3.29)	**1,904.8**(±12.35)	-131.2	0.30(±0.01)
	B	1,854.2(±17.51)	1,732.6(±32.33)	**1,551.2**(±9.60)	-181.4	0.50(±0.02)
	C	1,576.6(±55.36)	1,559.4(±36.44)	**1,480.8**(±2.40)	-78.6	0.31(±0.02)
	D	2,233.2(±51.30)	1,876.0(±7.35)	**1,756.2**(±0.98)	-119.8	0.66(±0.09)
	E	2,178.0(±24.79)	1,921.8(±19.69)	**1,640.4**(±1.96)	-281.4	0.34(±0.01)
20	F	5,739.2(±138.08)	4,743.2(±131.34)	**4,399.2**(±16.76)	-344.0	4.65(±0.82)
	G	5,509.0(±104.59)	4,974.2(±34.63)	**4,270.0**(±11.40)	-704.2	3.90(±0.54)
	H	6,413.8(±67.09)	5,316.2(±23.56)	**4,466.4**(±10.78)	-849.8	5.46(±0.32)
	I	6,393.0(±113.50)	4,926.4(±20.20)	**4,357.0**(±7.46)	-569.4	4.55(±0.76)
	J	6,190.0(±135.01)	4,886.6(±62.79)	**4,433.6**(±5.54)	-453.0	5.91(±0.65)
30	K	12,225.6(±210.47)	9,629.6(±56.06)	**8,844.6**(±10.50)	-785.0	306.28(±92.29)
	L	12,134.0(±138.16)	8,967.6(±139.99)	**8,563.6**(±4.63)	-404.0	160.45(±54.05)
	M	12,078.4(±101.20)	8,736.0(±84.53)	**7,916.2**(±19.45)	-819.8	284.18(±128.95)
	N	13,395.0(±247.19)	9,835.2(±136.96)	**8,711.6**(±12.14)	-1,123.6	26.00(±3.40)
	O	14,326.6(±170.25)	10,523.4(±144.19)	**9,936.6**(±24.65)	-586.8	18.68(±6.53)
40	P	20,750.0(±340.42)	14,695.0(±76.65)	**13,353.2**(±54.68)	-1,341.8	8,215.45(±1,413.75)
	Q	20,616.0(±156.47)	14,522.6(±123.26)	**13,505.0**(±49.67)	-1,017.6	581.65(±250.51)
	R	20,789.8(±248.05)	15,221.2(±270.75)	**13,279.8**(±26.03)	-1,941.4	4,090.23(±272.11)
	S	22,919.0(±593.57)	16,416.8(±121.63)	**14,944.4**(±54.75)	-1,472.4	182.71(±41.34)
	T	20,345.2(±363.01)	15,572.4(±206.49)	**13,896.6**(±43.23)	-1,675.8	3,286.66(±362.79)

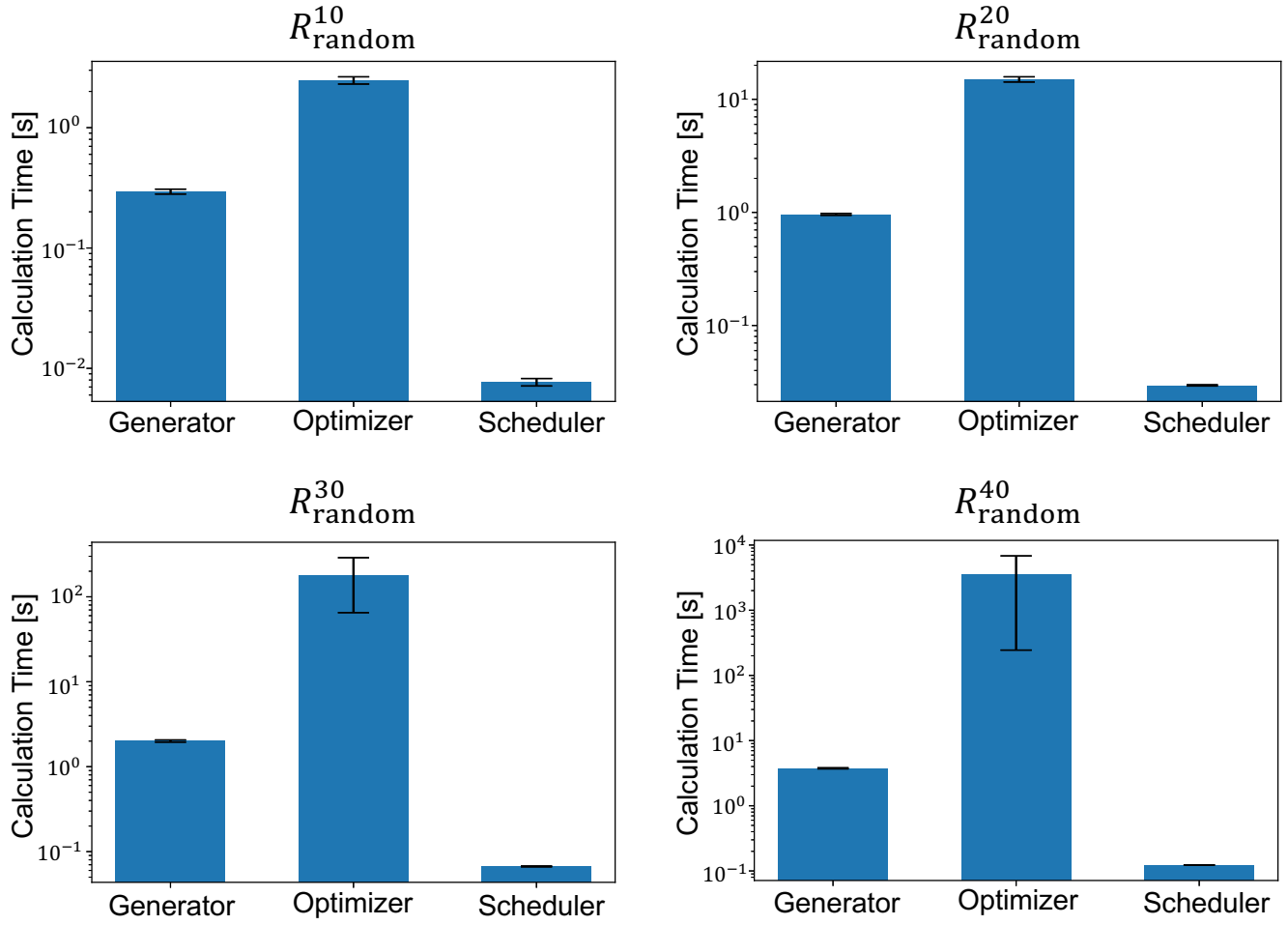

FIGURE 3.9: The computation time required by our generator, optimizer, and scheduler, respectively. The calculations were performed on five data types for each $R^n_{\text{random}}(n = 10, 20, 30, 40)$. The mean and standard deviation of the calculation time for five different data types are shown in these bar graphs. The axis representing the computation time is on the logarithmic axis.

For all problems, the proposed method has the best value. Therefore, it is effective for minimizing the total execution time. In addition, we consider that the proposed method can improve the efficiency of actual operations because many pallets can reach their destinations earlier owing to the shorter execution time.

3.4.4 Performance evaluation when requests are biased

In Sect. 3.4.3, we confirm the basic performance of our method. Rohto, which operates the AS/RS in the model case, may receive many storage/retrieval requests depending on the time of the day. For example, in the model case, trucks store many raw materials in the AS/RS in the morning. In this case, storage requests account for many of the total requests in the early hours. Conversely, the number of retrieval requests in the evening tends to increase owing to the decrease in incoming goods. Thus, depending on the time of day, storage/retrieval requests may be biased. Therefore, we compared the performance of each method for requests R^n_{in} and R^n_{out}.

The experiment was performed as in Sect. 3.4.3. Table 3.2 lists the results. The entries in the table are the same as in Table 3.1. 20 (Sto.) indicates that the number of requests is 20 and biased toward storage requests.

In all cases, the proposed method performs best. Therefore, the proposed method is expected to reduce the execution time of requests, even when storage/retrieval requests are biased, and contribute to improving the efficiency of operations.

TABLE 3.2: Comparison of the proposed method with the rule-based assignment when the data are biased. The view of this table is the same as in Table 3.1

#Rqs.	data	Execution Time [s]			diff	Calculation Time [s]
		FCFS	NN	Ours(Proposed)		
20(Sto.)	A′	5,983.0(±203.43)	5,351.4(±56.83)	**4,715.2**(±3.37)	-636.2	2.52(±0.12)
	B′	6,737.0(±128.09)	5,601.4(±102.21)	**4,873.6**(±12.16)	-727.8	2.67(±0.39)
	C′	5,633.2(±134.39)	5,110.8(±10.78)	**4,704.6**(±4.22)	-406.2	7.35(±1.64)
	D′	5,758.6(±202.33)	5,438.6(±0.49)	**4,513.4**(±8.36)	-925.2	13.33(±2.00)
	E′	5,702.8(±179.40)	4,997.8(±70.28)	**4,312.4**(±3.98)	-685.4	5.90(±1.01)
40(Sto.)	F′	20,157.4(±512.96)	17,293.4(±29.12)	**15,346.4**(±30.02)	-1,947.0	2,687.53(±456.11)
	G′	20,702.0(±413.07)	17,139.0(±191.02)	**15,203.6**(±40.44)	-1,935.4	699.02(±124.37)
	H′	22,936.4(±207.66)	19,125.0(±64.88)	**17,945.0**(±85.06)	-1,180.0	59.12(±3.45)
	K′	20,642.0(±306.55)	16,815.0(±227.05)	**15,486.4**(±11.74)	-1,328.6	1,273.52(±141.08)
	J′	21,194.2(±198.13)	17,180.2(±107.01)	**15,754.6**(±31.31)	-1,425.6	3,736.80(±521.17)
20(Ret.)	A″	6,161.4(±132.14)	5,792.0(±36.74)	**4,683.6**(±19.27)	-1,108.4	8.69(±0.31)
	B″	6,164.2(±135.77)	5,504.6(±36.24)	**4,783.8**(±7.98)	-720.8	4.10(±0.99)
	C″	6,331.8(±104.66)	5,774.6(±59.28)	**4,607.8**(±15.42)	-1,166.8	9.82(±0.72)
	D″	5,943.0(±96.51)	5,299.4(±33.80)	**4,640.0**(±6.10)	-659.4	5.60(±0.50)
	E″	5,893.6(±119.27)	5,391.6(±45.98)	**4,822.6**(±15.91)	-569.0	13.94(±3.88)
40(Ret.)	F″	19,576.2(±205.83)	16,008.4(±75.77)	**14,879.6**(±58.37)	-1,128.8	410.73(±147.34)
	G″	21,314.6(±544.15)	18,315.2(±33.46)	**15,753.8**(±13.44)	-2,561.4	1,537.86(±422.13)
	H″	19,549.8(±411.88)	15,673.8(±147.35)	**14,786.6**(±55.78)	-887.2	48.02(±7.02)
	I″	20,497.8(±469.43)	16,718.8(±147.51)	**15,613.6**(±13.81)	-1,105.2	618.59(±216.32)
	J″	21,565.4(±804.86)	19,367.4(±79.77)	**17,004.8**(±43.61)	-2,362.6	271.10(±172.34)

3.4.5 Relationship between calculation time and total execution time

The time required to compute the optimization problem increases with the number of requests. Considering the ongoing operations of AS/RSs, a large calculation time is undesirable. In addition, obtaining information on all requests executed by AS/RSs in advance is impractical. That is, calculating the request order is impossible before the AS/RSs being operating. Therefore, the proposed method is assumed to

be used in practical applications with insufficient calculation time. In this section, we discuss the impact of calculation time on the accuracy of the proposed method. While performing calculations, the total execution time is calculated when a feasible tentative solution is calculated, and the transition of the total execution time is observed.

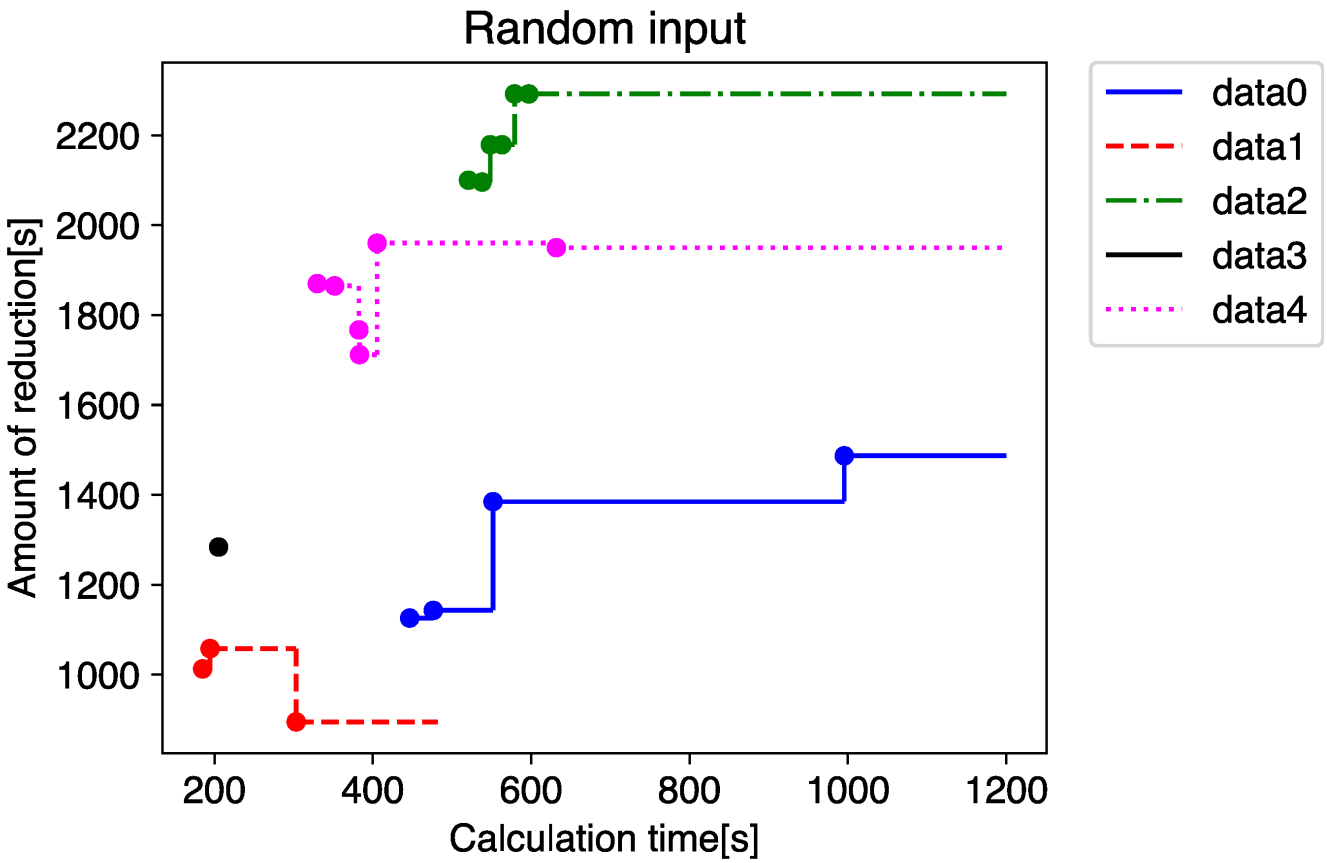

FIGURE 3.10: The figure shows the difference between the evaluation index calculated based on the output of the proposed method generated from the tentative solution and the NN evaluation index

In the experiment, we used five datasets from request set R^{40}_{random}. Figure 3.10 shows the results. This figure shows the difference between the evaluation index calculated by running our proposed scheduler using the tentative solution and NN evaluation index calculated via simulation based on the outputs generated. Each point represents the time required to obtain a tentative solution and the difference between the evaluation index calculated based on that solution and NN evaluation index. Note that because the travel time is discretized when generating the TEN, the difference is not necessarily large, even if it is better as a tentative solution owing to errors introduced at that time. However, data3 has only one point because the optimal solution was obtained first. The right figure plots the percentage of the NN evaluation index that the reductions calculated in the left figure represent.

Once a feasible provisional solution is obtained, a better request order than the best performing baseline NN can be obtained. However, if the calculation cannot be performed in time, no solution can be obtained, and no request order can be generated. Moreover, predicting the time required to obtain a

tentative solution is difficult. Therefore, in actual operation, using this method with a method that can be executed quickly, such as NN, is desirable. For example, data0 in Fig. 3.10 required 446 s to obtain the first tentative solution. Therefore, if more than 446 s can be used to compute data0, the proposed method can be used to create the request order. However, if less than 446 s are available, NN is used. This method enables us to maintain AS/RSs running at any time of day.

Chapter 4

Conclusion

This paper dealt with solving real-world problems using mathematical optimization. It focused on optimizing optical fiber routing in film-type wiring systems using mixed integer programming problems and the mobility optimization of automated warehouses using integer programming problems.

In the first half of this paper, we considered a method for routing optical fibers that employs a new routing method that uses films, using a mixed integer programming problem to reduce the number of films used as much as possible. This time, we formulated the problem as a mixed integer programming problem for 1d-LOFA, in which the wiring is done from the top end to the bottom end of the film, and the number of films to be used is minimized. Developing an algorithm that can automatically determine the wiring conditions will be necessary in the future. Each wiring condition was obtained this time by dividing the wiring conditions into cases, but it is not realistic to do this manually at every wiring stage. Also, there is a risk of overlooking necessary conditions due to the many combinations. Therefore, it is very effective in practical use to automatically check the wiring conditions based on the wiring position of the fiber and consider the appropriate conditions.

There is also a four-direction LOFA (4d-LOFA) for film-type wiring, in which the fiber inserted from the top end is routed toward all four sides of the film. This problem increases the bending of the fiber. This feature complicates the conditions under which two optical fibers are routed to the same film. The formulation of the solution given for the 4d-LOFA becomes very complicated, and it is considered difficult to find a solution. Therefore, it is necessary to develop an algorithm that can efficiently reduce the number of films used.

In the second half of the study, mobility optimization of transportation machines that transport goods to and from a factory was studied for an automated warehouse installed in a factory. MC-AS/RS, which requires multiple transporters to transport goods, is difficult to optimize because the movement of each machine affects the operational status of other machines. Therefore, we treated the mobility of objects in automated warehouses in a single optimization problem by representing the movement of transporters and requests on a TEN. Our method reduced the transport time compared to the transporter, which performs transport rule-based.

In the future, we plan to use this model in an actual AS/RS. In this study, due to various limitations

in the research, only simulation experiments were conducted using a simulator. However, requests are constantly generated in the actual AS/RS, and the transport must be operated accordingly. For example, while solving one optimization problem, a new request is generated, and the transport already assigned to carry the request continues to carry it out. It is required to accurately predict the future state of the warehouse to some extent and then to find a quick solution to apply this method in such an environment. By addressing these issues, we can expect further systemic progress toward better operation methods for automated warehouses and smart factories.

Bibliography

[1] Vincent Angilella, Matthieu Chardy, and Walid Ben-Ameur. "Cables network design optimization for the fiber to the home". In: *2016 12th International conference on the design of reliable communication networks (DRCN)*. IEEE. 2016, pp. 87–94.

[2] Andrew Bartlett et al. "An integer programming model for the Sudoku problem". In: *Journal of Online Mathematics and its Applications* 8.1 (2008), p. 1798.

[3] Tolga Bektaş and Seda Elmastaş. "Solving school bus routing problems through integer programming". In: *Journal of the operational Research Society* 58.12 (2007), pp. 1599–1604.

[4] Stefano Benati and Romeo Rizzi. "A mixed integer linear programming formulation of the optimal mean/value-at-risk portfolio problem". In: *European Journal of Operational Research* 176.1 (2007), pp. 423–434.

[5] Nils Boysen and Konrad Stephan. "A survey on single crane scheduling in automated storage/retrieval systems". In: *Eur. J. Oper. Res.* 254.3 (2016), pp. 691–704.

[6] Héctor J Carlo and Iris FA Vis. "Sequencing dynamic storage systems with multiple lifts and shuttles". In: *Int J Prod Econ* 140.2 (2012), pp. 844–853.

[7] Tzu-Li Chen et al. "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem". In: *Int J Prod Econ* 159 (2015), pp. 158–167.

[8] Juan De Vicente, Juan Lanchares, and Román Hermida. "Placement by thermodynamic simulated annealing". In: *Physics Letters A* 317.5-6 (2003), pp. 415–423.

[9] Randall J Diaz, Kent Shum, and Dennis G Vaillancourt. *Systems, apparatuses, and methods for managing cables with minimum bend radius*. US Patent 7,027,706. 2006.

[10] Lester R Ford Jr and Delbert Ray Fulkerson. "Constructing maximal dynamic flows from static flows". In: *Oper Res* 6.3 (1958), pp. 419–433.

[11] Amir Hossein Gharehgozli et al. "Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system". In: *Transp. Sci.* 51.1 (2017), pp. 19–33.

[12] Reginaldo Guirardello and Ross E Swaney. "Optimization of process plant layout with pipe routing". In: *Computers & chemical engineering* 30.1 (2005), pp. 99–114.

[13] Serkan Gunpinar and Grisselle Centeno. "An integer programming approach to the bloodmobile routing problem". In: *Transportation research part E: logistics and transportation review* 86 (2016), pp. 94–115.

[14] Min-Hong Han et al. "On sequencing retrievals in an automated storage/retrieval system". In: *IIE Trans* 19.1 (1987), pp. 56–66.

[15] Ya-Hong Hu et al. "Travel time analysis of a new automated storage and retrieval system". In: *Comput Oper Res* 32.6 (2005), pp. 1515–1544.

[16] Neal Anthony Janus, Anthony Pellegrino, and Randy Alan Reagan. *Fiber optic interconnection combination closure*. US Patent 6,385,381. 2002.

[17] Akinori Kanasugi. "Algorithms for Placement and Routing Problems (In Japanese)". In: *Journal of Japan Institute of Electronics Packaging* 2.3 (1999), pp. 184–187.

[18] Akinori Kanasugi and Naoshi Nakaya. "A Single-Layer Routing Method based on Maze Router and Genetic Algorithm (In Japanese)". In: *IPSJ SIG technical reports* 1999.12 (1998-SLDM-091) (1999), pp. 97–103.

[19] Ahmet B Keha, Ketan Khowala, and John W Fowler. "Mixed integer programming formulations for single machine scheduling problems". In: *Computers & Industrial Engineering* 56.1 (2009), pp. 357–367.

[20] Heungsoon Felix Lee and Samantha K Schaefer. "Sequencing methods for automated storage and retrieval systems with dedicated storage". In: *Comput Ind Eng* 32.2 (1997), pp. 351–362.

[21] Leandro Magatão, Lucia VR Arruda, and Flávio Neves Jr. "A mixed integer programming approach for scheduling commodities in a pipeline". In: *Computers & chemical engineering* 28.1-2 (2004), pp. 171–185.

[22] Inês Marques, M Eugénia Captivo, and Margarida Vaz Pato. "An integer programming approach to elective surgery scheduling: Analysis and comparison based on a real case". In: *OR spectrum* 34.2 (2012), pp. 407–427.

[23] Salman Mashayekh et al. "A mixed integer linear programming approach for optimal DER portfolio, sizing, and placement in multi-energy microgrids". In: *Applied Energy* 187 (2017), pp. 154–168.

[24] Kees Jan Roodbergen and Iris F.A. Vis. "A survey of literature on automated storage and retrieval systems". In: *Eur. J. Oper. Res.* 194.2 (2009), pp. 343–362. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2008.01.038. URL: https://www.sciencedirect.com/science/article/pii/S0377221708001598.

[25] Bhaba R Sarker et al. "Performance evaluation of a double shuttle automated storage and retrieval system". In: *Prod. Plan. Control.* 2.3 (1991), pp. 207–213.

[26] Haiteng Sui and Wentie Niu. "Branch-pipe-routing approach for ships using improved genetic algorithm". In: *Frontiers of Mechanical Engineering* 11.3 (2016), pp. 316–323.

[27] Shunji Tanaka and Mituhiko Araki. "Routing problem under the shared storage policy for unit-load automated storage and retrieval systems with separate input and output points". In: *Int J Prod Econ* 47.9 (2009), pp. 2391–2408.

[28] Jeroen P Van Den Berg and AJRM Gademann. "Optimal routing in an automated storage/retrieval system with dedicated storage". In: *IIE Trans* 31.5 (1999), pp. 407–415.

[29] Bo Xing et al. "Ant colony optimization for automated storage and retrieval system". In: *IEEE Congress on Evolutionary Computation*. IEEE. 2010, pp. 1–7.

[30] Akihiro Yoshida et al. "Practical end-to-end repositioning algorithm for managing bike-sharing system". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 1251–1258.

Appendix A

Appendix

A.1 Constraints on optical fiber pair routing

This section discusses the limitations of routing two fibers on the same film; for a single pair of fibers, there are limitations derived from design requirements (b) and (e). Design requirement (b) is the restriction that the fibers must not intersect at bends. Design requirement (e) is the restriction that two fibers wired in parallel must be spaced at least a certain distance apart. Both of these restrictions must be observed.

A.1.1 Modeling of wiring paths

To consider wiring conditions, model the wiring path. The routing path combines portions parallel to the x-axis, portions parallel to the y-axis, and bent portions, depending on the routing restrictions set in Sect. 2.3.1. Since the fiber follows design requirement (a), the bent portion draws a circular arc with a radius R. Let p_f be the routing path of a certain fiber f, and h be the height of the bend. It can be expressed as follows using the mediating variable t.

$$p_f(t) = \begin{cases} \left(x_0^f,\ y_0^f + 5t\left(h - R\sin\theta_f\right)\right) & \left(0 \le t < \frac{1}{5}\right) \\ \left(x_0^f + R(1-\cos\theta(t))\mathrm{sgn}_f,\ h - R\sin\theta_f + R\sin\theta(t)\right) & \left(\frac{1}{5} \le t < \frac{2}{5}\right) \\ \left(\frac{x_0^f + x_1^f}{2} + 5\left(t - \frac{1}{2}\right)\mathrm{sgn}_f \cdot \max\left\{0,\ \left|x_1^f - x_0^f\right| - 2R\right\},\ h\right) & \left(\frac{2}{5} \le t < \frac{3}{5}\right) \\ \left(x_1^f - R\left(1-\cos\theta'(t)\right)\mathrm{sgn}_f,\ h + R\sin\theta_f - R\sin\theta'(t)\right) & \left(\frac{3}{5} \le t < \frac{4}{5}\right) \\ \left(x_1^f,\ h + R\sin\theta_f + 5\left(t - \frac{4}{5}\right)\{H - h + R\sin\theta_f\}\right) & \left(\frac{4}{5} \le t \le 1\right) \end{cases} \tag{A.1}$$

We defined sgn_f as follows;

$$\mathrm{sgn}_f = \begin{cases} -1 & \left(x_1^f - x_0^f < 0\right) \\ 0 & \left(x_1^f - x_0^f = 0\right) \\ 1 & \left(x_1^f - x_0^f > 0\right) \end{cases}.$$

Additionally, $\theta_f \in \left[0, \frac{\pi}{2}\right]$ represents the optical fiber's bend angle and is defined as follows;

$$\theta_f := \arccos\left\{1 - \frac{1}{R}\min\left(\frac{\left|x_1^f - x_0^f\right|}{2}, R\right)\right\}.$$

Using this, $\theta_f(t), \theta'_f(t)$ is defined as follows;

$$\begin{aligned} \theta_f(t) &= 5\theta_f\left(t - \frac{1}{5}\right) && \left(\frac{1}{5} \leq t < \frac{2}{5}\right), \\ \theta'_f(t) &= 5\theta_f\left(\frac{4}{5} - t\right) && \left(\frac{3}{5} \leq t < \frac{4}{5}\right). \end{aligned}$$

A.1.2 Constraints on optical fiber pairs

Now, we consider two different fibers $f_1, f_2 \in F$. The condition for routing the fibers so they do not intersect at the bend depends on the position of the x coordinates of the top and bottom ends of f_1. For example, let x coordinates of the top end of the optical fiber $f \in F$ be x_0^f and x coordinates of the bottom end be x_1^f. Then, to omit the same bending combination from the symmetry of the optical fiber, let $x_0^{f_1} \leq x_1^{f_1}$.

The overlap of the other fiber at each bend determines whether f_1 and f_2 satisfy design requirement (b). Optical fibers should be routed to be parallel to the x- and y-axes except at the bends. Therefore, design requirement (b) is due to whether, for a given fiber f_1, f_2 passes inside or outside the bend of f_1. Furthermore, the bending height depends on the space between f_1 and f_2. The constraints in design requirement (b) for the fiber f_1, f_2 should be considered with these things in mind.

Here, the shape of the fiber routing path is partitioned to obtain the condition that no other fiber intersects the bend. Depending on the x coordinates x_0^f, x_1^f of the upper and lower ends of f, the shape of the routing path differs significantly (see Fig. 2.3). At $\left|x_1^f - x_0^f\right| > 2R$, $\frac{2}{5} \leq t < \frac{3}{5}$, the fiber is stretched parallel to the x axis. This means that they can intersect other fibers in this section. At $0 < \left|x_1^f - x_0^f\right| \leq 2R$, the section at $\frac{2}{5} \leq t < \frac{3}{5}$ is a single point. Therefore, it cannot intersect with other fibers at the section of $\frac{2}{5} \leq t < \frac{3}{5}$. At $\left|x_1^f - x_0^f\right| = 0$, the fiber does not bend. Therefore, there is no bent section in this fiber. Additionally, at $R < \left|x_1^f - x_0^f\right| \leq 2R$, the bending section of fiber f, $\frac{1}{5} \leq t < \frac{4}{5}$, completely intersects the bending section of the other fiber. From these facts, the constraint condition is considered for four patterns in $x_1^{f_1} - x_0^{f_1} > 2R$ (Fig. A.1 (1)), $R < x_1^{f_1} - x_0^{f_1} \leq 2R$ (Fig. A.1 (2)), $0 < x_1^{f_1} - x_0^{f_1} \leq R$ (Fig. A.1 (3)), $x_1^{f_1} - x_0^{f_1} = 0$ (Fig. A.1 (4)), paying attention to $x_0^{f_1} \leq x_1^{f_1}$.

Here, we consider the wiring condition for fiber f_1 where $x_1^{f_1} - x_0^{f_1} > 2d$. What is necessary to observe the design requirement (b) is to wire the bends so that they do not overlap each other. For example, if both ends of f_2 are to the left of $x_0^{f_1} - g_{f_2}$ (bottom left-most area in Fig. A.1 (1)), the bent portions of f_1 and

f_2 will not overlap at any bend height. Therefore, for these two fibers, f_1 and f_2, there are no restrictions on wiring; if the top end of f_2 is to the left of $x_0^{f_1} - g_{f_2}$ and the bottom end is between $x_0^{f_1}$ and $x_0^{f_1} + g_{f_1}$, they cannot be wired to the same film unless $y_{f_2} \geq y_{f_1}$. Under other conditions, the bent portion of f_1 and the bent portion of f_2 would overlap with the other.

Considering the wiring conditions for two fibers in a certain positional relationship, it is also possible to consider the wiring conditions for fibers in other positional relationships, where the positional relationship is rotated 180 degrees for the center point of the film. For example, consider the third condition from the bottom of the first left column in Fig. A.1 (1) ($x_0^{f_2} \leq x_0^{f_1} - g_{f_2}$ and $x_0^{f_1} \leq x_1^{f_2} \leq x_0^{f_1} + g_{f_1}$, in Fig. 2.6 ($b_3$)), rotated 180 degrees. The top, bottom, left, and right sides are interchanged in this operation, so the vertical and horizontal magnitude relationships are interchanged. In other words, the inequality sign relationship is reversed, and the plus and minus signs are also swapped. In addition, since the top and bottom edges are also swapped, x_0^f and x_1^f are swapped. In other words, the inverted state of "$x_0^{f_2} \leq x_0^{f_1} - g_{f_2}$ and $x_0^{f_1} \leq x_1^{f_2} \leq x_0^{f_1} + g_{f_1}$" becomes "$x_1^{f_1} - g_{f_1} \leq x_0^{f_2} \leq x_1^{f_1}$ and $x_1^{f_2} \geq x_1^{f_1} + g_{f_2}$". The given constraint condition is also reversed from $y_{f_2} \geq y_{f_1}$(in Fig. 2.6 b_3) to $y_{f_2} \leq y_{f_1}$(in Fig. 2.6 b_6). In this way, once the routing conditions for a specific fiber pair are determined, the routing conditions for its reversed state can also be obtained. However, it should be noted that the wiring conditions do not change even if the inversion is made when wiring is not possible (b_0) and when there are no constraints at the time of wiring (b_7).

The constraints for each pattern are shown in Fig. A.1. The $g(f) = R(1 - \cos\theta_f)$ represents the length of the bend on the x-axis. The dotted lines in Fig. A.1 are the boundaries where the conditions change. The most stringent constraint among adjacent constraints is adopted when a fiber pair falls on a line segment or at the intersection of dotted lines. For example, consider the upper left intersection in Fig. A.1. Here, the regions b_5, b_4, b_5, b_4, and b_0 are adjacent. The most stringent constraint among these is b_0. Therefore, pairs of fibers in this point condition cannot be routed to the same film. Additionally, note that no fibers share the same coordinates at their endpoints. In other words, no two or more fibers are inserted from the same location, and no fibers are put out at the same location.

The constraints in design requirement (e) are also considered in fiber f_1, f_2 as in (b). Again, $x_0^{f_1} \leq x_1^{f_1}$ is assumed due to the symmetry of the wiring. According to the condition in design requirement (e), fibers must be spaced at least a certain distance apart when wiring. Additionally, the wiring conditions are considered for three patterns where the endpoint of f_1 is $x_1^{f_1} - x_0^{f_1} > 2d$, $0 < x_1^{f_1} - x_0^{f_1} \leq 2d$, $x_1^{f_1} - x_0^{f_1} = 0$. Fig. A.2 shows the constraint conditions for each pattern. As in (b), the condition on the boundary line adopts the most severe constraint among the adjacent conditions.

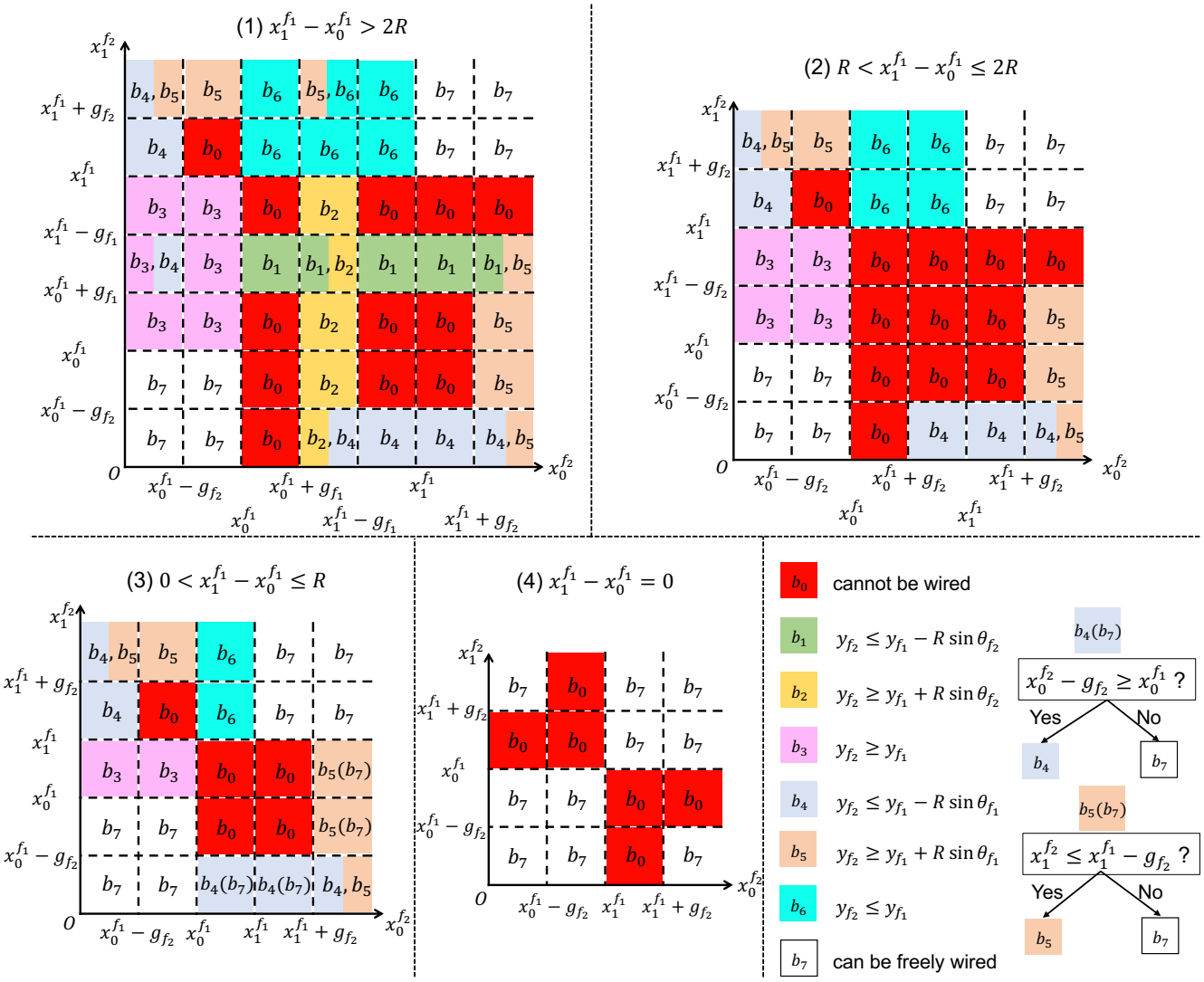

FIGURE A.1: Constraints in design requirement (b)

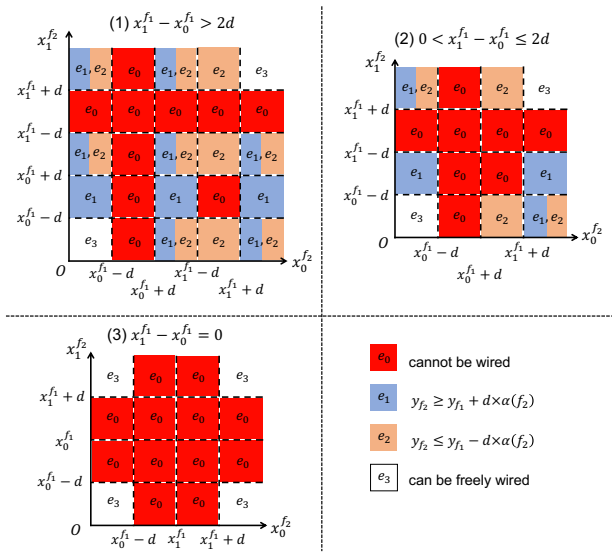

FIGURE A.2: Constraints in design requirement (e)

A.2 Time-expanded network generation details

In this section, we describe the details of the method for generating the TEN used in the proposed method (Sect. 3.3.1).

From the information in an AS/RS, S/R machines, and requests, we generate a graph $G = (N, A)$ that represents the locations involved in node movements and movement between nodes as directed arcs. G is generated according to Algorithm 2. Let $N_r = \{n_0, n_1, \ldots, n_{m_r}\} \subset N$ denote the source, via points, and destination of request r. In addition, we define s_μ as the node representing the initial point of S/R machine μ.

TEN $G^{\mathcal{T}} = (N^{\mathcal{T}}, A^{\mathcal{T}})$ is generated according to Algorithm 3. Let n_{sour_r} be the initial position of request r and s_μ the initial position of S/R machine μ. Furthermore, the destination of request r is denoted as n_{dest_r}. Section 3.3.1 defines the supersource and supersink of request r and S/R machine μ.

Algorithm 2 Generating graph G

Require: R_{sto}: A set of storage requests, R_{ret}: A set of retrieval requests, M: A set of S/R machines, N_r: A set of nodes of request $r \in R_{\mathrm{sto}} \cup R_{\mathrm{ret}}$

$N = \emptyset, A = \emptyset$
for $r \in R_{\mathrm{sto}} \cup R_{\mathrm{ret}}$ **do**
    $N \leftarrow N \cup N_r$ // Add the point through which request r passes to node set;
end for
for $\mu \in M$ **do**
    $N \leftarrow N \cup \{s_\mu\}$. // Add initial position s_μ of S/R machine μ to node set;
end for
for $(n, m) \in N \times N$ **do**
    if $n \neq m$ **then**
        if There are some S/R machines that can move between n, m **then**
            $A \leftarrow A \cup \{(n, m), (m, n)\}$. // Add arcs (n, m) and (m, n) to arc set;
        end if
    end if
end for

To model arcs, A in a graph G using $\mathcal{T}$, define $\dot{t}_{i,j} = \max(1, [t_{i,j}/\Delta]) \in \mathbb{N}$ as the travel time of an arc $(i, j) \in A$ on the graph, where $t_{a,b}$ is the time required to move from node a to node b. For example, define a moving arc $(i(t), j(t + \dot{t}_{i,j}))$ for all $t \in \mathcal{T}$ satisfying $t + \dot{t}_{i,j} \leq \max \mathcal{T}$. The set of staying arcs $H^{\mathcal{T}}$ is defined as $(n(t), n(t + 1))$ for each node $n \in N$ and period $t \in [0, \max \mathcal{T} - 1]$. A^L is the set of arcs flowing from the supersource to the corresponding node and from the corresponding node to the supersink. Arcs of A^L are defined for all S/R machines and requests.

The arcs leaving supersource $l_\mu \in L$ for S/R machine $\mu \in M$ are denoted by $(l_\mu, s_\mu(0)) \in A^L$ using node s_μ at the initial position. The arcs entering supersink $l'_\mu \in L$ are denoted by $(n(\max \mathcal{T}), l'_\mu)$ for any node $^\forall n \in N$.

The arc leaving the supersource $l_r \in L$ of request $r \in R$ is denoted by $(l_r, n_{\mathrm{sour}_r}(\tau_r)) \in A^L$, using node n_{sour_r} at the initial position n_{sour_r} of request r at time τ_r. τ_r is the time when request r occured. The arc entering supersink $l'_r \in L$ is denoted by $^\forall t \in \mathcal{T}, (n_{\mathrm{dest}_r}(t), l'_r) \in A^L$ for the destination $n_{\mathrm{dest}_r} \in N$

of r. Request r passes through this arc, indicating that the transfer is complete. We also add $^{\forall}n \in N \setminus \{n_{\mathrm{dest}_r}\}, (n(\max \mathcal{T}), l_r^{'}) \in A^L$ to allow for the case in which the conveyance is not completed even at the end time $\max \mathcal{T}$.

Algorithm 3 Generating a time-expanded network

Require: $G = (N, A)$: Graph G, $\mathcal{T}$: Set of time periods, R: Set of requests, M: Set of S/R machines, τ_r: Time when request $r \in R$ occurred, l_r: Request $r \in R$ supersource, l_r': Request $r \in R$ supersink, l_μ: S/R machine $\mu \in M$ supersource, l_μ': S/R machine $\mu \in M$ supersink

$N^{\mathcal{T}} = \emptyset, A^{\mathcal{T}} = \emptyset$
for $t \in \mathcal{T}$ **do**
  for $a = (n, m) \in A$ **do**
    // Expand graph G by $\mathcal{T}$ without loads;
    if $t + t_{n,m} \leq \max \mathcal{T}$ **then**
      $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(n(t), m(t + t_{n,m}))\}$
      $N^{\mathcal{T}} \leftarrow N^{\mathcal{T}} \cup \{n(t), m(t + t_{n,m})\}$
    end if
    // Expand graph G by $\mathcal{T}$ with loads;
    if $t + t'_{n,m} \leq \max \mathcal{T}$ **then**
      $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(n(t), m(t + t'_{n,m}))\}$
      $N^{\mathcal{T}} \leftarrow N^{\mathcal{T}} \cup \{m(t + t'_{n,m})\}$
    end if
    // Add staying arc;
    if $t + 1 \leq \max \mathcal{T}$ **then**
      $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(n(t), n(t + 1))\}$
      $N^{\mathcal{T}} \leftarrow N^{\mathcal{T}} \cup \{n(t), n(t + 1)\}$
    end if
  end for
end for
for $r \in R$ **do**
  // Add arcs regarding the supernode of request r;
  $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(l_r, n_{\mathrm{sour}_r}(\tau_r))\}$
  for $n \in N \setminus \{n_{\mathrm{dest}_r}\}$ **do**
    $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(n(\max \mathcal{T}), l_r^{'})\}$
  end for
  for $t \in \mathcal{T}$ **do**
    $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(n_{\mathrm{dest}_r}(t), l_r^{'})\}$
  end for
  $N^{\mathcal{T}} \leftarrow N^{\mathcal{T}} \cup \{l_r, l_r^{'}\}$
end for
for $\mu \in M$ **do**
  // Add arcs regarding the supernode of S/R machine μ;
  $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(l_\mu, s_\mu(0))\}$
  for $n \in N$ **do**
    $A^{\mathcal{T}} \leftarrow A^{\mathcal{T}} \cup \{(n(\max \mathcal{T}), l_\mu^{'})\}$
  end for
end for

A.3 Method for avoiding deadlocks in baselines

This section describes methods for avoiding deadlocks in NN and FCFS, which are used as baselines. Unlike the proposed method, these methods are executed immediately. Therefore, the deadlock avoidance

methods should also execute immediately. A deadlock is caused by a pallet unready for transportation. Hence, we use a method that guarantees the pallet to be reliably moved and that can execute immediately.

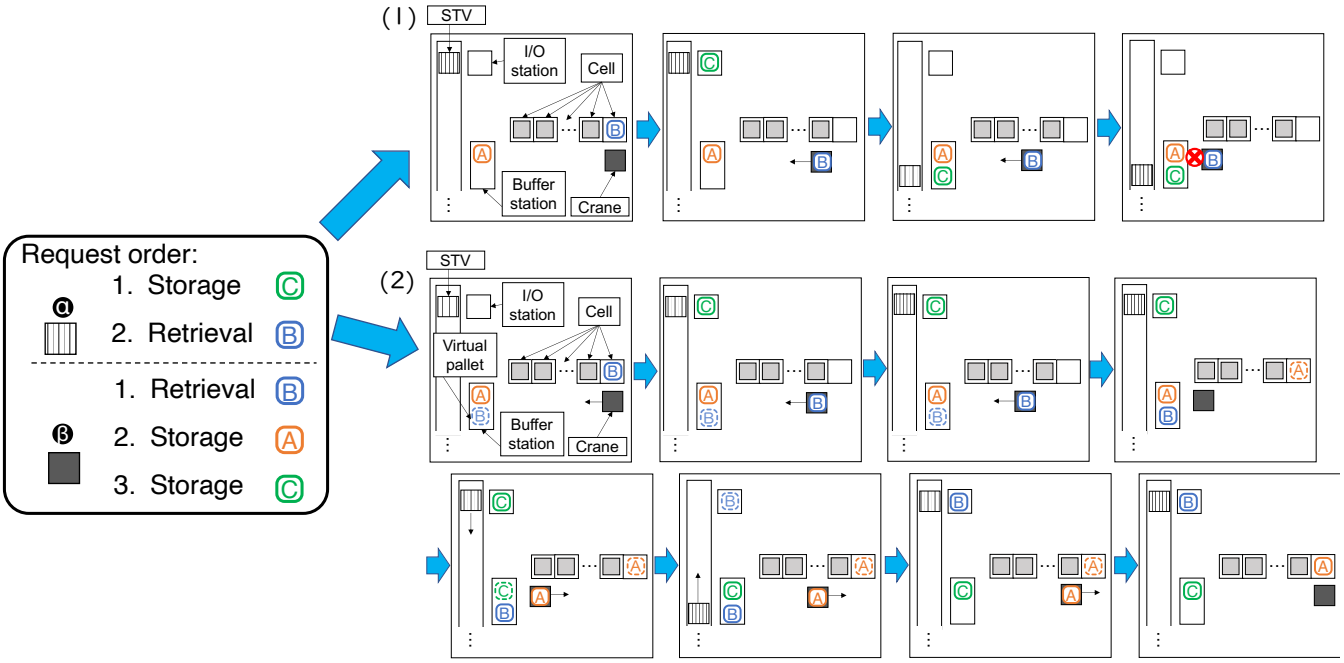

FIGURE A.3: Example of using the virtual pallet to avoid deadlocks when operating an AS/RS by NN or FCFS

For example, Fig. A.3 (1) shows an example of a deadlock caused by S/R machines α and β being assigned requests independently. In this case, β, initially executing the request, must unload pallet B to the buffer station first and then retrieve pallet A from the buffer station. However, because α can arrive at the buffer station before β, pallet C carried by α enters the buffer station first. Therefore, β, who arrives later, cannot unload B, resulting in a deadlock. To avoid this situation, we place a *virtual pallet* at the next destination of the request when the request begins execution. Then, by calculating the capacity of the buffer station, including the virtual pallet, the request is guaranteed to be reliably placed at the next destination. Therefore, the pallet can be reliably moved.

A.4 Hyperparameter determination method

To implement the proposed method, the optimization period U and time granularity Δ required for formulation must be set in advance. This section describes how we determine the parameters used in our experiments.

Setting these values by the actual conditions of the target AS/RS is desirable. However, in this experiment, determining these values appropriately from the AS/RS information used in the model case was difficult. Therefore, the following method was used.

U is obtained as follows. First, let t_{max} by the time required for the most time-consuming transfer among the transfers in the target-AS/RS. Then, for the request set R to be optimized, we set $U = \#R \times t_{max}$.

This method makes it possible to handle the requests in R as much as possible until the completion of the transfer in a single optimization problem.

Δ was determined as follows. First, a graph $G = (N, A)$ was generated according to Algorithm 2. Next, the travel times of all arcs in A were calculated. The lower bound of Δ was set to l. If the time is required to move an arc a is denoted as t_a, the discretized time is denoted as $\max(1, [t_a/d])$ using time granularity d. Using t_a, Δ was obtained by the following equation:

$$\Delta = \underset{l \leq d \leq 2l}{\arg\min} \sum_{a \in A} \|t_a - \max(1, [t_a/d])\| \tag{A.2}$$

The lower limit of l is set to $l = 10$ based on the preliminary experiment results (Sect. 3.4.2).

A.5 Computational environment

In the computational environment using our experiments, the CPU was an Intel (R) Core (TM) i9-7940X with a 3.10GHz CPU frequency, and the memory was 128GB. We used Gurobi Optimizer v9.5.1 to solve the optimization problem to make AS/RS operations efficient, and 10 threads were used to solve the problem.

List of Figures

List of Tables