

**Hierarchical Optimization for
Manufacturing and Logistics Systems:
Integrating Simulation, Surrogates, and
Mathematical Programming**

Issei Suemitsu

2026

Kyoto University

**Hierarchical Optimization for
Manufacturing and Logistics Systems:
Integrating Simulation, Surrogates, and
Mathematical Programming**

Issei Suemitsu

A dissertation submitted in partial fulfillment of
the requirements for the degree of Doctor of Engineering

March 2026

Contents

1	General Introduction	1
1.1	Background	1
1.2	Strategic Decision Making	2
1.3	Tactical Decision Making	2
1.4	Operational Decision Making	3
1.5	Research Objectives and Contributions	3
2	Simultaneous Optimization of Layout and Task Schedule for Robotic Cellular Manufacturing Systems	6
2.1	Introduction	6
2.2	Multi-robot Cellular Manufacturing System	7
2.3	Design Variables	8
2.3.1	Assumptions for RCMS Layout Design Problem	8
2.3.2	Encoding Technique for Task Allocation	8
2.3.3	Encoding Technique for Component Positioning	10
2.3.4	Design Variables for RCSM Layout Design Problem	13
2.4	Design Criteria for Layout Design	13
2.4.1	Layout Design Problem	13
2.4.2	Operation Time	14
2.4.3	Manipulability	15
2.4.4	Layout Area	16
2.4.5	Optimization Problem	17
2.4.6	Constraints for Collision Avoidance	17
2.5	Optimization Technique	18
2.5.1	Multiobjective Genetic Algorithm	18
2.5.2	Crossover	18
2.5.3	Mutation	18

2.5.4	Selection	19
2.6	Example Problem	19
2.6.1	Problem Settings	19
2.6.2	Two-objective Optimization Problem	22
2.6.3	Parameter Calibration for Dummy Components	23
2.6.4	Three-objective Optimization Problem	26
2.7	Conclusion	29
3	Surrogate-assisted Scenario-generation Method for Simulation-based Stochastic Programming Problems	30
3.1	Introduction	30
3.2	Background	31
3.2.1	Stochastic Programming	31
3.2.2	Simulation-based Optimization	32
3.2.3	Simulation-based Stochastic Programming Problem	32
3.3	Scenario Generation	33
3.3.1	Problem Statement and Method Classification	33
3.3.2	Distribution-driven Methods	34
3.3.3	Problem-driven Methods	34
3.3.4	Cost-space Scenario Clustering Method	36
3.4	Inferred Cost-space Scenario Clustering	37
3.4.1	Overview of Proposed Method	37
3.4.2	Mathematical Development	38
3.4.3	Pre-training Phase	40
3.4.4	Scenario-generation Phase	41
3.5	Portfolio Optimization Problem	43
3.5.1	Problem Statement	43
3.5.2	Experiment Settings	43
3.5.3	Surrogate Model Selection	44
3.5.4	Optimization Performance Comparison	45
3.6	Stochastic Server Location Problem	51
3.6.1	Problem Statement and Experiment Settings	51
3.6.2	Optimization Performance Comparison	52
3.7	Inventory Placement Optimization Problem	52
3.7.1	Problem Statement	52

3.7.2	Experiment Settings	54
3.7.3	Surrogate Model Selection	55
3.7.4	Optimization Performance Comparison	56
3.8	Conclusion	60
4	Integration of Sales, Inventory, and Transportation Resource Planning by Dynamic-demand Joint Replenishment Problem with Time-varying Costs	61
4.1	Introduction	61
4.2	Related Works	63
4.2.1	Joint Replenishment Problem	63
4.2.2	Simulation-based Optimization Approaches	64
4.2.3	Hybrid Simulation and Optimization Approaches	65
4.3	Problem Description and Assumptions	66
4.3.1	Inventory Policy	66
4.3.2	Description of Joint Replenishment Problem	68
4.3.3	Problem Assumptions	69
4.4	Formulation of CDJRP-TC	71
4.4.1	Notations	71
4.4.2	Relevant Costs	74
4.4.3	Analytical Model Formulation: Mixed Integer Linear Programming	75
4.4.4	Discrete Simulation Model Implementation	77
4.5	Solution Approach	77
4.5.1	Overview of Proposed Method	77
4.5.2	Phase 1: Simplified MILP Optimization	79
4.5.3	Phase 2: Policy Variable Estimation	80
4.5.4	Phase 3: Update of Ideal Inventory Quantities	82
4.6	Computational Experiments	83
4.6.1	Conditions	83
4.6.2	Optimization Result Comparison	84
4.6.3	Planning Capability for Different Unit Transportation Costs	88
4.7	Conclusion	91
5	Fast Simulation-based Order Sequence Optimization Assisted by Pre-trained Bayesian Recurrent Neural Network	93
5.1	Introduction	93

5.2	Related Works	96
5.2.1	Order Sequencing Problem	96
5.2.2	Simulation-based Combinatorial Optimization	97
5.2.3	Surrogate-assisted Optimization	97
5.3	Methodology	98
5.3.1	Problem Setting and Formulation	98
5.3.2	Surrogate Pruning Optimization Framework	99
5.3.3	Selection of Surrogate Models	100
5.3.4	Proposed Optimization Method: BRNN-SA	102
5.4	Evaluation	103
5.4.1	Conditions	103
5.4.2	Prediction Accuracy Comparison	105
5.4.3	Optimization Runtime Comparison	106
5.5	Conclusion	108
6	Hyper-heuristics Framework for Online Order Batching and Sequencing Problem in Automated Logistics Warehouses with Order Consolidation Process	109
6.1	Introduction	109
6.2	Online Order Batching and Sequencing Problem	111
6.2.1	Problem Definition and Representative Solution Methods	111
6.2.2	Mathematical Optimization Model of OBSP	112
6.2.3	Formulation of Simulation-Based OOBSP	114
6.3	Methodology	118
6.3.1	Hyper-heuristics	118
6.3.2	Low-level Heuristics: LLHs	119
6.3.3	High-level Strategy: HLS	121
6.4	Numerical Experiments	123
6.4.1	Experimental Conditions	123
6.4.2	Experimental Results	125
6.5	Conclusion	129
7	Discussion and General Conclusion	132
7.1	Summary of Case Studies and Key Insights	132
7.2	Proposed Decision-making Framework	133
7.3	Conclusion and Future Directions	135

CONTENTS

References	137
Research Accomplishments	157

Chapter 1

General Introduction

1.1 Background

Since the Industrial Revolution, manufacturing systems have continually adapted to shifting social needs and technological progress [1]. Successive paradigms — line and mass production, statistical quality control, Just-in-Time (JIT), cellular manufacturing, and mass customization — emerged as context-appropriate responses. Today, pervasive IoT, digital twins, and automation [2], heightened supply uncertainty from the COVID-19 pandemic and geopolitical tensions [3], [4], and increasing pressure for lower environmental impact and regulatory compliance [5] are pushing firms to advance technology while improving efficiency to stay competitive.

Efficiency and resilience require coordination beyond isolated factory and warehouse improvements, integrating upstream (procurement, replenishment) and downstream (shipping, last-mile delivery) activities. Effective decisions therefore rely on an end-to-end Supply Chain Management (SCM) perspective with integrated planning and control [6]. SCM decisions are commonly organized into three levels by time horizon and scope: strategic, tactical, and operational (Figure 1.1) [7], [8]. These levels differ in granularity and frequency yet are tightly linked: strategic decisions set the long-term network structure; tactical planning balances supply and demand within that structure; operational control executes and adapts daily activities.

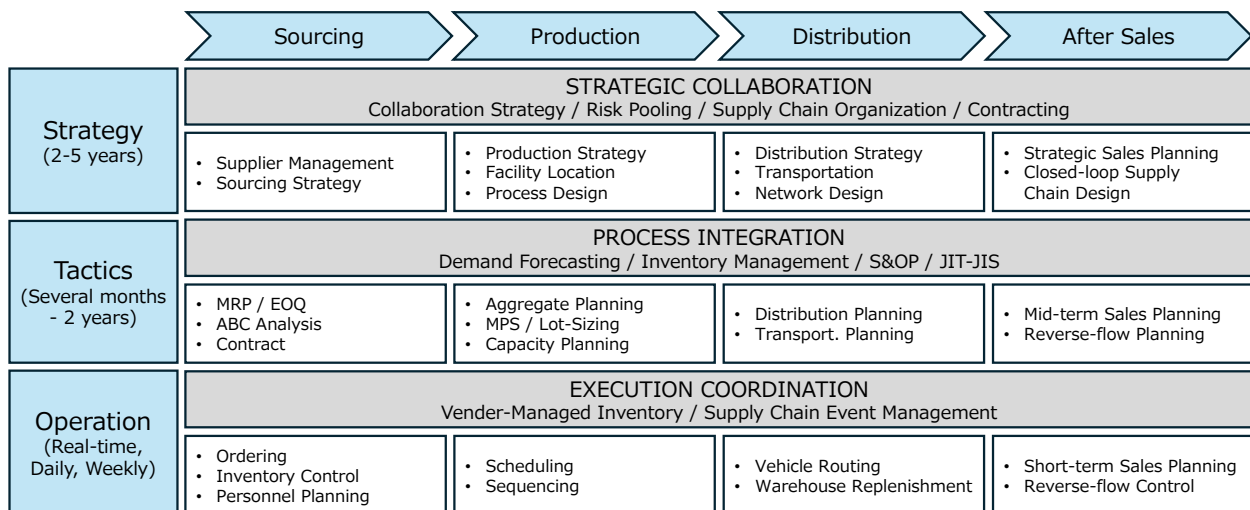


Figure 1.1. Decision matrix in supply chain management and operations

1.2 Strategic Decision Making

At the strategic level, firms define the long-term supply chain structure and policies for a two- to five-year horizon [9]. Because later changes are expensive once capital is committed, these decisions require careful analysis. Main decisions include the number, location, and capacity of plants and distribution centers; selection and qualification of key suppliers; the product portfolio; and sourcing policies.

Key challenges are (i) uncertain long-term demand and market conditions and (ii) the scale and complexity of models that must capture many discrete choices—facility locations, capacity allocations, and flow assignments—and multiple, often conflicting objectives (cost, service, environmental impact) [10], [11]. Thus, strategic design is framed as a large-scale combinatorial, often multi-objective, optimization problem [12]. Common solution approaches employ mixed integer linear programming (MILP) and metaheuristics, such as genetic algorithms and simulated annealing, supported by simulation to assess candidate networks [13]. Given sparse and uncertain data, highly complex statistical or machine-learning models trained on historical records are often unjustified. Strategic supply chain design has been widely studied. In facility location, numerous large-scale studies employ metaheuristics to obtain near-optimal solutions when exact methods are infeasible [14]. Reviews of network design emphasize multi-objective models combining cost, service, and sustainability, and stochastic formulations to address demand uncertainty [12].

1.3 Tactical Decision Making

The tactical level translates the strategic design into medium-term plans that match supply and demand and allocate resources over horizons of several months to about two years (e.g., quarterly or annual) [9]. Typical activities include demand forecasting and integrated planning of production, inventory, transportation, and procurement. This cross-functional work is often organized as Sales and Operations Planning (S&OP), which builds enterprise-wide consensus between demand (sales/marketing) and supply (manufacturing/procurement/logistics) to improve overall performance [15].

The main challenge is uncertainty and nonstationarity in both demand and supply. Because forecasts are imperfect, plans require continuous adjustment. Many firms run monthly, rolling S&OP cycles to incorporate updated signals on demand variability, inventory positions, open orders, and capacity limits. Compared with the strategic level, tactical planning benefits from richer data and models that capture more detailed constraints and cost structures. Common methods include statistical and machine-learning demand forecasting; MILP for integrated production-inventory-distribution planning; and discrete-event simulation for policy evaluation. The S&OP literature

documents process maturity and its performance impacts [16] and offers integrative reviews and best practices [17]. Illustrative models include integrated optimization of demand, inventory, and production [18] and profit-maximizing global network models [19].

1.4 Operational Decision Making

Operational decisions execute tactical plans over short horizons—from real time to a few weeks—and include factory and warehouse scheduling, vehicle routing and job dispatching [9]. The central challenge is timeliness: even with sound upper-level plans, forecast errors, disruptions, and local anomalies arise daily, leaving little time for computation and forcing frequent re-optimization. Most operational problems are combinatorial and NP-hard (e.g., job-shop scheduling, vehicle routing), so exact optimization is rarely feasible within required response times; practitioners therefore rely on heuristics and metaheuristics. Classical operations research provides key building blocks, such as Johnson’s rule for two-machine flow shops [20], and standard formulations and datasets—e.g., Solomon’s VRP instances [21]—that have driven decades of algorithmic progress.

Operational decision making is becoming more complex as automation expands—e.g., with automated guided vehicles (AGVs), automated storage and retrieval systems (AS/RS), and collaborative robots (cobots). These systems introduce equipment-specific, dynamic behaviors that must be modeled. Consequently, practitioners increasingly combine high-fidelity simulation (digital twins) with optimization and adopt machine learning and reinforcement learning for real-time control, such as robotic trajectory optimization and warehouse fleet coordination [2].

1.5 Research Objectives and Contributions

As noted above, supply chain decision making has long been central to Operations Research and Industrial Engineering, but the need for new methods has intensified. Trends such as greater product variety, shorter lifecycles, and same-day delivery have turned SCM decisions into larger-scale optimization problems with tighter constraints [2]–[4]. To meet these service targets, firms are adopting automated production and logistics systems that leverage robots and AI. At the same time, the pandemic and geopolitical shocks mean that factors once treated as fixed — demand, supply, and transportation capacity — are now nonstationary and uncertain. Against this background, this thesis focuses on two major challenges facing modern production and logistics systems:

- (A) Optimizing the design and operation of automated and complex production systems.
- (B) Planning production systems under time-varying uncertainty and nonstationarity.

Simulation models are widely used to evaluate such decisions because they provide higher

fidelity to dynamic real-world behavior than analytical models [22]–[24]. However, most established methods were designed for analytical formulations (e.g., mathematical programming) and do not transfer directly to simulation. Key obstacles include the lack of exploitable mathematical structure (e.g., gradients), long run times, and complex, nonlinear input–output relationships. As a result, obtaining timely solutions is difficult and solution quality (**optimality**) may degrade.

Note that real-world decision-making problems are highly diverse, making it difficult to derive a single, universally optimal solution approach. This diversity arises because the problem setting inevitably changes with the decision maker’s context — such as factory and product characteristics, corporate strategy, and the managerial and financial situation. At the same time, this characteristic poses a major obstacle to practicing quantitative decision-making: each problem often requires its own tailored model and solution method, and developing them repeatedly becomes costly and time-consuming.

To address this challenge, we focus on the observation that, although individual problems vary widely, the information available to decision makers and the outcomes required can be broadly categorized according to the decision-making hierarchy (e.g., strategic, tactical, and operational levels). Building on this insight, we systematically organize an approach for developing modeling and optimization methods that achieve an appropriate balance between model fidelity and solution optimality for each hierarchical level. We then formalize this organization into a practical framework that provides guidance for designing solution methods for new problems that share similar characteristics. For this purpose, this thesis studies decision making at three levels — (1) strategic, (2) tactical, and (3) operational — through five case studies that cover challenges (A) and (B) in Chapters 2–6. In each case, we formalize the problem and propose systematic methods that balance model fidelity with decision optimality. We then synthesize the results into a unified framework to develop a decision-making methodology for modern production and logistics systems in Chapter 7.

The structure and contents of this thesis are outlined in Figure 1.2. Chapter 1 introduces the background, objectives, and contributions. Chapter 2 examines strategic decision-making for automated manufacturing systems. It proposes a multi-objective conceptual design framework for robot-cell systems that integrates productivity, layout, and manipulability, making trade-offs explicit and enabling transparent and quantitative decision-making. Building on the need to reason about (B) long-term uncertainty, Chapter 3 proposes surrogate-assisted scenario generation, which can be applied to (1) strategic and (2) tactical decision-making. This chapter presents a surrogate-assisted, general scenario generation method that captures uncertainty with fewer scenarios, improving the scalability of robust decision-making. At (2) the tactical level for (A) complex supply chains under (B) nonstationarity, Chapter 4 addresses emerging S&OP needs where transportation costs and fleet capacities vary over time. It integrates sales, inventory, and transport capacity planning by coupling mathematical programming with simulation-based evaluation, yielding solutions within

Chapter 1: General Introduction		
(A) Automated / Complex System		(B) Uncertainty / Nonstationarity
(1) Strategy (2-5 years)	Chapter 2: Simultaneous Optimization of Layout and Task Schedule for Robotic Cellular Manufacturing Systems	Chapter 3: Surrogate-assisted Scenario-generation Method for Simulation-based Stochastic Programming Problems
(2) Tactics (Several months - 2 years)	Chapter 4: Integration of Sales, Inventory, and Transportation Resource Planning by Dynamic-demand Joint Replenishment Problem with Time-varying Costs	
(3) Operation (Real-time, Daily, Weekly)	Chapter 5: Fast Simulation-based Order Sequence Optimization Assisted by Pre-trained Bayesian Recurrent Neural Network	Chapter 6: Hyper-heuristics Framework for Online Order Batching and Sequencing Problem in Automated Logistics Warehouses with Order Consolidation Process
Chapter 7: General Conclusion		

Figure 1.2. Thesis structure and contents

operational time limits without sacrificing realism. At the operational level (3), Chapter 5 addresses short-term planning in (A) automated warehouses by optimizing picking sequences. To support fast decisions grounded in high-fidelity simulations that reflect complex system dynamics, it introduces a scheduling approach that combines surrogate models with metaheuristics. Chapter 6 develops a reinforcement learning-based hyper-heuristic for (3) real-time operations in (A) automated warehouses with (B) dynamic environments, where shipping orders arrive sequentially. It enables online, context-aware decision making that improves end-to-end system performance under dynamic order arrivals. Finally, Chapter 7 outlines a decision-making framework that synthesizes the findings of Chapter 2–6 and discusses future research directions.

Simultaneous Optimization of Layout and Task Schedule for Robotic Cellular Manufacturing Systems

2.1 Introduction

Robotic cellular manufacturing systems (RCMS) utilize one or more articulated industrial robots that perform a large number of complex assembly operations normally conducted by human operators in conventional cellular manufacturing systems. RCMS have advantageous features similar to those of conventional human-based cellular manufacturing systems, such as high flexibility with respect to product changes and low material handling cost. However, due to the benefits of factory automation, RCMS provide significant production advantages compared with manual assembly systems. To fully exploit the advantageous features of RCMS, quick deployment of manufacturing systems that have efficient configurations is paramount. But since a number of articulated robots in an advanced RCMS cooperatively execute highly complex operations, RCMS design requires deep knowledge and abundant experience in the field of robotics and assembly processes, so development of effective support systems for RCMS layout design is a practical necessity.

Layout design is one of the most important decision-making processes when developing a new RCMS [25]. During the layout design process, industrial robots and other manufacturing system components are allocated at specific positions so that the assigned assembly operations can be conducted appropriately. Since the layout design process is upstream of the detailed robot motion design process, many robot motion design decisions are based on the result of the layout design process, which typically determines, and may limit, fundamental performances of the RCMS.

Numerous optimization techniques for RCMS facility layout problems have been proposed, to support decision-making at the layout design stage [26][27]. Unfortunately, these methods require explicit constraint handling regarding component overlapping, since component coordinates are handled as design variables, and this implementation obstructs global searching of the solution space. On the other hand, the technique of sequence-pair representation, when used in layout problems, can avoid overlapping among components and accommodate a variety of rectangle sizes

that represent facility components [28]. Several papers report that this type of representation enables very effective layout optimization in packing-type problems [29] and facility layout problems [30][31].

Izui et al. [32] proposed a layout optimization method using sequence-pair representation to assist the conceptual design of a RCMS. Practical layout design problems ordinarily have multiple metrics [33][34], and layout design problems should be formulated as multiobjective optimization problems rather than single objective optimization problems. Therefore, Izui et al. [32] clarified the design criteria used at the conceptual design stage and a multiobjective genetic algorithm was employed to obtain non-dominated solutions, but it was assumed that only a single industrial robot was present. We now extend that method to handle cases in which two or more industrial robots operate simultaneously to carry out complex assembly operations.

2.2 Multi-robot Cellular Manufacturing System

A typical RCMS comprises one or more industrial robots, part boxes, tool exchangers, and an assembly table. The positions of these components are determined during the RCMS layout design process. After the component layout is set, manufacturing engineers determine detailed robot motions based on the provided layout design [35], which makes RCMS operational efficiency highly dependent on such layout design results [36]. Furthermore, if, due to a poorly realized layout design, it is found to be infeasible during the detailed motion design stage when the numerous tasks to be performed are assigned and choreographed, the design process must return to the layout design stage to implement design changes, causing unwelcome delays. Hence the need for a systematic layout design method that provides reasonable design solutions while considering the detailed tasks to be performed. In cases where multiple robots are used, the design process becomes far more complex because the robots typically carry out their assigned tasks in parallel. Multiple assembly operation tasks must be allocated to each robot and the usual procedure is to iteratively conduct the RCMS layout design process, based on the results of task allocations considering the need to avoid robot collisions during task operations [37].

In any case, task allocation and layout design results are interdependent [38]; choices made during the layout design and task allocation affect each other. Therefore, obtaining efficient manufacturing systems requires that decision-making for both layout design and task allocation be conducted simultaneously [39]. We therefore propose an optimization technique for multi-robot cellular manufacturing systems that simultaneously deals with both task allocation and manufacturing layout design.

In the following sections, the design variables for task allocation and layout design are codified, the RCMS design criteria are clarified, and an optimization procedure employing a multiobjective

genetic algorithm is then presented.

2.3 Design Variables

2.3.1 Assumptions for RCMS Layout Design Problem

In this paper, we propose a design optimization technique for the layout of a RCMS in which multiple robots carry out assembly operations. We assume that the list of assembly operations that the robots will perform is preassigned, and the performances and sizes of manufacturing system components such as the robots and the assembly base are also given in advance. During the layout design process, design engineers must determine which robot will conduct each assembly operation and also determine the position of each manufacturing system component, decision-making tasks that are interrelated. For instance, since the time to complete a single task by a given robot depends on the robot's posture as it completes various tasks, different component layouts will result in unique optimal task sequences. Moreover, to avoid collisions between robots, task allocations and component locations must be appropriately adjusted during the creation of a layout design. Design variable encoding techniques for determining task allocations and component locations are explained next.

2.3.2 Encoding Technique for Task Allocation

The total operation time required to complete all allocated tasks must be computed to evaluate a layout solution candidate. Operational tasks performed by robots can be classified into three categories: assembly tasks, part acquisition tasks, and tool changing tasks. Furthermore, the time required for each of these tasks is considered to be the sum of the time required to carry out the actual operation and the time required for robot movement. Moreover, when multiple robots are used in a RCMS, robot idling time must also be considered. To obtain highly efficient RCMS design solutions, the allocation of the various operational tasks must consider these factors for each robot. Although the time required to conduct a task operation can be easily estimated, robot movement time and idling time depend on both the task allocation and the layout design, so these two aspects must be simultaneously optimized.

Consider that a total of N_p assembly tasks are allocated to N_r industrial robots. A single end-effector of a given robot is seldom appropriate for performing all tasks, so industrial robots usually need to change their configuration to match the particular assembly task that must be performed. We assume that there are K_p types of task, *i.e.*, N_p assembly tasks can be classified into K_p types of task. In the following, the $\mathbf{A}$ vector represents a sequence of tasks and the required operational

task types, where the i -th element of the vector represents the i -th task.

$$\mathbf{A} = [A_1, A_2, \dots, A_{N_p}] \quad (2.1)$$

We use vector Φ to represent integer design variables for allocated tasks, as follows:

$$\Phi = [\phi_1, \phi_2, \dots, \phi_{N_p}], \quad (2.2)$$

$$\phi_i \in \{1, 2, \dots, N_r\}, \quad (2.3)$$

where ϕ_i indicates the robot number that is to perform the i -th task. Using the above vectors, we can now construct task allocation vectors $\mathbf{B}^l$ for the l -th robot, as follows:

$$\mathbf{B}^l = [B_1^l, B_2^l, \dots, B_{N_p}^l], \quad (2.4)$$

$$B_i^l = \begin{cases} A_i & \text{if } \phi_i = l \\ 0 & \text{if } \phi_i \neq l, \end{cases}$$

where $B_i^l = 0$ means that the l -th robot does not have an i -th task, a condition that may cause an idling state.

Each operational task indicated by A_i is now separated into assembly tasks, part acquisition tasks, and tool changing tasks. In this paper, these separate tasks are denoted as detailed tasks and the detailed tasks of task X assigned to robot l are represented as X_A^l , X_T^l and X_P^l , with the subscripts A , T and P denoting Assembly, Tool Change, and Part Acquisition tasks, respectively. These detailed tasks include the robot movement times needed to change robot postures, since each robot assumes different postures according to detailed task requirements.

The detailed task sequence for each robot is obtained based on the following conditions.

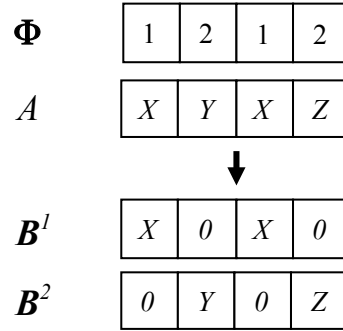
Definition 1 *Detailed task requirements*

1. When robot l retrieves a part from a part box or attaches a part to a product being assembled, the robot must be equipped with the end-effector that is appropriate for the particular part.
2. Before robot l can use a part for an assembly task, it must retrieve the part from the appropriate part box.
3. Robot l can assemble only a single part during a given assembly task.
4. Multiple assembly tasks cannot be performed during the same time period, although multiple robots exist. That is, a robot cannot start an assembly task if another robot is conducting an assembly task.
5. All tasks must be performed as soon as possible after a given task is completed.

An example of the construction process for an assembly task schedule based on the design variables Φ is shown in Fig. 2.1. This example illustrates a case in which two assembly robots conduct four assembly tasks that contain three types of tasks, X, Y and Z, which correspond to such as screw fastening and part attaching in actual assembly operations. In this example, the number of robots $N_r = 2$, the number of assembly tasks $N_p = 4$ and the number of the types of tasks $K_p = 3$.

In the first process, task allocation vectors $\mathbf{B}^1$ and $\mathbf{B}^2$ for two robots are constructed by separating the task-sequence vector $\mathbf{A}$ according to Eq. 2.5. In the second step, task schedules for each robot are generated from task allocation vectors $\mathbf{B}^1$ and $\mathbf{B}^2$ according to Def.1. In this example, the first X task of allocation vector $\mathbf{B}^1$ and the Y and Z tasks of allocation vector $\mathbf{B}^2$ are each separated into their respective detailed tasks. The second X task of robot 1 is divided into only two detailed tasks, an assembly task and a part acquisition task. The tool changing task is not need to be conducted because robot 1's preceding task was X, hence it is already equipped with the correct end-effector for the X task.

1. Construct task allocation vector $\mathbf{B}^i$ from Φ and $\mathbf{A}$



2. Divide $\mathbf{B}^i$ and generate task schedules

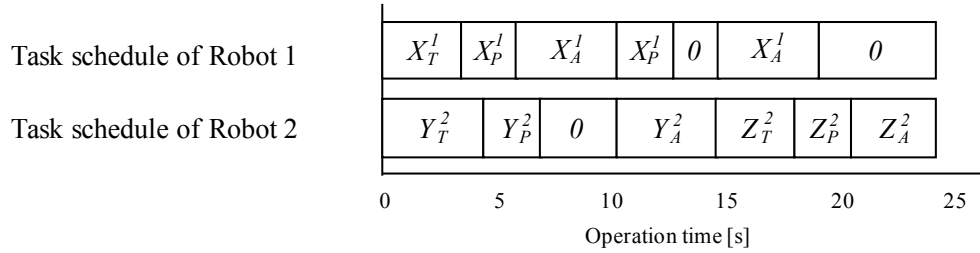


Figure 2.1. Construction process example for a detailed task schedule.

2.3.3 Encoding Technique for Component Positioning

Component positioning design problems can be viewed as problems in which the planar positions of variously sized components must be optimized. One intuitive method for obtaining optimal

positions would be to optimize the coordinates of the components, but this is problematic since additional constraints must be used to prevent overlapping of the components, which may occur when using conventional mixed-integer approaches [40][41]. We therefore use a sequence-pair representation technique that automatically avoids component overlapping [28].

The sequence-pair approach employs a pair of sequences, Γ_+ and Γ_- , where each sequence indicates the relative positions of manufacturing system components that are assumed to be rectangles. Furthermore, in this paper, an additional binary variable, $\Theta = \{ \theta_m \}$ ($m = 1, 2, \dots, M$), is introduced to represent the orientation of each allocated rectangular component, where m denotes the component number and M is the total number of components. Component orientation, *i.e.*, whether the long side is vertical or horizontal, is represented as $\theta_m \in \{0, 1\}$.

Layouts using sequence-pair representation can be decoded using an oblique grid. For example, given $\Gamma_+ = \{d b c e f g a\}$ and $\Gamma_- = \{b f c a d g e\}$, the two permutations, Γ_+ and Γ_- , represent the relative positions of the components on this oblique grid, and the component sequences in the two grid line labels are the same as those of the permutations, as shown in Fig. 2.2. That is, the label sequence from the top corner toward the right is the same the Γ_+ sequence, and the label sequence from the leftmost corner toward the top corner is the same the Γ_- sequence. As shown in Fig. 2.2, components are located at the intersection of corresponding pairs of the diagonally drawn component grid lines.

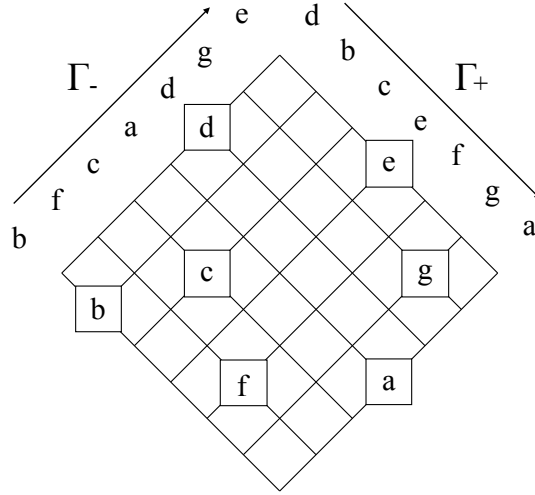


Figure 2.2. Relative arrangement.

Two directed and vertex-weighted graphs can be constructed on the oblique grid to diagram horizontal and vertical constraints, as shown in Fig. 2.3. In these graphs, the position of each vertex indicates the corresponding component layout. In the horizontal constraint graph, two vertexes indicated two components are connected by an edge as illustrated by arrow lines when one component is to the left of another component in both sequences. Components in the vertical

constraint graph are connected when a component letter label is to the left of the corresponding component letter label in Γ_+ and, analogously, to the right in Γ_- . The position of each component is then calculated by solving the longest path problem, considering the orientation of the components indicated by their θ_m value. Ultimately, a layout is obtained, as shown in Fig. 2.4. This procedure is discussed in detail in the reference [28].

In the sequence-pair approach, the sequence of letter labels represents the relative position of the components, but the space between the components is not considered. In RCMS layout design problems, however, the inclusion of adequate space between components is extremely important, since robot manipulation requires a certain minimal distance from task points, to allow complex assembly tasks to be conducted at these points [42]. In this paper, we use a dummy component scheme [32] to represent these crucial spaces in RCMS layout problems. In this scheme, a set number of rectangular dummy components, representing an empty space rather than any manufacturing component, are included in the optimal layout design problem, as shown in Fig. 2.5, and the layout of both normal and dummy components is simultaneously optimized. Consequently, dummy components are allocated at places where space is required so that layout optimization using sequence-pair representation can consider the spacing requirements for the particular robots being used.

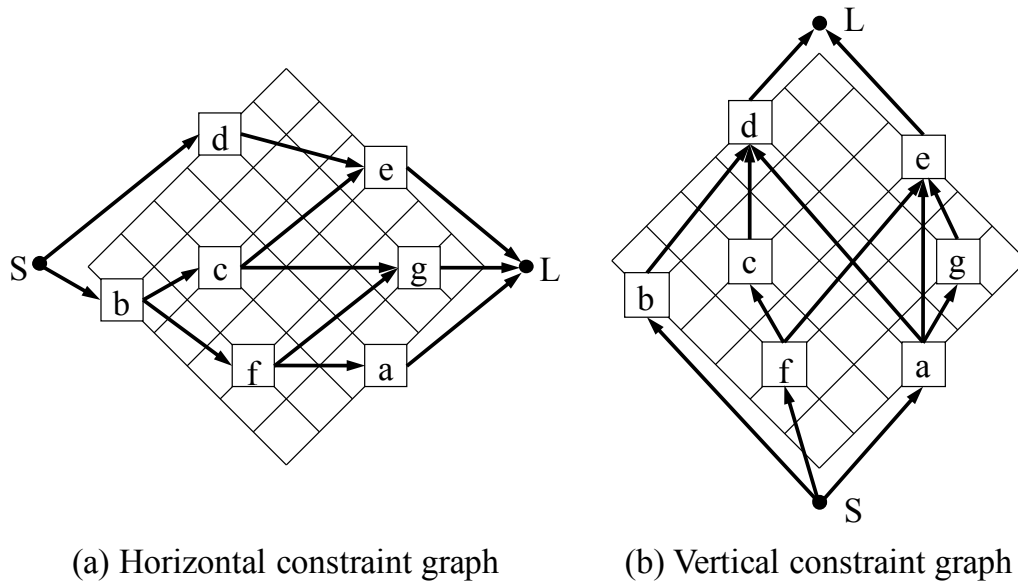


Figure 2.3. Constraint graphs.

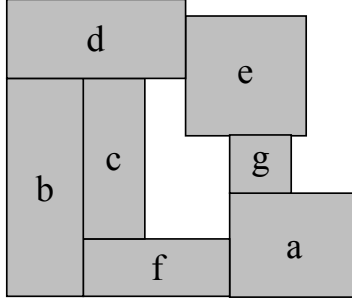


Figure 2.4. Component layout.

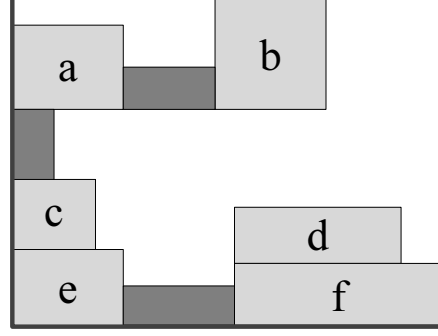


Figure 2.5. Dummy representation of space between components.

2.3.4 Design Variables for RCSM Layout Design Problem

We now propose the design variables that must be solved in the RCSM layout design problem, using a combination of the above methods. To solve this optimization problem, the design variable $\mathbf{x}$ is defined as follows so that the task allocation and layout design can be simultaneously optimized.

$$\mathbf{x} = [\Gamma_+, \Gamma_-, \Theta, \Phi] \quad (2.5)$$

With respect to variable $\mathbf{x}$, Γ_+ and Γ_- represent the relative positions of each component using sequence-pair representation with integer variables, and Θ expresses the orientation of these components using binary variables. These three variables are termed layout variables in the following sections. Integer variable Φ , on the other hand, indicates the allocation of assembly tasks to each robot in the robotic cell. In this paper, the layout design problem is represented as $\mathbf{x}$, the component layout is decoded from layout variables Γ_+ , Γ_- , and Θ , and the allocated tasks are regenerated from Φ using the decoding methods explained in the above sections. All of the variables in $\mathbf{x}$ are integers and nominal scale. Therefore, the optimization problem in which $\mathbf{x}$ is used as the design variable can be formulated as a combinatorial optimization problem.

2.4 Design Criteria for Layout Design

2.4.1 Layout Design Problem

The most important consideration when developing new manufacturing systems is productivity. Barral et al. [26] and Tay et al.[27] have proposed single-objective layout optimization algorithms that minimize RCMS operation time in order to maximize productivity. However, during the RCMS layout design stage, multiple design requirements must be considered, and feasibility verification of

assembly tasks is also an essential process. Another highly important consideration is minimization of layout area, to enhance the efficiency of factory-level layouts. These requirements are closely related to the design variables discussed above, and their quantitative criteria are discussed in the following subsections.

2.4.2 Operation Time

The operational efficiency of a RCMS can be evaluated by considering the cycle time. RCMS cycle time is the maximum value of the total operation time for the group of robots, which can be estimated as the sum of the setup time (including robot movement time), the assembly time, and idling time. Unfortunately, precise robot motions are not known at the conceptual design stage, since detailed motion plans are as yet undetermined. Therefore, we evaluate robot movement times using the angle through which robot arm joints rotate between the starting and terminal points for each robot motion. Since the maximum angular velocity for each assembly robot joint is usually given at the layout design stage, we calculate approximate robot movement times by assuming that the joint motion has a uniform velocity based on this maximum angular velocity. Thus, robot movement time $t_{l,j,k}^m$ for the k -th joint of robot l performing the j -th motion is given as

$$t_{l,j,k}^m = \frac{|\bar{q}_{l,j,k}|}{\omega_{l,k}}, \quad (2.6)$$

where $\omega_{l,k}$ indicates the maximum angular velocity of the k -th joint of robot l , and $\bar{q}_{l,j,k}$ is the angle through which the k -th joint moves during the j -th motion performed by robot l . We assume that all joints can simultaneously have their angle changed, and the j -th movement time of robot l is now evaluated using the maximum joint motion time, as follows:

$$t_{l,j}^m = \max_k \left[\frac{|\bar{q}_{l,j,k}|}{\omega_{l,k}} \right]. \quad (2.7)$$

In addition, we define a standard processing time $t_l^s(A_j)$ for operational task A_j of robot l . This standard processing time is a constant whose value depends on the particular assembly robot and tool type, and does not include the robot movement time before and after task A_j is performed. Thus, the robot task completion time $T_{l,j}$, which represents the sum of the movement time from the final posture at the $j - 1$ -th task to the initial posture at the j -th task plus the actual operation time required by the j -th task of robot l , is given as

$$T_{l,j} = t_{l,j}^m + t_l^s(B_j^l). \quad (2.8)$$

By using Eq. 2.8 and task allocation vectors $\mathbf{B}^l$ described in in section 2.3.2, the total operation

time T for the RCMS is now defined as

$$T = \max_l \left\{ \sum_j^{N_l^l} T_{l,i} \mid l = 1, 2, \dots, N_r \right\}, \quad (2.9)$$

where N_l^l is the number of detailed tasks allocated to robot l .

2.4.3 Manipulability

Ensuring that robot arms reach their appropriate task positions is only one of the necessary conditions required for conducting an assembly task, but this condition alone is insufficient because RCMS robots must carry out complex movements when performing assembly tasks. If the robot's posture at a task position is close to a singular point, where the robot's end-effector loses one or more degrees-of-freedom of its available motion, the assembly task may not be successfully completed. Therefore, consideration of how to avoid such singular points at the layout design stage is essential for obtaining a feasible assembly process.

To avoid singular points, evaluation of kinematic manipulability [43] is beneficial. Manipulability is a quantitative measure of the manipulating ability of robot arms that can be calculated based on the relationship between the position of the end-effector and the joint vectors. Consider an n degree-of-freedom assembly robot that conducts tasks at various positions, with its h -th joint variable at the k -th task position denoted as q_k^j , using joint vector $\mathbf{q}^j$. The robot posture vector $\mathbf{r}^j$ at the j -th position ($j = 1, 2, \dots, h$) is defined as follows:

$$\mathbf{q}^j = [q_1^j, q_2^j, \dots, q_n^j]^T \quad (j = 1, 2, \dots, h) \quad (2.10)$$

$$\mathbf{r}^j = [r_1^j, r_2^j, \dots, r_m^j]^T, \quad (2.11)$$

where n is the number of dimensions needed to represent the position and angle of the end-effector.

In the following discussion, the notation for task position j is ignored for simplicity. By the chain-rule, velocity vector $\mathbf{v}(= \dot{\mathbf{q}})$ is obtained as

$$\mathbf{v} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \quad (2.12)$$

where $\dot{\mathbf{q}}$ indicates the joint velocity vector and $\mathbf{J}(\mathbf{q})$ is the Jacobian. Consider subset S_v of the realizable velocity v such that the corresponding joint velocity $\dot{\mathbf{q}}$ satisfies

$$\|\dot{\mathbf{q}}\| = (\dot{q}_1^2 + \dot{q}_2^2 + \dots + \dot{q}_n^2)^{1/2}. \quad (2.13)$$

This subset is an n -dimensional Euclidean ellipse and its volume, V , is given as

$$V = c_n w, \quad (2.14)$$

where coefficient c_n is a constant value that depends on n . The volume of the n -dimensional ellipse can be evaluated using w , which is referred to as a manipulability measure [43]. The larger the manipulability w , the faster the end-effector can move in any direction, implying that larger manipulability values of w indicate that the robot can more easily conduct complex tasks. Therefore, we use calculated values of w to evaluate the feasibility of the assembly tasks. In the case of non-redundant robots, w is calculated as follows:

$$w = |\det \mathbf{J}(\mathbf{q})|. \quad (2.15)$$

Therefore, using superscript notation to represent robot l and task position j , the overall feasibility of assembly task W in a RCMS can be evaluated using a summation of manipulability values $w_{l,j}$ as follows:

$$W = \sum_{l=1}^{N_r} \sum_{j=1}^{N_l^l} w_{l,j}, \quad (2.16)$$

where $w_{l,j}$ is the manipulability at the j -th task position of robot l .

2.4.4 Layout Area

In this paper, every manufacturing system component is assumed to be rectangular and layout area S is evaluated using the area of the smallest rectangle that can contain all allocated components, as shown in Fig. 2.6.

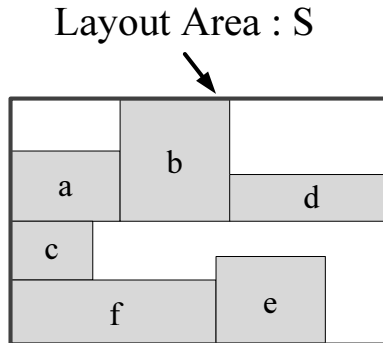


Figure 2.6. Example of layout area S .

2.4.5 Optimization Problem

Using the above three design criteria for the RCMS layout, a multiobjective optimization problem for its design can be formulated as

$$\begin{cases} \text{minimize } T, S \\ \text{maximize } W. \end{cases} \quad (2.17)$$

If one or more task positions lie beyond the working envelope of a robot, a very large penalty value is assigned to each objective function. Another constraint condition for this optimization problem is to avoid overlapping among components, and to handle this constraint, a sequence-pair representation scheme is introduced in the following section.

2.4.6 Constraints for Collision Avoidance

Sequence-pair representation can be used to avoid the problem of overlapping among system components. An additional important constraint that must be considered in RCMS layout designs is the avoidance of collisions between robots during assembly operations. In this paper, two types of collision states are considered: 1) collision between two robot arms, as shown in Fig. 2.7, and 2) collision between a robot arm and a robot base, as shown in Fig. 2.8. In the proposed method, the potential for a collision to occur is detected at every start and end time of the operational tasks. If a collision is detected, the layout solution is significantly penalized so that more feasible solutions are obtained.

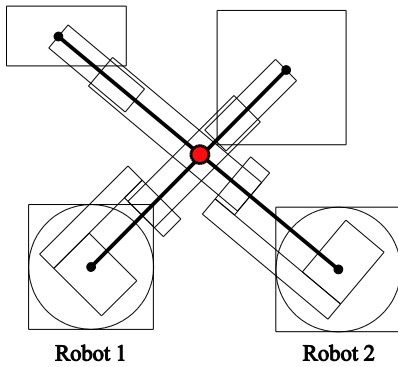


Figure 2.7. Component collision, type 1.

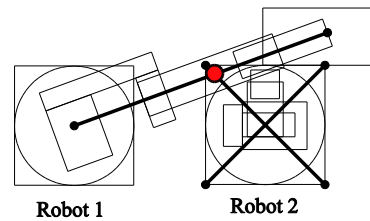


Figure 2.8. Component collision, type 2.

2.5 Optimization Technique

2.5.1 Multiobjective Genetic Algorithm

The proposed layout optimization formulation includes integer design variables that represent task allocations, and sequence and binary design variables that represent the location and orientation of components, respectively. The optimization problem to be solved is a large-scale combinatorial problem that can be efficiently handled by meta-heuristic approaches such as genetic algorithms [30] [44], simulated annealing [45], and swarm intelligence algorithms [46][47][48][49].

In this paper, a multiobjective genetic algorithm (MOGA) is used for the RCMS layout design problem since it can provide comprehensive non-dominated solutions without having to convert the original multiobjective problem to a single-objective optimization problem. An additional benefit is that the obtained non-dominated solution set can be helpful to decision-makers when analyzing trade-off relationships between the objective functions, which facilitates the selection of satisfactory design solutions. The following sections describe crossover, mutation and selection operators of MOGA.

2.5.2 Crossover

For the crossover process, we use a hybrid approach in which both the Placement-based Partially Exchanging Crossover (PPEX) scheme [50] and a uniform crossover scheme are used, as shown in Fig. 2.9. PPEX is an effective crossover algorithm for layout optimization problems that rely on sequence-pair representation. In the PPEX algorithm, a pair of individuals in MOGA is chosen first, then one component in one individual of the pair is selected at random, and a window domain is created around this component. The sequence and orientation of components inside this window are exchanged with those in the other individual. However, for the design variable $\mathbf{x}$ explained in Section 2.3.4, PPEX is applicable only to the layout variables, Γ_+ , Γ_- and Θ . We therefore apply the uniform crossover algorithm, a general crossover approach to interger variables, to the task allocation variable Φ .

2.5.3 Mutation

During the mutation process, random values are probabilistically applied to integer variables. Firstly, for the mutation procedure applied to Γ_+ and Γ_- , the positions of two genes selected at random are probabilistically exchanged. Secondly, the some Θ genes are selected probabilistically, and their binary variable values are inverted. Finally, probabilistically selected genes in Φ are changed to a random integer value based on the number of robots.

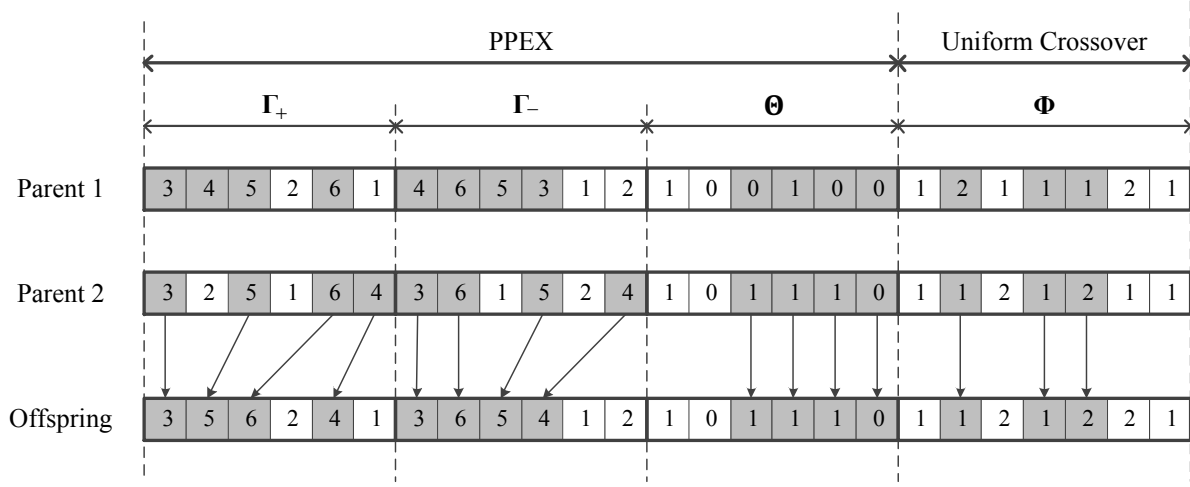


Figure 2.9. Genetic operation: Crossover.

2.5.4 Selection

NSGA-II (Non-Dominated Sorting Genetic Algorithm-II) [51] is used in the selection process so that well-distributed non-dominated solutions are obtained in the multiobjective optimization problem. In the NSGA-II selection scheme, non-dominated solutions are preserved and carried over into the next generation, and a crowdedness criterion is used to select non-dominated solutions existing in sparsely distributed areas, to obtain widely distributed non-dominated solutions.

2.6 Example Problem

2.6.1 Problem Settings

The proposed method was applied to an example problem in which two industrial robots assemble a product. Figure 2.10 shows the configuration of the industrial robots, RV-2SQ models manufactured by Mitsubishi Electric Corporation, and the maximum angular velocity data for the joints, used to estimate robot movement times, are provided in Table 2.1.

The manufacturing system components in this example are specified in Table 2.2. Three kinds of parts are provided at Part Boxes 1, 2, and 3. A robot removes a single part from a box and brings it to the assembly table. The part is then assembled to a target object in a single assembly operation. The length and width of the components are used to determine the size of the rectangles that the sequence-pair layout representation scheme must handle.

Dedicated robot end-effectors, *i.e.*, part-handling tools for grasping specific parts, are stored in an individual Tool Box for each robot. When a robot is tasked with an assembly operation

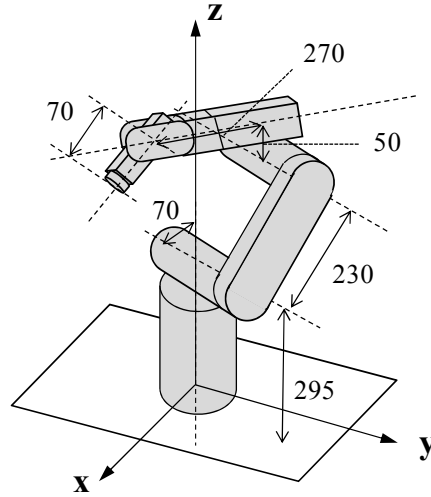


Figure 2.10. RV-2SQ robot.

Table 2.1. Maximum angular velocities of joints.

Joint number	Maximum angular velocity [deg/s]
1	225
2	150
3	275
4	412
5	450
6	720

that requires a part that is not the same as that used in the preceding operation, the robot must exchange the current end-effector for a different one at its Tool Box. Operational tasks that are performed using the three kinds of parts stored in Part Boxes 1, 2 and 3, are denoted X , Y and Z , respectively, in Table 2.2. For example, robot 2 may change its end-effector to handle the parts located in Part Box 1. Next, the robot picks up a part from this box and then assembles it to a target object at the Assembly Table. This series of processes can be described using the following task combinations: tool change X_T^2 , part acquisition X_P^2 , and assembly X_A^2 . Note that the RCMS automatically performs tool changes, and the processing schedule for these detailed tasks is constructed under the constraints described in Section 2.3.2.

Table 2.3 shows standard processing times for changing an end-effector, picking up a part, and for assembly. The standard times and expected robot movement times depend on the type of part being manipulated.

Table 2.4 shows the sizes of the rectangular dummy components used during calculations. The use of various sizes improves the optimization results, compared with those obtained when

Table 2.2. Layout components.

Component name	ID	Size (Length, Width) [mm]	Task
Robot 1	1	(255,255)	-
Robot 2	2	(255,255)	-
Assembly Table	3	(200,200)	$X_A^1, Y_A^1, Z_A^1, X_A^2, Y_A^2, Z_A^2$
Tool Box 1	4	(200,100)	X_T^1, Y_T^1, Z_T^1
Tool Box 2	5	(200,100)	X_T^2, Y_T^2, Z_T^2
Part Box 1	6	(150,100)	X_P^1, X_P^2
Part Box 2	7	(150,100)	Y_P^1, Y_P^2
Part Box 3	8	(150,100)	Z_P^1, Z_P^2

Table 2.3. Standard processing times.

Part	Time (s)		
	Tool change t_T	Pick-up t_P	Assembly t_A
X	3	4	2
Y	3	3	3
Z	1	2	3

a uniform dummy component size is used [32]. Given a variety of dummy component sizes, the optimization algorithm can make specific and appropriate selections that enhance robot manipulability. Concerning the genetic algorithm parameters, the population size, crossover rate, and mutation rate were set to 100, 0.8, and $1/2M$, respectively, where M is the number of components including dummy components. The genetic algorithm was iterated until the 20,000th generation was reached.

Table 2.4. Dummy component characteristics (Case 1).

ID	Size (Width, Depth) [mm]	Quantity
7, 8	(120,130)	2
9, 10	(130,140)	2
11, 12	(140,150)	2
13, 14	(150,160)	2
15, 16	(160,170)	2

The following computational experiments were run on a Dell Workstation with 32GB RAM and dual Intel Xeon E5-2630 CPUs ($2.30\text{GHz} \times 2$). The algorithm was implemented in MATLAB.

2.6.2 Two-objective Optimization Problem

First, the proposed method was applied to a two-objective optimization problem to verify its effectiveness in a simple problem, since performance evaluation is easier when the distribution of non-dominated solutions is planar. This example problem considers only two objective functions, operation time T and manipulability W , with layout area S ignored.

The solution distribution for this problem is shown in Fig. 2.11. Each point in Fig. 2.11 indicates an obtained non-dominated solution. Since the objectives of this optimization problem are minimization of operation time T and maximization of manipulability W , points closer to the upper-left corner represent better solutions in this figure. The distribution of the obtained solutions clearly shows the trade-off relationship between operation time T and manipulability W . That is, larger manipulability values are correlated with longer operation times, and short operation times can only be achieved when manipulability is relatively low.

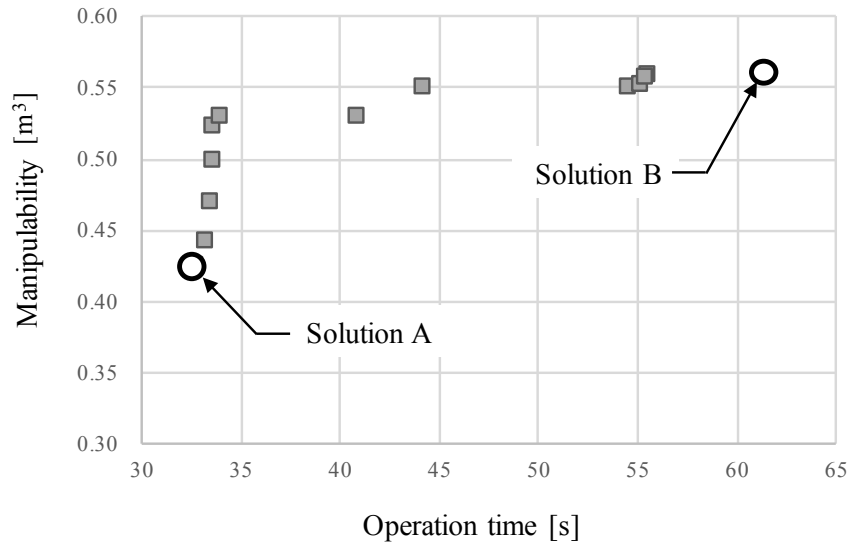


Figure 2.11. Distribution of non-dominated solutions.

Solution A in Fig. 2.11 is a layout design solution that has a superior operation time, but inferior manipulability. The component layout of solution A is shown in Fig. 2.12 and the corresponding task schedule is shown in Fig. 2.13. The numbers in the rectangles in Fig. 2.12 represent the component IDs listed in Table 2.2 and the gray rectangles indicate the dummy components listed in Table 2.4. The advantageous features provided by solution A are that tasks are well allocated to both robots, and certain components that are used, especially IDs 3 (Assembly Table), 6, 7 and 8 (Part Boxes 1–3) are in locations that both robots can access quickly, without risk of collision. These aspects of the layout help to shorten the operation time because tasks can be conducted in parallel, and the time the two robots spend moving between the components is shortened. However,

since components 4, 5 and 8 are too close to the robots, the manipulability becomes low when robots must access these components.

Solution B is the best solution with respect to manipulability, but poorest in terms of operation time. The component layout for solution B is shown in Fig. 2.14 and the corresponding task schedule is shown in Fig. 2.15. In this solution, Robot 2 alone conducts all operational tasks; Robot 1 remains idle. Several components, IDs 3, 5, 6, 7 and 8, are located far from Robot 2, which enhances manipulability at these locations, at the expense of operation time.

Figures 2.11, 2.12 and 2.14 illustrate that the use of dummy components in the proposed method provides layout configurations that incorporate adequate spacing between components, while ensuring that collisions are avoided during assembly operations. Additionally, these layout solutions demonstrate that the proposed method can provide widely-spread non-dominated solution sets that include a number of reasonable solutions, as the result of a single computation.

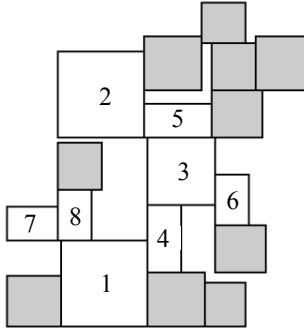


Figure 2.12. Layout of solution A.

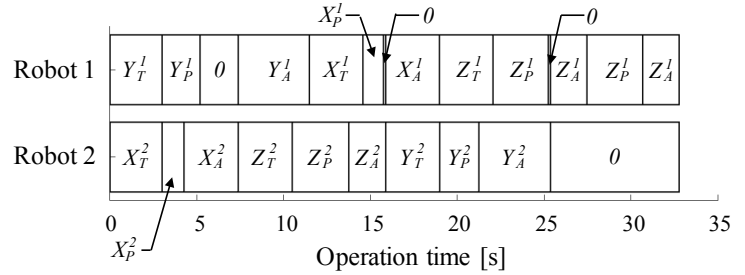


Figure 2.13. Gantt chart of solution A.

2.6.3 Parameter Calibration for Dummy Components

In this section, we discuss how appropriate parameters pertaining to dummy components are set to facilitate solving the layout problem. The provision of adequate spacing between components is very important in RCMS layout design problems, since robot manipulation requires certain minimal distances between task points to allow complex assembly tasks to be conducted at these points. We therefore calibrate the parameters for the sizes and numbers of dummy components, which strongly influence inter-component spacing.

Five cases that include different sizes and numbers of dummy components are considered as candidates for the parameters, as shown in Table 2.4, Table 2.5, and Table 2.6. The results for each case are compared in figures and quantitative metrics for the hypervolume indicator [52][53] are given. Larger values of the hypervolume indicator imply that better non-dominated solutions are obtained. Additionally, in this problem, the hypervolume indicator is calculated between operation

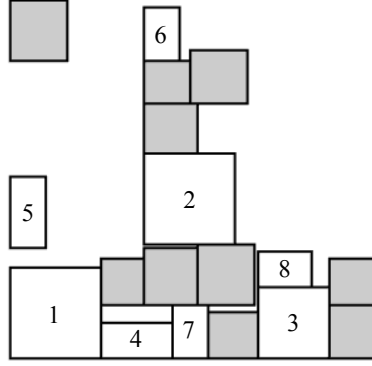


Figure 2.14. Layout of solution B.

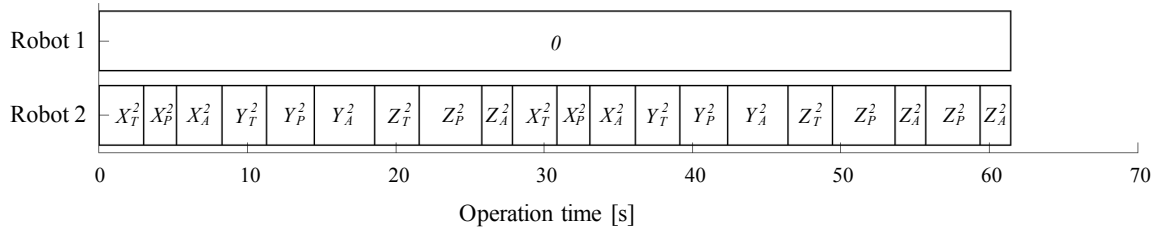


Figure 2.15. Gantt chart of solution B.

time $\mathbf{T}$ and an inverse value of manipulability $\mathbf{W}$, since a hypervolume is defined in a multi-objective minimization problem.

Table 2.5. Dummy component characteristics (Case 2 and Case 3).

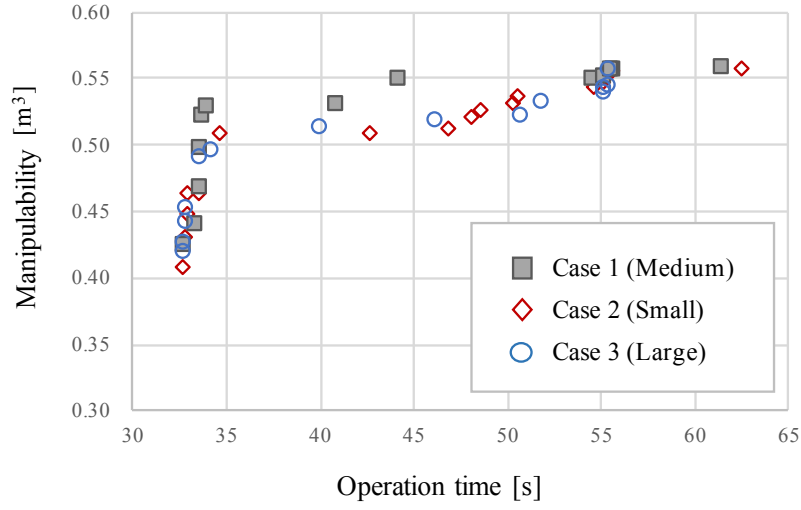
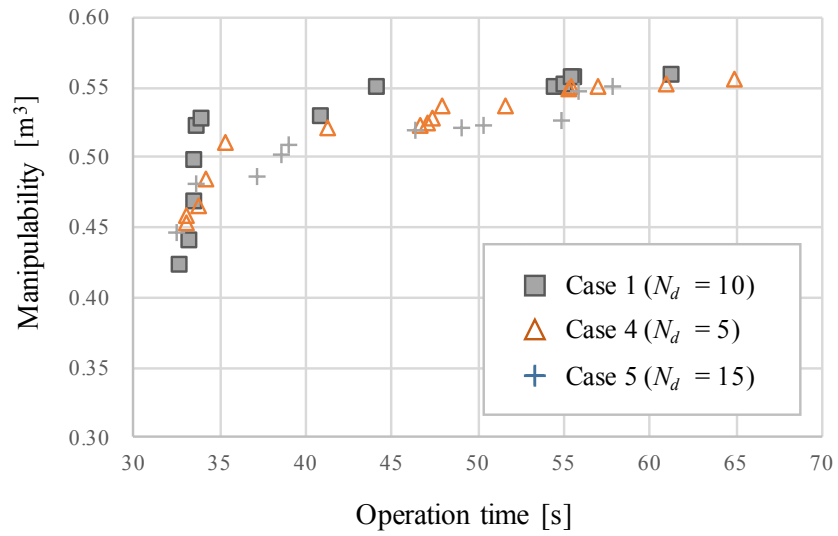
ID	Case 2: Small dummies		Case 3: Large dummies	
	Size (Width, Depth) [mm]	Quantity	Size (Width, Depth) [mm]	Quantity
7, 8	(5,10)	2	(150,160)	2
9, 10	(10,15)	2	(170,180)	2
11, 12	(15,20)	2	(190,200)	2
13, 14	(20,25)	2	(210,220)	2
15, 16	(25,30)	2	(230,240)	2

Fig. 2.16 shows non-dominated solutions for the three cases corresponding to the different sizes of dummy components listed in Table 2.4 and Table 2.5. Fig. 2.17 shows the obtained non-dominated solutions when the number of dummy components N_d is set to the values listed in Table 2.4 and Table 2.6. These figures indicate that the non-dominated solutions obtained in Case 1 have higher manipulability for short operation times than those of the other four cases.

Next, Table 2.7 offers a comparison of the hypervolume indicator values obtained using dummy components for each case, and reveals that the hypervolume indicator value for Case 1 is the largest of the five candidates. Based on these results, we used the Case 1 values shown in Table 2.4

Table 2.6. Dummy component characteristics (Case 4 and 5).

ID	Size (Width, Depth) [mm]	Quantity (Case 4)	Quantity (Case 5)
7, 8	(120,130)	1	3
9, 10	(130,140)	1	3
11, 12	(140,150)	1	3
13, 14	(150,160)	1	3
15, 16	(160,170)	1	3

**Figure 2.16.** Comparison of solutions for different dummy component sizes.**Figure 2.17.** Comparison of solutions when different numbers of dummy components are used.

to set the sizes and numbers of dummy components in the following numerical example for the three-objective optimization problem.

Table 2.7. Comparison of hypervolume indicator

Case of dummy component	Hypervolume indicator
Case 1	0.7886
Case 2	0.7238
Case 3	0.7123
Case 4	0.7312
Case 5	0.6954

2.6.4 Three-objective Optimization Problem

Finally, the proposed method is applied to a three-objective optimization problem considering operation time T , manipulability W , and layout area S . This example uses the same settings as those in the two-objective optimization problem discussed above for the robot configuration, number and size of rectangular dummy components, and genetic algorithm parameters.

The non-dominated solutions obtained using the proposed method for this problem are shown in Fig. 2.18. Since there are three objective functions, the solutions are distributed in a three-dimensional objective function space. In this problem, the trade-off relationship between layout area and manipulability is especially significant, as illustrated in Fig. 2.19. Four representative layout solutions, labeled C, D, E, and F, were selected from the obtained non-dominated solutions.

Solution C provides the minimum cycle time, D the maximum manipulability, E the minimum layout area, and F is a compromise solution for the three objectives. The layouts and task schedules for these solutions are shown in Fig. 2.20 through Fig. 2.27.

In solution C, the two robots perform operational tasks by turns to minimize the cycle time. Comparing solutions A and C, both have almost the same operation time, but solution C has a smaller layout area since it was generated with the goal of minimizing layout area S . Solutions B and D have similar features, with all operational tasks allocated to Robot 2. These two extreme solutions ignore minimization of the cycle time and minimization of layout area in order to maximize manipulability. In contrast, solution E, which achieves the minimum layout area, demonstrates that both robots can be utilized to perform operational tasks despite of the extreme nature of this solution. This is because appropriate task scheduling enables the avoidance of collisions between robots and components, and the operation time is reduced by using both robots rather than just one.

Figure 2.26 and Fig. 2.27 show the layout and Gantt chart for the compromise solution F, respectively, which provides reasonable performance values for all three criteria. These figures

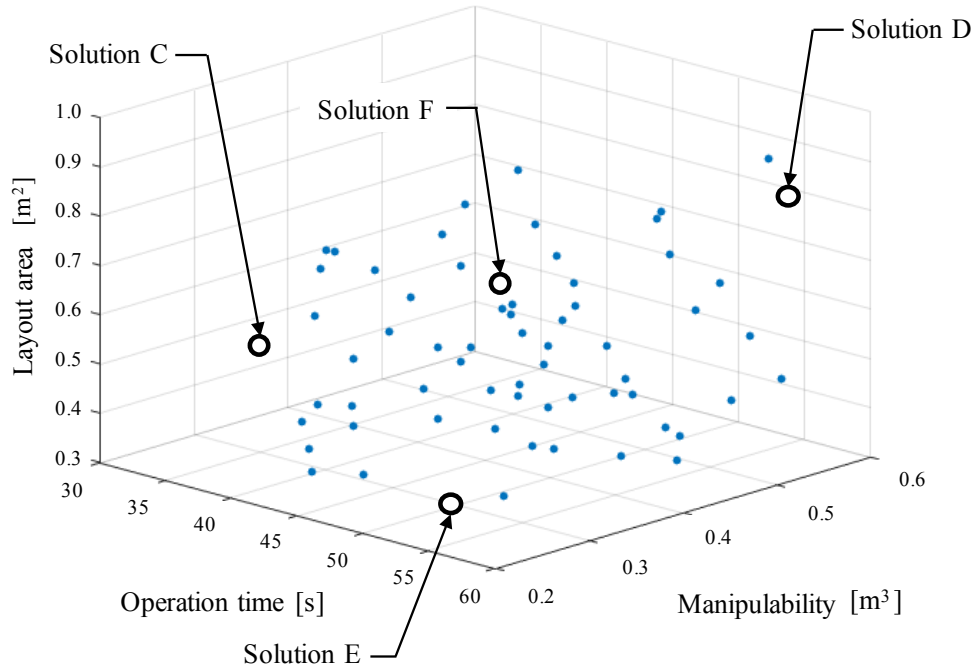


Figure 2.18. Distribution of non-dominated solutions.

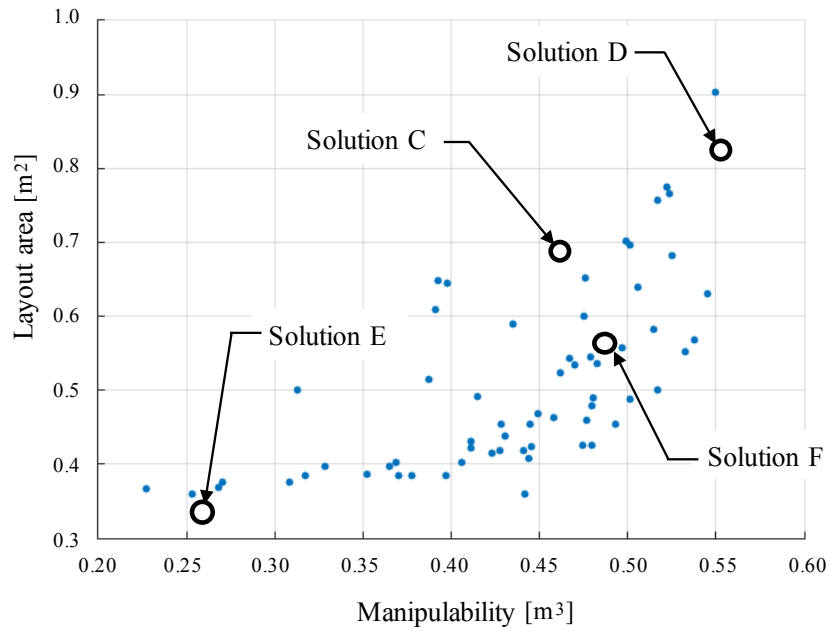


Figure 2.19. Non-dominated solutions mapped to the Layout area-Manipulability plane.

also imply the existence of a trade-off relationship between layout area and manipulability. That is, a smaller layout area reduces manipulability since the robot must perform its tasks at locations close to the robot where the manipulability is very low. Thus, an adequate layout area is necessary

to obtain sufficient manipulability. These figures illustrate that the proposed method can provide reasonable solutions for RCMS layout designs.

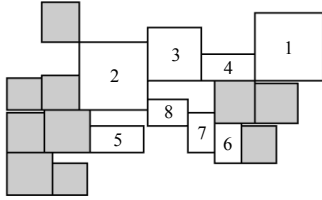


Figure 2.20. Layout of solution C.

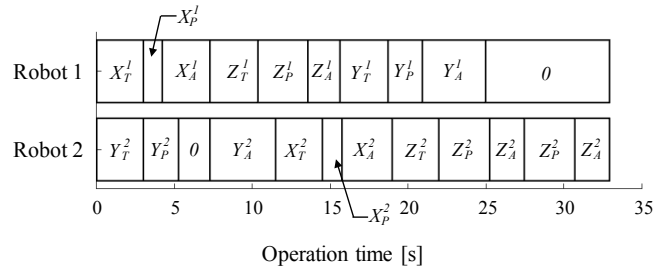


Figure 2.21. Gantt chart of solution C.

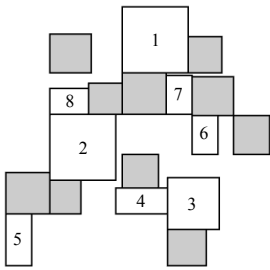


Figure 2.22. Layout of solution D.

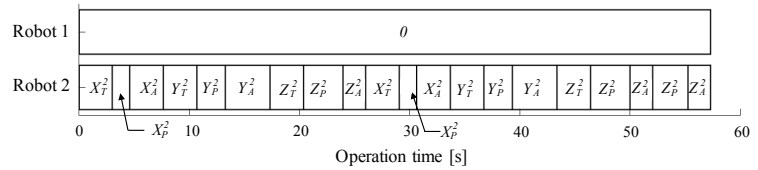


Figure 2.23. Gantt chart of solution D.

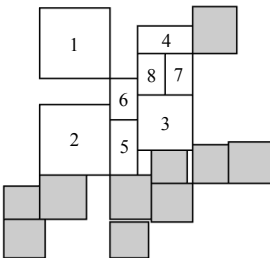


Figure 2.24. Layout of solution E.

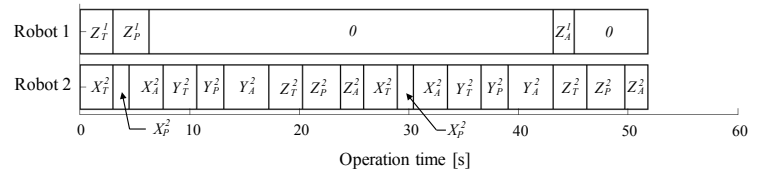


Figure 2.25. Gantt chart of solution E.

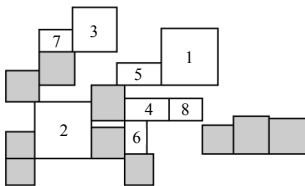


Figure 2.26. Layout of solution F.

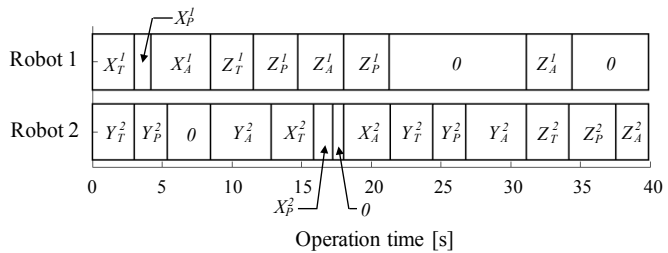


Figure 2.27. Gantt chart of solution F.

2.7 Conclusion

This paper proposed a multiobjective optimization technique to assist the layout design for multi-robot cellular manufacturing systems. A new coding technique was proposed, which represents both task allocation and component positions. Integer variables were used to allocate assembly tasks for multiple robots, and sequence-pair and binary variables were used to represent the location and orientation of components, respectively. To improve the positioning of components used during assembly tasks, the concept of dummy components was introduced to represent required spaces between the components. Design criteria for RCMS layouts were clarified and a layout design problem was formulated as a three-objective optimization problem considering collision constraints. A multiobjective genetic algorithm (MOGA) was used to obtain comprehensive non-dominated solutions for the RCMS layout designs. For the crossover process in the MOGA, a hybrid approach using a Placement-based Partially Exchanging Crossover (PPEX) scheme and a uniform crossover scheme was used, to facilitate coding. The proposed method was applied to two- and three-objective optimization problems, and the obtained solutions demonstrated its effectiveness.

Surrogate-assisted Scenario-generation Method for Simulation-based Stochastic Programming Problems

3.1 Introduction

Industries frequently face situations requiring decisions under uncertainty, such as production planning and supply chain management, where future demand is uncertain. In these cases, stochastic programming is an effective optimization approach to find a robust optimal solution by minimizing objective values under uncertainty. Discrete simulators are also often used in pragmatic decision-making to consider analytically intractable factors in real-world systems. This paper discusses stochastic programming formulated on the basis of a simulator, referred to as a *simulation-based stochastic programming problem* (SBSP). Simulation-based optimization typically relies on computationally intensive black-box optimization methods due to the lack of efficient solving techniques. SBSPs also requires iterative simulation runs proportional to the number of scenarios to evaluate uncertainty for each solution candidate, significantly increasing computational time. Consequently, scenario generation, which selects minimal and essential scenarios, is crucial for solving SBSPs.

For scenario generation, distribution-driven approaches, such as Monte Carlo sampling and clustering methods, are widely used to approximate the probability distribution with a finite number of scenarios. Problem-driven approaches have been proposed recently, leveraging information from the target optimization problem to identify scenarios that significantly impact the decision outcome. However, it remains challenging to apply these methods to SBSPs. Distribution-driven methods often require a large number of scenarios to achieve high-quality solutions. Conversely, most problem-driven methods rely on solving subproblems of the target problem, which is difficult to do quickly in SBSPs. Given these challenges, we propose a surrogate-assisted scenario-generation method called *Inferred Cost-Space Scenario Clustering* (ICSSC) tailored for SBSPs.

This study focuses on practical situations in which SBSPs with similar configurations are solved iteratively, such as inventory placement planning where a logistics network is constant, but shipping order sets change daily. In such situations, using past simulation results to solve the

upcoming SBSPP is reasonable since these results can share a similar tendency characterized by the consistent part of the optimization model. On the basis of this offline dataset, a *surrogate model* is trained to infer simulation evaluations much faster than the original simulator. ICSSC selects the representative scenario set using k -medoids clustering based on a new cost-space distance metric, *cost-space Euclidean distance*, evaluated by the pre-trained surrogate model.

The contributions of this paper are as follows:

- Proposal of the surrogate-assisted scenario-generation method, ICSSC, applicable to SBSPP.
- Selection of the appropriate surrogate model for SBSPP.
- Evaluation of ICSSC's effectiveness in solving SBSPPs compared with conventional methods.

In Section II, we introduce the background of this study. We describe SBSPPs and scenario generation in Section III. In Sections IV, V, and VI, we discuss the performance evaluation of ICSSC through Markowitz portfolio optimization, stochastic server location, and inventory placement optimization problems, respectively. We conclude in Section VII with a summary and future work.

3.2 Background

3.2.1 Stochastic Programming

In *stochastic programming problem* (SPP), some or all parameters of the optimization problem are uncertain but follow given probability distributions. The goal is to find the decision variables that optimize the expected performance measure across all possible scenarios, which are the realizations of the uncertain parameters. A general formulation of this problem is as follows [54], [55]:

$$\min_{x \in X} \int_{\Xi} F(x, \xi) P(d\xi), \quad (3.1)$$

where the decision variables x are defined on X , which is a closed subset of $\mathbb{R}^n$. The scenario ξ involves random parameters defined on a given probability distribution Ξ , which is a closed subset of $\mathbb{R}^d$, $F(x, \xi)$ is the objective function, and $P(*)$ represents the given probability measure modeling the uncertainty in the problem.

In practice, closed-form solutions for Equation 3.1 are rarely available. The most widely used technique to solve real-world SPPs is *sample average approximation*, which transforms the problem into the deterministic equivalent problem using a set of discrete scenarios $S = \{\xi_1, \dots, \xi_N\}$ and their positive probabilities $\{p_1, \dots, p_N\}$ for approximating the integral by a finite sum [56]. Equation 3.1 can be reformulated with a number of the original scenarios N as follows:

[Original problem: Q]

$$\min_{x \in X} \left\{ f(x) := \sum_{i=1}^N p_i F(x, \xi_i) \right\}, \quad (3.2)$$

where ξ_i is the i -th scenario defined on $\Xi \in \mathbb{R}^d$, and p_i is the probability of the i -th scenario. Generally, N must be a large positive integer to obtain a statically valid solution for a multi-dimensional probability distribution, thus this problem becomes computationally intractable in practical situations.

3.2.2 Simulation-based Optimization

Many real-world systems, such as supply chain networks, are too complex to be described with analytical models such as mixed-integer linear programming (MILP) models, especially when there are uncertain elements and interdependent parts. Simulation models are widely used in such cases since they can directly handle non-linear behaviors and complex interactions among their components more effectively. Thus, *simulation-based optimization* (SBO), which integrates simulation with optimization techniques to solve complex problems, is widely used to support real-world decision-making. With SBO approaches, solutions are evaluated by the simulation to understand the objective function's landscape, and black-box optimization methods, such as genetic algorithms (GA) and covariance matrix adaptation evolution strategy (CMA-ES) [57], are widely used as search algorithms [58].

When evaluations of the objective or constraint functions involve time- or resource-intensive simulations, a *surrogate model* (*metamodel*) is used to approximate the objective and/or constraint functions and reduce the computational cost [59]. A surrogate model emulates the behavior of a complex simulator at negligible cost per run [60]. Various machine learning (ML) models, such as linear regression (LR) [61], Gaussian process (GP) [62], gradient boosting trees (GBT) [63], and deep neural networks (DNN) [64], are used as surrogate models.

3.2.3 Simulation-based Stochastic Programming Problem

SBO approaches are often used for solving optimization problems including uncertain information. We refer to this type of SBO as a simulation-based stochastic programming problem (SBSPP). For example, Baharom and Hamzah [65] developed an inventory simulation model of an electronics company, determining the optimal reorder policy with the lowest average total inventory cost using scenarios of uncertain demand and lead times obtained with an Monte Carlo method. Pacheco et al. [66] used discrete event simulation to model a supply chain subject to demand uncertainty and proposed a reorder point update procedure for order-up-to-level policies in continuous review systems.

Note that most simulators cannot handle stochastic distributions as input. Instead, an Monte Carlo method is frequently used to generate a set of scenarios to approximate the original distribution. However, to solve SBSPPs, repeated simulation runs are required to evaluate the performance metrics for each parameter and scenario combination. This often results in excessive computational intensity, making it difficult to obtain a trustworthy solution. It is crucial to carefully select a set of necessary and sufficient scenarios to derive a robust solution within a practical time. Therefore, we developed a scenario-generation method that obtains a set containing as few but critical scenarios as possible for quickly solving SBSPPs.

3.3 Scenario Generation

3.3.1 Problem Statement and Method Classification

Scenario generation or *scenario reduction* is a process of creating scenarios as a concise, discrete realization of a given probability distribution or a scenario set with as small a number of scenarios as possible [67]. Scenario generation is crucial in stochastic programming because the quality of its solution depends on approximating the underlying uncertainty. In scenario generation, we consider a problem $\tilde{Q}$ that approximates the original problem Q , defined as Equation 3.2, using a number of *representative scenarios* $K \ll N$ as follows:

[Approximated problem: $\tilde{Q}$]

$$\min_{x \in X} \left\{ \tilde{f}(x) := \sum_{k=1}^K \tilde{p}_k F(x, \tilde{\xi}_k) \right\}, \quad (3.3)$$

where the set of scenarios is denoted as $\tilde{S} = \{\tilde{\xi}_1, \dots, \tilde{\xi}_K\}$, their probabilities as $\tilde{p} = \{\tilde{p}_1, \dots, \tilde{p}_K\}$, and the optimal solution as $\tilde{x}^*$. Note that an upper tilde ($\sim$) indicates the values related to the representative scenarios. Scenario generation aims to find $\tilde{S}$ and $\tilde{p}$ to minimize the *implementation error* (i.e., approximation error) [68], [69], defined as

$$f(\tilde{x}^*) - v^* = \sum_{i=1}^N p_i \{F(\tilde{x}^*, \xi_i) - F(x^*, \xi_i)\} \geq 0, \quad (3.4)$$

where x^* is the optimal solution of Q , and v^* is the optimal objective value. The implementation error is minimized when the solution obtained with K scenarios $\tilde{x}^*$ is similar to the true solution x^* . Thus, in scenario generation, sufficient scenarios should be generated to capture the impact of uncertainty while the scenario set size should be small to solve $\tilde{Q}$ quickly.

Scenario-generation methods can be classified into *distribution-driven* and *problem-driven* [70].

The key differences lie in the generation of scenarios that reflect the impact of uncertainty in target problems. Distribution-driven methods focus on generating a subset of scenarios to approximate the original distribution or scenario set statistically. These methods do not take into account the target problem’s performance metrics and model when selecting a scenario set. Problem-driven methods generate a scenario set by determining how changes in uncertain parameters affect decision outcomes, often using problem information to identify critical scenarios.

3.3.2 Distribution-driven Methods

Distribution-driven methods generally provide a scenario set to approximate the given distribution statistically and are widely used in academic and practical fields. Major distribution-driven methods involve sampling a finite number of scenarios from the original distribution, such as Monte Carlo methods (e.g., Kleywegt et al. [71]), quasi-Monte Carlo methods (e.g., Heitsch et al. [72]), and sparse grid quadrature (e.g., Chen et al. [54]). Other methods involve minimizing a metric to evaluate the distance between the original probability distributions and generated scenario set, such as the minimization of the nested distance built on the Wasserstein distance [73], moment matching [74], and various clustering algorithms, e.g., k -means [75] and k -medoids [76].

Löhndorf [77] conducted an empirical analysis of widely-used scenario-generation methods, such as quasi-Monte Carlo, moment matching, and competitive learning, using the newsvendor problem with multiple distributions. Their results indicate that no single scenario-generation method is universally superior and that the Wasserstein distance as a probability metric is ineffective for choosing scenarios in higher dimensions. Jiang et al. [78] explored a method for bundling scenarios in a progressive hedging heuristic applied to stochastic service network design. They compared scenario bundling with hard clustering methods such as k -means, soft clustering methods, such as Fuzzy C-Means and Gaussian Mixture Model (GMM), and the cover method. Their experimental results indicate that GMM-based progressive hedging heuristic yielded the best performance among all methods they investigated. Escudero et al. [79] proposed a multistage scenario cluster Lagrangean decomposition to obtain strong lower bounds on large-scale multistage mixed binary SPPs by using the scenario tree structure, referred to as a “break stage.” Note that distribution-driven methods can generally generate a vast number of scenarios, many of which may have a negligible impact on solution, leading to extended computational times to solve SPPs.

3.3.3 Problem-driven Methods

To overcome these challenges, problem-driven methods focus on the decision problem and the impact of uncertainty on the decisions [70], [80], [81]. These methods often use the performance measures of the target optimization problem to identify essential scenarios that evaluate the so-

lution’s robustness. This approach can reduce the number of scenarios handled in optimization while maintaining the accuracy of uncertainty evaluation by excluding scenarios that do not affect decision outcomes. Problem-driven methods generate different scenario sets for different problems, even if the probability distributions are the same. Thus, problem-driven methods can obtain more effective and smaller scenario sets from a logical perspective than distribution-driven methods.

Henrion and Römisich [55] formulated an optimal scenario-generation problem for linear two-stage SPPs with chance constraints based on minimal information distances and applied it to the news-vendor problem with random demand. Fairbrother et al. [82] proposed the aggregation sampling method and aggregation reduction method for SPPs in which a tail risk measure, such as conditional value-at-risk, is applied to a loss function, and investigated the effectiveness in the portfolio selection problem. Ryan et al. [83] proposed an optimization-driven scenario grouping method, which leads to stronger non-anticipative relaxation bounds by estimating bound improvement with single-scenario subproblem solutions. For mixed-integer SPPs, Keutchan et al. [68] introduced a cost-space scenario clustering (CSSC) method that forms clusters based on the distance in the cost space defined as the objective values of single-scenario subproblems.

In recent years, ML technologies have been combined with problem-driven scenario-generation methods to generate more informative scenarios [70]. Dumouchelle et al. [84] proposed Neur2SP, a fast solution method for mixed-integer linear/nonlinear SPPs by replacing the expected value of the objective function with a rectified linear unity (ReLU) neural network that can be expressed as an MILP model. Wu et al. [85] studied CVAE-SIPA that uses a conditional variational autoencoder (CVAE) to learn scenario embeddings through semi-supervised learning for the objective function and execute clustering for scenario reduction.

However, it is challenging to apply these problem-driven methods to SBSPPs directly. Most of these methods, such as those by Henrion and Römisich [55] and Keutchan et al. [68], are designed for mixed-integer/quadratic SPPs, and leverage the characteristic that these single-scenario subproblems can be solved immediately. In contrast, it is difficult to solve the single-scenario subproblem of an SBSPP within a short time frame because iterative simulation runs are required. While the ML-based methods by Dumouchelle et al. [84] and Wu et al. [85] can be applied to SBSPPs, both require a large dataset consisting of solutions to the single-scenario subproblems, making the cost of preparing training data prohibitively high for practical use. For example, Wu et al. [85] reported that the amount of labeled data obtained from the subproblem solutions can be reduced to 1% of the entire dataset in supervised learning of CVAE-SIPA. However, in SBSPPs, the cost of preparing the labeled training data is still significant since it requires thousands of iterative simulation runs to solve one subproblem. Given this background, it is crucial to develop an effective problem-driven scenario-generation method for SBSPPs that allows for practical computation times in preparing a training dataset and solving the target optimization problem.

3.3.4 Cost-space Scenario Clustering Method

We focus on CSSC [68], which is widely applicable to SPPs regardless of the modeling scheme (analytical or simulation-based) and probability distributions, except for computational costs to solve single-scenario subproblems. CSSC does not evaluate the distance between scenarios in the *space of random outcomes* ($\mathbb{R}^d$), such as the distance metric $d(\xi_i, \xi_j)$ but instead in the *space of cost values* ($\mathbb{R}$), such as $F(\cdot, \xi_i)$ and $F(\cdot, \xi_j)$. For example, if two scenarios ξ_i and ξ_j are very different in the space of random outcomes ($d(\xi_i, \xi_j) \gg 0$) but lead to the same cost value ($F(\cdot, \xi_i) = F(\cdot, \xi_j)$), they can be grouped into the same cluster without changing the solution. Then, K scenarios are produced as the centroids of scenario clusters estimated in the cost space.

The original scenario set $S = \{\xi_1, \dots, \xi_N\}$ is first randomly selected from a probability distribution Ξ . We denote the set $\{1, \dots, N\}$ as $\mathcal{N}$ and $\{1, \dots, K\}$ as $\mathcal{K}$. A single-scenario subproblem Q_i containing only scenario ξ_i is formulated as

[Single-scenario subproblem: Q_i]

$$\min_{x \in X} F(x, \xi_i), \quad \forall i \in \mathcal{N}. \quad (3.5)$$

The opportunity cost $V_{i,j}$ defined by the following equation is then calculated for all combinations of i and j [86]:

$$V_{i,j} = F(x_i^*, \xi_j), \quad \forall i, j \in \mathcal{N}, \quad (3.6)$$

where x_i^* is a solution of Q_i . Scenario clustering determines a partition of the scenario index set $\mathcal{N}$ into K clusters $\{C_k \mid k \in \mathcal{K}\}$ and their representative scenario indexes $\{r_k \mid r_k \in C_k, k \in \mathcal{K}\}$ by minimizing the *clustering discrepancy* as

$$\min_{\{(C_k, r_k) \mid k \in \mathcal{K}\}} \sum_{k \in \mathcal{K}} \tilde{p}_k \left| V_{r_k, r_k} - \frac{1}{|C_k|} \sum_{j \in C_k} V_{r_k, j} \right|, \quad (3.7)$$

where $|C_k|$ is the number of scenarios in C_k and $\tilde{p}_k = |C_k|/N$. This scenario clustering is called the *mixed-integer partitioning problem* (MIPP). CSSC obtains the set of the K representative scenarios $\tilde{\xi}_k = \xi_{r_k} (k \in \mathcal{K})$ that approximately minimize the implementation error in Equation 3.4 by solving MIPP with a general MILP solver.

CSSC outperforms other distribution-driven methods on two stochastic MILP problems: network design problem and facility location problem [68]. It is also superior to other problem-driven methods because no underlying probability distribution is needed, and no arbitrary metric in the space of random outcomes must be specified. However, CSSC has the following two issues when applying it to SBSPPs. First, it requires many computational resources to solve single-scenario

subproblems to calculate opportunity costs. Therefore, it can only be applied when subproblems can be solved quickly, such as linear programming problems. Additionally, the number of the original scenarios N and the representative ones K must be small enough since MIPP becomes intractable when N and/or K is large. This limits performance for the high-dimensional scenario space because many scenarios are necessary to approximate the probability distribution.

We discuss the following two challenges:

Challenge 1: Quickly solve SBSPP single-scenario subproblems for calculating opportunity costs,

Challenge 2: Conduct scalable scenario clustering regardless of the number of original and representative scenarios (N, K) .

Our method expands CSSC to overcome these challenges. We first use a surrogate model to quickly approximate the objective function value without simulation. We then use a scenario-clustering method that combines a new distance metric between scenarios in the cost space and an unsupervised clustering algorithm, which is more scalable than MIPP.

3.4 Inferred Cost-space Scenario Clustering

3.4.1 Overview of Proposed Method

The proposed method, *Inferred Cost-Space Scenario Clustering* (ICSSC), expands CSSC to overcome the two challenges mentioned above. ICSSC addresses the following constrained SPP:

$$\begin{aligned} \min_{\mathbf{x} \in \mathbf{X}} \quad & \sum_{k \in \mathcal{K}} \tilde{p}_k F(\mathbf{x}, \tilde{\xi}_k), \\ \text{s.t.} \quad & G_l(\mathbf{x}) \leq 0, \quad \forall l \in \mathcal{L}. \end{aligned} \tag{3.8}$$

where $F(\mathbf{x}, \xi)$ is the objective function implemented using a simulator, $G_l(\mathbf{x})$ is the l -th constraint, L is the number of constraints, and the set $\{1, \dots, L\}$ is denoted as $\mathcal{L}$.

The overall flow of ICSSC is shown in Figure 3.1 and consists of two phases: *pre-training* and *scenario-generation*. In the pre-training phase, a surrogate model $\hat{F}(\mathbf{x}, \xi)$, which infers the objective function $F(\mathbf{x}, \xi)$, is built using offline simulation datasets. The opportunity cost is then calculated with the surrogate model in the scenario-generation phase. The first challenge can be resolved by quickly inferring objective function values with a surrogate model instead of simulation. To resolve the second challenge, we use k -medoids clustering based on a new distance metric using the opportunity cost to find the representative scenarios instead of solving MIPP.

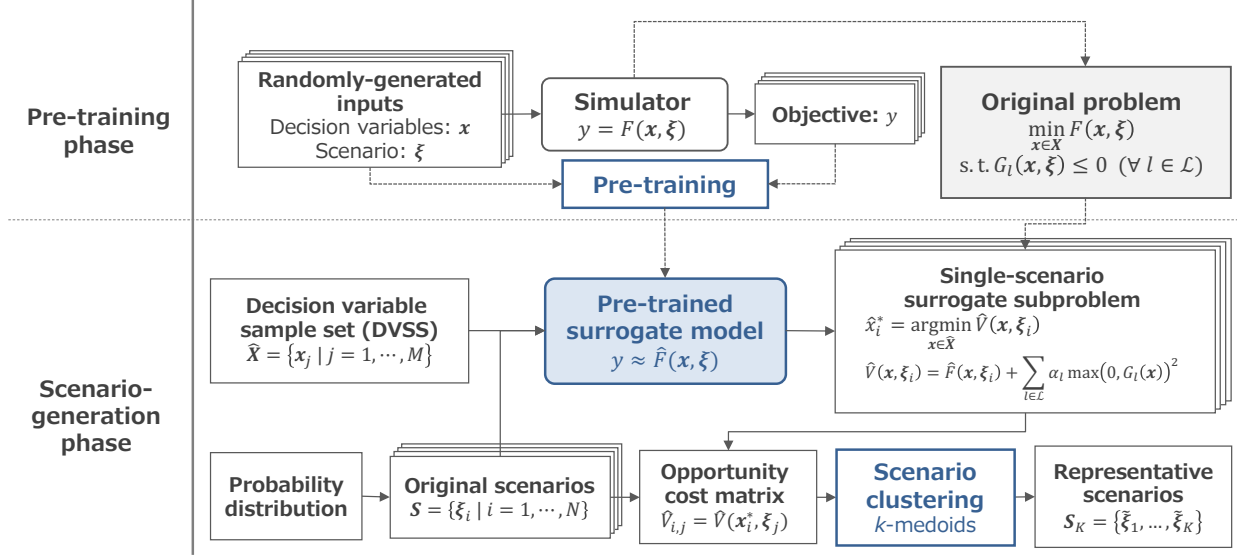


Figure 3.1. Flowchart of ICSSC. Surrogate model of objective function is trained in pre-training phase. In scenario-generation phase, K representative scenarios are selected by k -medoids clustering on basis of the opportunity cost obtained with surrogate model.

3.4.2 Mathematical Development

In this section, we derive the error bound to be minimized with ICSSC from the implementation error defined in Equation 3.4 on the basis of the discussion by Keutchan et al. [68]. Given that $\tilde{f}(\tilde{x}^*) \leq \tilde{f}(x^*)$ since $\tilde{x}^*$ is a minimum of $\tilde{f}$, and $f(x^*) \leq f(\tilde{x}^*)$ since x^* is a minimum of f , the implementation error can be bounded as

$$|f(\tilde{x}^*) - v^*| = |f(\tilde{x}^*) - f(x^*)| \quad (3.9)$$

$$= |f(\tilde{x}^*) - \tilde{f}(\tilde{x}^*) + \tilde{f}(\tilde{x}^*) - f(x^*)| \quad (3.10)$$

$$\leq |f(\tilde{x}^*) - \tilde{f}(\tilde{x}^*)| + |\tilde{f}(\tilde{x}^*) - f(x^*)| \quad (3.11)$$

$$\leq 2 \max_{x \in \{x^*, \tilde{x}^*\}} |\tilde{f}(x) - f(x)| \quad (3.12)$$

$$\leq 2 \max_{x \in \tilde{X}} |\tilde{f}(x) - f(x)| \quad (3.13)$$

where $\tilde{X} \subseteq X$ is any feasible set including the two optimal solutions of the original and approximate problems, $\{x^*, \tilde{x}^*\}$, both unknown in scenario generation. The $\tilde{X}$ has to be sufficiently large to ensure the validity of the bound while sufficiently small to guarantee its tightness. For detailed discussions of Equations 3.9 – 3.13, see Keutchan et al. [68]. Further expanding the right side

of Equation 3.13, we have

$$\max_{x \in \tilde{X}} |\tilde{f}(x) - f(x)| = \max_{x \in \tilde{X}} \left| \sum_{k \in \mathcal{K}} p_k F(x, \tilde{\xi}_k) - \frac{1}{N} \sum_{i \in \mathcal{N}} F(x, \xi_i) \right| \quad (3.14)$$

$$= \max_{x \in \tilde{X}} \left| \sum_{k \in \mathcal{K}} \frac{|C_k|}{N} F(x, \tilde{\xi}_k) - \frac{1}{N} \sum_{k \in \mathcal{K}} \sum_{i \in C_k} F(x, \xi_i) \right| \quad (3.15)$$

$$= \frac{1}{N} \max_{x \in \tilde{X}} \left| \sum_{k \in \mathcal{K}} \sum_{i \in C_k} (F(x, \tilde{\xi}_k) - F(x, \xi_i)) \right| \quad (3.16)$$

$$\leq \frac{1}{N} \sum_{k \in \mathcal{K}} \sum_{i \in C_k} \sup_{x \in \tilde{X}} |F(x, \tilde{\xi}_k) - F(x, \xi_i)| \quad (3.17)$$

Therefore, we formulate the following problem to approximately minimize the implementation error:

$$\min_{\{(C_k, \xi_k) | k \in \mathcal{K}\}} \sum_{k \in \mathcal{K}} \sum_{i \in C_k} \sup_{x \in \tilde{X}} |F(x, \xi_{r_k}) - F(x, \xi_i)|, \quad (3.18)$$

where r_k is an index of a representative scenario of cluster k and $\tilde{\xi}_k = \xi_{r_k}$. This problem can be considered as minimizing the sum of the intra-cluster distances in the cost space within each cluster and defined as

$$D(\xi_{r_k}, \xi_i) = \sup_{x \in \tilde{X}} |F(x, \xi_{r_k}) - F(x, \xi_i)|. \quad (3.19)$$

This metric indicates the supremum value of the cost difference between scenarios r_k and i .

As mentioned above, ICSSC uses a surrogate model to evaluate $F(x, \xi)$ instead of the simulator. However, the scenario distance in Equation 3.19 is highly sensitive to the prediction errors of the surrogate model since it is determined by a single x providing the supremum value. To mitigate the impact of prediction errors and approximately minimize the supremum value, we introduce the following distance metric based on the root mean squared error (Euclidean distance), as follows:

$$D(\xi_{r_k}, \xi_i) = \sqrt{\sum_{x \in \tilde{X}} (F(x, \xi_{r_k}) - F(x, \xi_i))^2}. \quad (3.20)$$

Since minimizing the squared error reduces larger differences more aggressively, the supremum value can be minimized approximately. We refer to this distance metric as *cost-space Euclidean distance* (CSED).

As Keutchan et al. [68] proposed, we consider a set $\tilde{X}$ composed of optimal solutions to all of the single-scenario subproblems, $\tilde{X} = \{x_j^* | j \in \mathcal{N}\}$. It leverages the findings by Hewitt et al. [86] that the subproblem solutions can be used to effectively search for commonalities between

scenarios. Equation 3.20 can then be rewritten as

$$D(\xi_{r_k}, \xi_i) = \sqrt{\sum_{j \in \mathcal{N}} \left(F(\mathbf{x}_j^*, \xi_{r_k}) - F(\mathbf{x}_j^*, \xi_i) \right)^2} \quad (3.21)$$

$$= \sqrt{\sum_{j \in \mathcal{N}} (V_{j,r_k} - V_{j,i})^2}. \quad (3.22)$$

Finally, ICSSC solves the following problem to approximately minimize the implementation error:

$$\min_{\{(C_k, \xi_k) | k \in \mathcal{K}\}} \sum_{k \in \mathcal{K}} \sum_{i \in C_k} \sqrt{\sum_{j \in \mathcal{N}} (V_{j,r_k} - V_{j,i})^2}. \quad (3.23)$$

3.4.3 Pre-training Phase

ICSSC uses a surrogate model $\hat{F}(\mathbf{x}, \xi)$ to estimate the objective function values $F(\mathbf{x}, \xi)$ without time-consuming simulation. Training an accurate surrogate model requires a large dataset with various simulation inputs and outputs. Such a large simulation dataset requires significant computational resources to run various simulation patterns. In practical situations, ensuring such a long computation time before solving the problem is difficult since the complete problem information is generally available just before solving it.

We thus focus on practical situations regarding data availability, where similar problems with partially different configurations are often solved iteratively, especially in periodic decision-making processes. Such an optimization problem can be decomposed into constant and uncertain factors. For example, a logistics network is rarely reconfigured in inventory placement planning because significant budget and effort are necessary. However, uncertain parts, such as shipping orders, can be changed for every decision-making situation. In such a situation, the offline dataset, like past simulation results, can be used effectively to train the surrogate model, the input features of which consist of the decision variables and parameters of the uncertain part since these results share a similar tendency characterized by the constant part [87]. We thus assume the following two conditions [88];

Assumption 1: Rather than assuming that the information of the uncertain part is only available just before solving the target problem, we assume that the distribution of the uncertain part obtained from offline information is always available.

Assumption 2: The constant factors in the model do not change enough to obtain simulation datasets containing various uncertain factors in certain periods.

In the pre-training phase, therefore, random input data, which are decision variables $\mathbf{x}$ and uncertain parameters ξ , are generated following the domains of decision variables $\mathbf{X} \in \mathbb{R}^n$ and

the distribution $\Xi \in \mathbb{R}^d$, uniformly at random. The objective function value $y = F(x, \xi)$ is then evaluated by running the simulation, and the training dataset $\mathcal{D} = \{(x, \xi, y)\}$ is obtained. Using $\mathcal{D}$, the surrogate model $\hat{F}(x, \xi)$ is trained to predict $F(x, \xi)$ before scenario generation. In the following discussion, we use an upper hat ($\hat{\cdot}$) to indicate the values and functions related to surrogate models.

3.4.4 Scenario-generation Phase

The scenario-generation phase is executed after determining the target optimization problem, as shown in Algorithm 3-1. To generate the representative scenarios by solving Equation 3.23, we must obtain the optimal solutions for each Q_i in Equation 3.5. However, when Q_i is modeled with a simulator, it is not easy to quickly find its solution x_k^* by solving Q_i directly. Therefore, we approximate x_i^* by solving the following *single-scenario surrogate subproblem* $\hat{Q}_i$, which is formulated by replacing the objective function with $\hat{F}(x, \xi_i)$ and relaxing the constraints as the penalty terms.

Algorithm 3-1 ICSSC scenario-generation phase

Input: Decision variable space X , Scenario space Ξ ,

Pre-trained surrogate model $\hat{F}(x, \xi)$, and constraint functions $G_l(x, \xi) \forall l \in \mathcal{L}$.

Output: Representative K scenarios $\tilde{S} = \{\tilde{\xi}_1, \dots, \tilde{\xi}_K\}$, and their probabilities $\tilde{p} = \{\tilde{p}_1, \dots, \tilde{p}_K\}$.

- 1: Generate N scenarios $S = \{\xi_1, \dots, \xi_N\}$ by sampling from Ξ uniformly at random.
 - 2: Generate the decision variable sample set $\hat{X} = \{x_1, \dots, x_M\}$ by Sobol sequence sampling from X .
 - 3: Calculate $\hat{V}(x, \xi) = \hat{F}(x, \xi) + \sum_{l \in \mathcal{L}} \alpha_l \max(0, G_l(x))^2$, for all $M \times N$ combinations of $x \in \hat{X}$ and $\xi \in S$.
 - 4: Obtain all subproblem solutions: $\hat{x}_i^* = \underset{x \in \hat{X}}{\operatorname{argmin}} \hat{V}(x, \xi_i)$.
 - 5: Calculate the cost-space Euclidean distance $D(\xi_i, \xi_j)$ for all $N \times N$ combinations of $\xi_i, \xi_j \in S$.
 - 6: Evaluate K scenario clusters by k -medoids clustering using $D(\xi_i, \xi_j)$ as a distance matrix.
 - 7: **return** K scenarios $\tilde{S} = \{\tilde{\xi}_1, \dots, \tilde{\xi}_K\}$, and their probabilities $\tilde{p} = \{\tilde{p}_1, \dots, \tilde{p}_K\}$
-

[Single-scenario surrogate subproblem: $\hat{Q}_i$]

$$\min_{x \in X} \hat{F}(x, \xi_i) + \sum_{l \in \mathcal{L}} \alpha_l \max(0, G_l(x))^2, \quad (3.24)$$

where α_l is a penalty coefficient for constraint G_l .

By solving Equation 3.24, we obtain the solution of the single-scenario surrogate subproblem $\hat{x}_i^*$, which is used instead of the true subproblem solution x_i^* . However, solving $\hat{Q}_i$ is still difficult, and the solution accuracy must be worse than the prediction accuracy of $\hat{F}(x, \xi_i)$. Therefore, $\hat{x}_i^*$ is searched within the *decision variable sample set* (DVSS) $\hat{X} = \{x_1, \dots, x_M\} \in X$ defined as the subset of X consisting of M possible decision variable samples obtained by Sobol sequence sampling. Using DVSS, $\hat{x}_i^*$ is obtained as follows:

$$\hat{x}_i^* = \underset{x \in \hat{X}}{\operatorname{argmin}} \hat{V}(x, \xi_i), \quad (3.25)$$

$$\text{where } \hat{V}(x, \xi) = \hat{F}(x, \xi) + \sum_{l \in \mathcal{L}} \alpha_l \max(0, G_l(x))^2, \quad (3.26)$$

$$\alpha_l = \max_{x \in \hat{X}_l^+, i \in \mathcal{N}} (\hat{F}_{\max} - \hat{F}(x, \xi_i)) / G_l(x)^2, \quad (3.27)$$

$$\hat{F}_{\max} = \max_{x \in \hat{X}, i \in \mathcal{N}} \hat{F}(x, \xi_i), \quad (3.28)$$

$$\hat{X}_l^+ = \{x \mid G_l(x) > 0, x \in \hat{X}\}. \quad (3.29)$$

Equation 3.26 defines the penalized objective function, and Equation 3.27 calculates the penalty coefficient α_l for constraint G_l . The coefficient is set so that $V(x, \xi)$ of the solution violating the constraint is always larger than that of the solution satisfying the constraint. Equation 3.28 calculates the maximum objective value $\hat{F}_{\max}$ among all solutions, Equation 3.29 defines the set of solutions violating each constraint $\hat{X}_l^+$, Equation 3.25 can be solved immediately by parallel inferences, then the subproblem solutions $\hat{X}^* = \{\hat{x}_1^*, \dots, \hat{x}_N^*\}$ are obtained. The opportunity cost is then approximately obtained as

$$V_{i,j} = \hat{V}(\hat{x}_i^*, \xi_j), \forall i, j \in \mathcal{N}. \quad (3.30)$$

Then, N scenarios are divided into K clusters on the basis of $\hat{V}_{i,j}$ by solving Equation 3.23 with the opportunity-cost obtained from Equation 3.30. This problem can be solved by k -medoids clustering, the distance matrix of which is defined as Equation 3.22. The k -medoids clustering is a common clustering algorithm related to k -means. However, instead of using the mean of the data points in a cluster to define the cluster center, it uses an actual data point from the cluster, called the “medoid”. This makes k -medoids more robust to noise and outliers compared to k -means [76]. By using k -medoids clustering, we can avoid the computational complexity of MIPP to select representative scenarios. Finally, K representative scenarios $\tilde{S} = \{\tilde{\xi}_1, \dots, \tilde{\xi}_K\}$ are obtained as the medoids of K clusters $\{C_1, \dots, C_K\}$, and their probabilities $\tilde{p} = \{\tilde{p}_1, \dots, \tilde{p}_K\}$ are calculated as $\tilde{p}_k = |C_k|/N$.

3.5 Portfolio Optimization Problem

In the first group of experiments, we investigated a portfolio optimization problem based on the Markowitz model to evaluate the solution accuracy of ICSSC compared with conventional scenario generation methods and an exact solution.

3.5.1 Problem Statement

The Markowitz portfolio optimization problem[89] is an extensively studied nonlinear SPP formulated as a quadratic programming (QP) problem. This problem takes into account a portfolio consisting of n assets, and the decision variables are the portfolio allocation vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$, where x_i is the proportion of the total investment in asset i . The objective function is defined as the portfolio variance, which is a measure of the risk of the portfolio as follows:

$$\min_{\mathbf{x} \in \mathbf{X}} \frac{1}{2} \mathbf{x}^T \mathbf{\Sigma} \mathbf{x}, \quad (3.31)$$

$$\text{s.t. } \boldsymbol{\mu}^T \mathbf{x} \geq R, \quad (3.32)$$

where $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is the covariance matrix, $\boldsymbol{\mu} \in \mathbb{R}^n$ is the expected return vector, R is the lower bound of the expected return, and $\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \sum_{i=1}^n x_i \leq 1, x_i \geq 0\}$ is the decision variable space. Equation 3.31 defines the portfolio variance. Equation 3.32 means that the expected return must be larger than R . Here, Equation 3.31 can be reformulated with K return scenarios as

$$\min_{\mathbf{x} \in \mathbf{X}} \sum_{k=1}^K p_k \mathbf{x}^T (\boldsymbol{\xi}_k - \boldsymbol{\mu})^T (\boldsymbol{\xi}_k - \boldsymbol{\mu}) \mathbf{x}, \quad (3.33)$$

where p_k is the probability of scenario k , and $\boldsymbol{\xi}_k \in \mathbb{R}^n$ is the random vector of asset returns, which follows the multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$, described as $\boldsymbol{\xi}_k \sim \text{Normal}(\boldsymbol{\mu}, \mathbf{\Sigma})$. In this study, $\boldsymbol{\mu}$ is given as a known parameter. From Equation 3.8, the objective and constraint functions are denoted as

$$F(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{x}^T (\boldsymbol{\xi}_k - \boldsymbol{\mu})^T (\boldsymbol{\xi}_k - \boldsymbol{\mu}) \mathbf{x}, \quad (3.34)$$

$$G_1(\mathbf{x}) = R - \boldsymbol{\mu}^T \mathbf{x}. \quad (3.35)$$

3.5.2 Experiment Settings

In this experiment, we considered a portfolio consisting of $n = 7$ assets (Elsevier, Fortis, Getronics, Heineken, Philips, Royal Dutch, Unilever), discussed in Engles et al. [90]. In [90], $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ were

given as follows:

$$\mu = \begin{bmatrix} 0.266 & 0.274 & 0.162 & 0.519 & 0.394 & 0.231 & 0.277 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} 0.345 & 0.150 & 0.183 & 0.088 & 0.186 & 0.090 & 0.095 \\ 0.150 & 0.399 & 0.204 & 0.107 & 0.236 & 0.130 & 0.127 \\ 0.183 & 0.204 & 1.754 & 0.075 & 0.325 & 0.110 & 0.091 \\ 0.088 & 0.107 & 0.075 & 0.243 & 0.096 & 0.064 & 0.086 \\ 0.186 & 0.236 & 0.325 & 0.096 & 0.734 & 0.147 & 0.114 \\ 0.090 & 0.130 & 0.110 & 0.064 & 0.147 & 0.221 & 0.093 \\ 0.095 & 0.127 & 0.091 & 0.086 & 0.114 & 0.093 & 0.219 \end{bmatrix}.$$

where R was set to 0.35. The QP solver could easily obtain the optimal solution for this problem, and the solution was $x^* = [0.113, 0.000, 0.007, 0.361, 0.004, 0.276, 0.276]$ with an objective value of $v^* = 0.126$. The computing environment was a Ryzen 3950X (3.70GHz, 32 threads) CPU, 128GB RAM, and NVIDIA RTX3080 GPU.

3.5.3 Surrogate Model Selection

We discuss the suitable surrogate model for the objective function in Equations 3.34, trained as a regression problem. In this experiment, DNN, GBT, GP, and LR were compared in terms of root mean square error (RMSE), symmetric mean absolute percent error (SMAPE), and coefficient of determination (R^2) evaluated by 4-fold cross-validation. The training dataset $\mathcal{D}$ consisted of 50000 evaluation results, the inputs of which are decision variables x generated uniformly at random and return scenarios ξ sampled from the multivariate normal distribution, $Normal(\mu, \Sigma)$. The DNN consisted of three fully connected layers with 200 neurons (n_d) followed by a ReLU activation function, a batch normalization layer, and a dropout layer, the dropout rate of which was set to 10% (r_d). These parameters were selected to achieve the highest validation accuracy through grid search within the following ranges: $n_d \in [50, 100, 200, 300]$ and $r_d \in [1, 5, 10, 20]$. This DNN was trained using the Adam optimizer [91] with a learning rate automatically determined by the one-cycle policy [92]. The number of GBT estimators was set to 1000, and GP was trained with 300 randomly selected samples to shorten the training time to be equivalent to other models.

Table 3.1 shows the prediction performances of each surrogate model. These results indicate that DNN obtained lower prediction errors than the other models, while GP and LR failed to predict the objective function. DNN's inference time (1.28×10^{-6} sec./sample) is fast enough and competitive with them. These results suggest that DNN is a suitable surrogate model for a nonlinear function regarding accuracy and inference speed. In the following experiment, DNN was used as

the surrogate model for ICSSC.

Table 3.1. Objective value prediction results in portfolio optimization.

Surrogate model	RMSE	SMAPE	R^2	Inference time [sec./sample]
DNN	0.031	23.1%	0.990	1.28×10^{-6}
GBT	0.068	41.1%	0.953	1.88×10^{-6}
GP	0.135	66.1%	0.815	10.0×10^{-6}
LR	0.311	102.3%	0.023	0.32×10^{-6}

3.5.4 Optimization Performance Comparison

We investigated the effectiveness of ICSSC regarding implementation error and total runtime compared with conventional scenario-generation methods. The original scenario set Ξ was first generated by sampling $N = 1000$ scenarios from $Normal(\mu, \Sigma)$ uniformly at random. Each scenario-generation method listed below was then applied to obtain K scenarios. Finally, the portfolio optimization problem was solved with these representative scenarios using a QP solver. Ten trials were conducted for each method with $K = 5, 10, 20, 50, 100$, and the averaged evaluation results, such as solution accuracy and implementation error, were evaluated with the original scenario set Ξ . As ablation studies for ICSSC, the effectiveness of CSED was evaluated by comparing MIPP and CSED-based k -medoids clustering, denoted as *MIPP* and *CSED*, respectively. We also compared the method of obtaining $\hat{x}_k^*$ between directly solving $\hat{Q}_k$ and approximately solving $\hat{Q}_k$ within DVSS, denoted as *Direct* and *DVSS*, respectively. Thus, the following methods were compared.

- **Monte Carlo** [77]: K scenarios are obtained by random sampling from Ξ .
- **Moment matching** [71]: K scenarios are selected to approximate the statistical moments (mean, standard deviation, skewness, kurtosis, correlation) of Ξ .
- **k -means** [75]: K scenarios are selected as the centroids of the k -means clustering from Ξ .
- **k -medoids** [76]: K scenarios are selected as the medoids of the k -medoids clustering from Ξ .
- **GMM** [78]: K scenarios are selected as the centroids of the GMM clustering from Ξ .
- **CVAE-SIPA** [85]: K scenarios are selected as the medoids of the k -medoids clustering from Ξ , the features of which are the latent representation obtained by CVAE.
- **CSSC** [68]: K scenarios are selected by the original CSSC using MIPP.
- **ICSSC (DVSS, CSED)**: K scenarios are selected by ICSSC with CSED-based k -medoids, and $\hat{Q}_k$ is solved within DVSS. (Proposed method)

- **ICSSC (DVSS, MIPP)**: K scenarios are selected by ICSSC with MIPP, and $\hat{Q}_k$ is solved within DVSS.
- **ICSSC (Direct, CSED)**: K scenarios are selected by ICSSC with CSED-based k -medoids, and $\hat{Q}_k$ is solved directly.
- **ICSSC (Direct, MIPP)**: K scenarios are selected by ICSSC with MIPP, and $\hat{Q}_k$ is solved directly.

When using MIPP, the time limit of MIPP is set to 100 sec, and 200 scenarios sub-sampled from Ξ are used as the original scenario set since $N = 1000$ is too large to handle in MIPP. Single-scenario subproblems are solved using a QP solver in CSSC, and CMA-ES with 20 populations in CVAE-SIPA and ICSSC. The number of DVSS samples is $M = 2^{12}$.

Optimization Performances and Runtimes

Figure 3.2 shows the mean implementation errors, defined in Equation 3.4, as the percentage of the optimal objective values for each method. Table 3.2 lists the detailed evaluation results of the objective values, with boldface indicating the top three methods. In addition, Table 3.3 shows total runtimes in $K = 5$. These results indicate that problem-driven methods outperformed the distribution-driven methods, especially when K was small, while CSSC's solution significantly deteriorated in $K \geq 50$. In terms of total runtime, CVAE-SIPA and ICSSC (DVSS, CSED) were faster than the other problem-driven methods using MIPP.

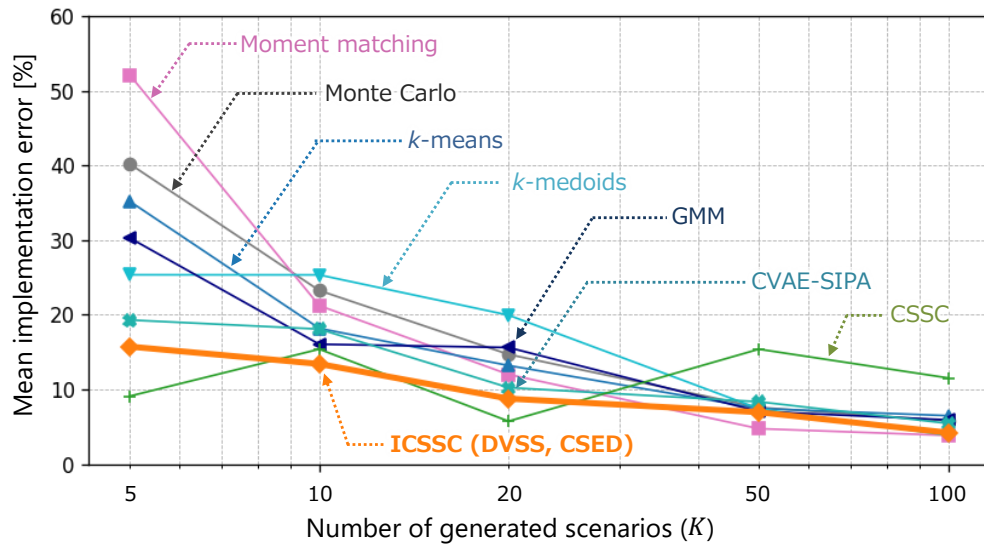


Figure 3.2. Mean implementation errors in Markowitz portfolio optimization problem compared with each scenario generation method.

Table 3.2. Objective function value comparison of Markowitz portfolio optimization.

Classification	Method	Objective function value [$\times 10^{-3}$]				
		$K = 5$	$K = 10$	$K = 20$	$K = 50$	$K = 100$
Distribution-driven	Monte Carlo	176 ± 38	155 ± 20	144 ± 10	135 ± 3	133 ± 3
	Moment matching	191 ± 39	152 ± 10	141 ± 7	132 ± 3	131 ± 2
	k -means	170 ± 0	149 ± 13	142 ± 11	135 ± 4	134 ± 3
	k -medoids	158 ± 30	158 ± 33	151 ± 25	135 ± 6	131 ± 2
	GMM	164 ± 14	146 ± 10	145 ± 11	135 ± 4	133 ± 3
Problem-driven	CVAE-SIPA	150 ± 13	148 ± 15	139 ± 5	136 ± 4	133 ± 3
	CSSC	137 ± 0	145 ± 0	133 ± 0	145 ± 0	140 ± 1
	ICSSC (DVSS, CSed)	145 ± 8	143 ± 9	137 ± 9	134 ± 5	131 ± 3
	ICSSC (DVSS, MIPP)	183 ± 0	170 ± 0	159 ± 0	145 ± 0	133 ± 0
	ICSSC (Direct, CSed)	172 ± 34	144 ± 9	147 ± 16	134 ± 5	132 ± 2
	ICSSC (Direct, MIPP)	184 ± 34	185 ± 73	180 ± 42	153 ± 17	136 ± 6
Reference	Quadratic programming	126				

Table 3.3. Runtime comparison of Markowitz portfolio optimization.

Classification	Method	Runtime ($K = 5$) [sec.]		
		Scenario	Optimize	Total
Distribution-driven	Monte Carlo	0.000	0.008	0.008
	Moment matching	7.006	0.004	7.010
	k -means	0.077	0.002	0.079
	k -medoids	0.008	0.005	0.013
	GMM	0.031	0.002	0.034
Problem-driven	CVAE-SIPA	0.017	0.003	0.020
	CSSC	97.154	0.003	97.158
	ICSSC (DVSS, CSed)	1.885	0.008	1.893
	ICSSC (DVSS, MIPP)	143.304	0.005	143.309
	ICSSC (Direct, CSed)	23.623	0.004	23.628
	ICSSC (Direct, MIPP)	167.229	0.012	167.240
Reference	Quadratic programming	0	0.001	0.001

Figure 3.3 shows the implementation errors obtained in the ablation study of ICSSC. ICSSC (DVSS, CSED) outperformed the other variants for all K . MIPP calculations in ICSSC (DVSS, MIPP) and ICSSC (Direct, MIPP) tended to finish by reaching the time limit while the relative optimality gaps remained around 90% when K is large. This means the MIPP results are far from the optimal solution, leading to lower accuracies than CSED-based clustering. When subproblems were directly solved in ICSSC (Direct, CSED) and ICSSC (Direct, MIPP), these computational times increased significantly. However, the objective function values did not improve since the solution accuracy of the subproblems can be limited by the prediction error of the surrogate model.

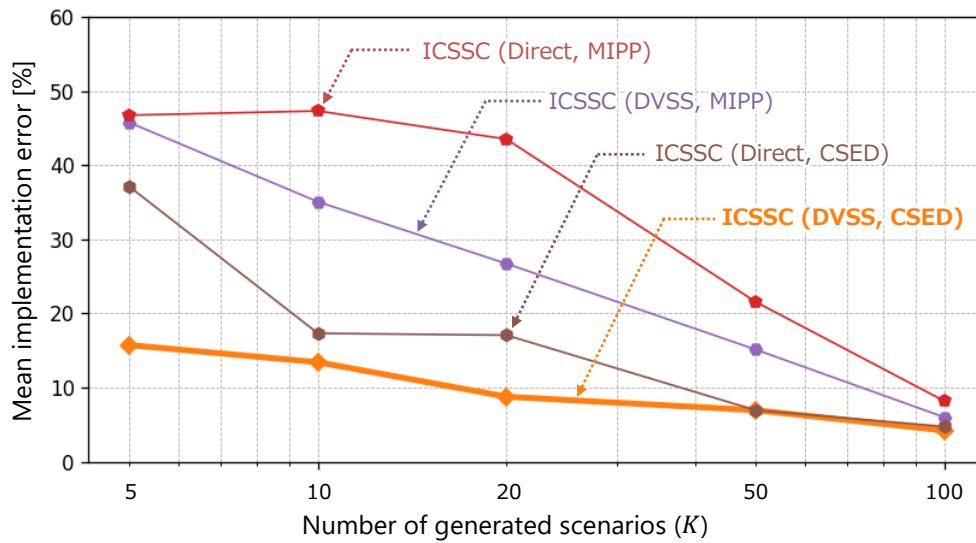


Figure 3.3. Mean implementation errors in Markowitz portfolio optimization problem obtained in ablation study of ICSSC.

Impact of Prediction Errors of Surrogate Models

Next, we investigated the impact of prediction errors of surrogate models on the optimization results. Figure 3.4 compares the implementation errors of the variants of ICSSCs, the DNNs of which were trained with 500, 1000, 5000, 10000, and 50000 training data, respectively. As the amount of training data increases, the prediction error (SMAPE) decreases, i.e., 92.8, 82.9, 40.5, 32.6, and 22.7%, respectively. The implementation error when the true objective function value was used instead of the surrogate model is also plotted. Figure 3.4 shows that the smaller the prediction error of the surrogate model, the smaller the implementation error.

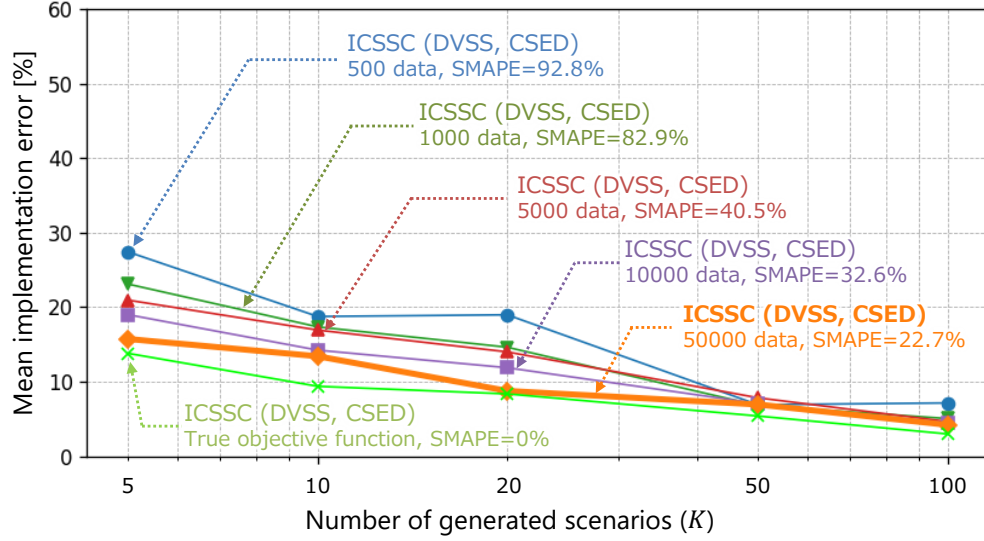


Figure 3.4. Mean implementation errors in Markowitz portfolio optimization problem compared with surrogate models with different prediction accuracies.

Characteristics of the Generated Scenarios

For deeper analysis of the proposed method, we investigated the characteristics of the generated representative scenarios with k -medoids clustering and ICSSC (DVSS, CSED). Figure 3.5 shows the representative scenarios $\tilde{\xi}$ obtained with these methods in $K = 5$. Figure 3.5-A shows the original scenario set, where each line represents a scenario. Figure 3.5-B shows that k -medoids selected $\tilde{\xi}$ that are evenly distributed in the scenario set. Figure 3.5-D, however, illustrates that ICSSC (DVSS, CSED) generated $\tilde{\xi}$ with various uncertain parameters in the dimensions $k = 2, 3$ and 5. In the optimal solution x^* , the decision variables in these dimensions (0.000, 0.007, 0.004) are smaller than those in the others (0.113, 0.361, 0.276, 0.276), indicating that these dimensions have a significant impact on the objective function. From Figure 3.5-C, each cluster was divided by the opportunity cost $V_{i,j}$ in the cost space, indicating that ICSSC (DVSS, CSED) selected these representative scenarios because the opportunity cost changes significantly due to the uncertain parameters in these dimensions. Therefore, ICSSC (DVSS, CSED) can generate more critical scenarios to evaluate the impact of uncertain parameters on performance measures than conventional distribution-driven methods.

Scalability to Larger Original Scenario Sets

Finally, we investigated the scalability of the proposed method for handling larger sets of original scenarios with sizes $N = 1000, 5000, 10000, 50000$, and 100000. In this experiment, we assessed the objective function values and runtimes of ICSSC (DVSS, CSED) with $K = 10$ using a newly

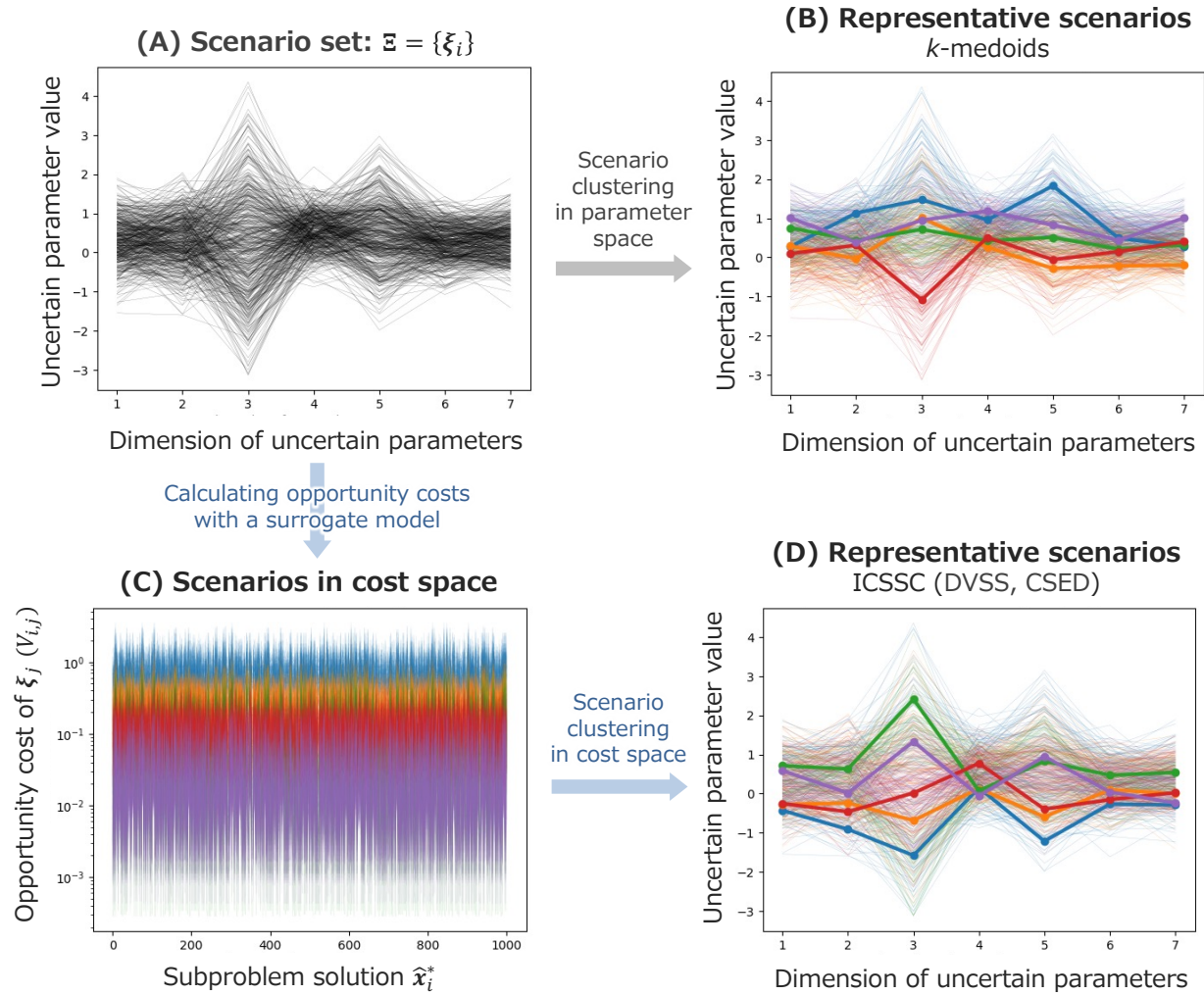


Figure 3.5. Visualization of cost space and representative scenarios of Markowitz portfolio optimization problem in $K = 5$. Figure A shows the original scenario set. Figure B illustrates representative scenarios obtained by k -medoids clustering. Figures C and D show scenarios in cost space and representative scenarios of ICSSC (DVSS, CSED). Each line means scenario, and its color indicates cluster it belongs to. Thick line is representative scenario of each cluster.

sampled set of 100,000 scenarios. Consequently, the results may differ slightly from those reported in Table 3.2. The evaluation results are shown in Table 3.4. It demonstrates that ICSSC (DVSS, CSED) can be applied with a slight increase in runtime when $N \leq 50000$, which is practically fast for real-world applications. However, when $N \geq 100000$, the runtime rises significantly. Table 3.4 indicates that the computation of k -medoids clustering takes up most of the runtime due to its $O(N^2)$ computational complexity. These results suggest that ICSSC could be widely applied to problems involving tens of thousands of scenarios. However, for managing larger original scenario sets, it might be necessary to alter the clustering method to reduce computational complexity.

Table 3.4. Objective function value and runtime comparison of ICSSC with different size of original scenario sets in Markowitz portfolio optimization problem.

Original scenario size N	Objective function value [$\times 10^{-3}$]	Runtime for scenario gen.	
		Clustering [sec.]	Total [sec.]
1000	144.0 ± 12.9	0.004	1.70
5000	142.6 ± 18.0	0.81	6.11
10000	141.2 ± 15.8	3.54	13.19
50000	139.9 ± 9.6	182.79	241.45
100000	139.0 ± 9.7	1297.14	1408.58

3.6 Stochastic Server Location Problem

In the second experiment, we applied ICSSC to the Stochastic Server Location Problem (SSLP) to examine the applicability in more detail. Note that we refer to ICSSC (DVSS, CSED) as ICSSC for the following sections.

3.6.1 Problem Statement and Experiment Settings

The SSLP is a common benchmark problem for two-stage stochastic MILPs, as described in [93]. First-stage binary variables represent the placement of servers at given locations. Second-stage variables indicate whether to serve a particular customer from a given server. The uncertainty lies in the potential customer demand for the service. SSLP instances are denoted as $m.n.s$, where m is the number of potential server locations, n is the number of potential clients, and s is the number of potential customer demand scenarios. We evaluated three SSLP instances from SIPLIB [94]: 5.50.100, 10.50.2000, and 15.45.15, representing 100, 2000, and 15 original scenarios, respectively. For each instance and scenario generation method, we performed five trials with $K = 5$, assessing average performances based on the original scenario set from SIPLIB. A DNN

was used as the surrogate model for all instances, and all other parameters remained consistent with the portfolio optimization problem described in Section 3.5.

3.6.2 Optimization Performance Comparison

Table 3.5 lists the results of the objective values, with boldface indicating the top three methods. These results demonstrate that problem-driven methods outperformed distribution-driven methods in SSLP, and ICSSC ranked among the top three for all instances. From these findings, we confirmed that ICSSC is also effective for SSLP.

Table 3.5. Objective function value comparison of SSLP.

Classification	Method	Objective function value ($K = 5$)		
		(5.50.100)	(10.50.2000)	(15.45.15)
Distribution-driven	Monte Carlo	-126.66	-345.53	-252.53
	<i>k</i> -means	-127.01	-340.57	-252.77
	<i>k</i> -medoids	-126.66	-341.75	-252.85
	GMM	-126.66	-346.00	-253.08
Problem-driven	CVAE-SIPA	-126.66	-347.01	-252.68
	CSSC	-127.37	-347.01	-252.20
	ICSSC	-127.01	-347.01	-253.32
Reference	SIPLIB [94]	-127.37	-347.26	-253.60

3.7 Inventory Placement Optimization Problem

Finally, we applied ICSSC to SBSPPs, an inventory placement optimization problem under demand uncertainty. As mentioned above, CSSC cannot be directly applied to SBSPPs. Therefore, ICSSC was compared with distribution-driven methods (Monte Carlo method and *k*-medoids) and a state-of-the-art problem-driven method (CVAE-SIPA [85]) regarding solution accuracy and computational time for preparing the dataset and solving this SBSPP using demand scenarios obtained by each scenario-generation method.

3.7.1 Problem Statement

We investigated an inventory placement optimization problem of the single-product multi-echelon supply chain (SC) network with uncertain demands. Figure 3.6 shows the target SC model, which consists of a Retailer ($s = 0$), Warehouse 1 ($s = 1$), Warehouse 2 ($s = 2$), and Manufacturer ($s = 3$), where s is the site index. Algorithm 3-2 describes the simulation processes, where S is the number

of sites, BS_s is the base stock, $OQ_{s,t}$ is the order quantity, $DQ_{s,t}$ is the demand quantity, $SQ_{s,t}$ is the shipment quantity, $AQ_{s,t}$ is the arrival quantity, and $RQ_{s,t}$ is the shortage quantity of the site s at period t . Each site has an order lead time L_s^O and transportation lead times L_s^T . In this problem, only the Manufacturer can produce the new inventory, and only Warehouses 1 and 2 can store it and order the inventory to the supply-side site with the base stock (order-up-to-level) policy. This SC model is implemented in Python and takes 0.15 sec for one simulation run.

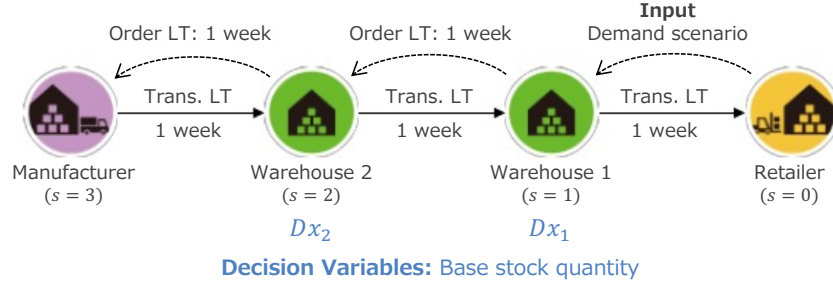


Figure 3.6. Target single-product multi-echelon SC network.

Algorithm 3-2 Overview of fulfillment simulation

Input: Base stocks $\{BS_s\}$, Retailer demands $\{DQ_t\}$.

Output: Inventory qty. $\{IQ_{s,t}\}$, Shortage qty. $\{RQ_{s,t}\}$.

- 1: Set $IQ_{s,0} = BS_s$, $AQ_{s,0} = OQ_{s,0} = 0$, $\forall s \in \{0, \dots, S-1\}$
 - 2: **for** $t = 1$ to T **do**
 - 3: **for** $s = 1$ to $S-1$ **do**
 - 4: **if** (Site s is Retailer) **then**
 - 5: Order items according to input demand: $OQ_{s,t} = DQ_t$
 - 6: Receive arrived items: $AQ_{s,t} = SQ_{s+1,t-L_s^T}$
 - 7: **else if** (Site s is Warehouse) **then**
 - 8: Receive order as demand: $DQ_{s,t} = OQ_{s-1,t-L_s^O}$
 - 9: Receive arrived items: $AQ_{s,t} = SQ_{s+1,t-L_s^T}$
 - 10: Dispatch inventory for shipment: $SQ_{s,t} = \min\{DQ_{s,t}, IQ_{s,t} + AQ_{s,t}\}$
 - 11: Evaluate shortage: $RQ_{s,t} = \max\{0, DQ_{s,t} - SQ_{s,t}\}$
 - 12: Update inventory: $IQ_{s,t} = IQ_{s,t-1} + AQ_{s,t} - SQ_{s,t}$
 - 13: Order items with base stock policy: $OQ_{s,t} = \max\{0, BS_s - IQ_{s,t}\}$
 - 14: **else if** (Site s is Manufacturer) **then**
 - 15: Ship ordered items: $SQ_{s,t} = OQ_{s-1,t-L_s^O}$
 - 16: **return** $\{IQ_{s,t}\}, \{RQ_{s,t}\}$
-

This optimization problem aims to determine the base stock at Warehouses 1 and 2 (BS_1, BS_2) for minimizing the total cost consisting of the inventory holding cost and shortage penalty cost over the evaluation period T_E under uncertain retailer demands $DQ_t \sim \Xi$ occurring weekly. To mitigate the effect of initial conditions, a burn-in period T_B was added before the evaluation period. Thus, the total simulation period was $T = T_B + T_E$. This problem is formulated as

$$\min_{\mathbf{x} \in \mathbf{X}} \sum_{k=1}^K p_k \left\{ \sum_{s=1}^2 IHC_s(\mathbf{x}, \xi_k) + SPC(\mathbf{x}, \xi_k) \right\} \quad (3.36)$$

$$\text{s.t. } IHC_s(\mathbf{x}, \xi) = \sum_{t=T_B+1}^T a_s IQ_{s,t}(\mathbf{x}, \xi), \quad (3.37)$$

$$SPC(\mathbf{x}, \xi) = \sum_{t=T_B+1}^T b RQ_{1,t}(\mathbf{x}, \xi), \quad (3.38)$$

$$D(x_1 + x_2) \leq B. \quad (3.39)$$

where the decision variable $\mathbf{x} = [x_1, x_2]$, demand scenario $\xi = [DQ_1, \dots, DQ_T]$, D is the demand scale parameter, base stock of site s is defined as $BS_s = Dx_s$. a_s is the inventory holding unit cost at site s , and b is the shortage unit cost. Equation 3.36 defines the objective function that sums up the inventory holding cost (IHC) and shortage penalty cost (SPC). Equations 3.37 and 3.38 define the IHC at site s (IHC_s) and shortage penalty cost for retailer demands (SPC) in the evaluation period ($t = T_B + 1, \dots, T$), respectively. Equation 3.39 defines the budget constraint that the total base stock quantity does not exceed the given upper limit B . Following Equation 3.8, the objective and constraint functions are notated as

$$F(\mathbf{x}, \xi) = \sum_{s=1}^2 IHC_s(\mathbf{x}, \xi) + SPC(\mathbf{x}, \xi), \quad G_1(\mathbf{x}) = D(x_1 + x_2) - B. \quad (3.40)$$

3.7.2 Experiment Settings

In this experiment, the demand distribution Ξ was defined by the following multivariate normal distribution [95]:

$$DQ_t \sim \frac{D}{10} \text{Normal}(\mu(t), \sigma(t, t')), \forall t, t' \in \{1, \dots, T\}, \quad (3.41)$$

where

$$\mu(t) = 10 + \sum_{p=1}^4 \gamma_p \phi_p(t + \tau), \quad \sigma(t, t') = \sum_{p=1}^4 \lambda_p \phi_p(t + \tau, t' + \tau), \quad \phi_p(t) = \sqrt{2} \sin(2\pi p t / T).$$

When preparing the training dataset for CVAE-SIPA and ICSSC, the parameters τ , γ_p , and λ_p were randomly sampled from the uniform distributions $[-2, 2]$, $[-5, 5]$ and $[0, 5]$, respectively. In evaluation, the following fixed parameters are used: $\tau = 0$, $\gamma_p = [1, 0.5, -0.5, -1]$ and $\lambda_p = [1, 0.5, 0.25, 0.5]$. The decision variables (x_1, x_2) were bounded in $[0, 5]$, respectively. The total simulation period was $T = 56$ weeks, $T_B = 4$ weeks, and $T_E = 52$ weeks. Other parameters were set as follows: $L_s^{\text{Order}} = L_s^{\text{Trans}} = 1$, $D = 1000$, $a_1 = 0.0001$, $a_2 = 0.0005$, $b = 1$, and $B = 6000$.

3.7.3 Surrogate Model Selection

In the pre-training phase, the surrogate model of the objective function in Equation 3.40 was trained as a regression problem. We compared the prediction accuracies of DNN, GBT, GP, and LR in terms of RMSE, SMAPE, and R^2 , and evaluated using 4-fold cross-validation. The training dataset $\mathcal{D}$ consisted of 50000 simulation results with input data as decision variables $\mathbf{x}$ generated uniformly at random from $[0, 5]$ and demand scenarios ξ sampled from Ξ defined in Equation 3.41. We used a DNN with the same architecture and training method described in Section 3.5.3 with hidden layers selected as $[200, 100, 100]$ through grid search. The number of estimators of GBT was set to 1000, and GP was trained with 300 randomly selected samples to be equivalent training time to the other models. For CVAE-SIPA, single-scenario subproblems were solved using CMA-ES with 20 populations.

Table 3.6 shows the prediction results of the surrogate models. These results indicate that DNN and GBT could accurately predict the objective function values, whereas GP and LR failed to do so. We assume that DNN and GBT had high model expressiveness and could learn the complex behavior of the simulator from the large dataset. We also confirmed that the DNN inference time (5.66×10^{-6} sec./sample) was much faster than the simulation runtime (0.15 sec./sample). These results suggest that DNN and GBT are suitable surrogate models for this optimization problem, and we chose to use DNN for the following experiments.

Table 3.6. Objective value prediction results in inventory placement problem.

Surrogate model	RMSE	SMAPE	R^2	Inference time [sec./sample]
DNN	431	4.24%	0.999	5.66×10^{-6}
GBT	548	4.53%	0.998	2.54×10^{-6}
GP	1.45×10^4	82.8%	0.139	13.6×10^{-6}
LR	7.48×10^7	66.4%	$\ll 0$	0.30×10^{-6}

3.7.4 Optimization Performance Comparison

Finally, the effectiveness of ICSSC is evaluated in comparison to the distribution-driven method (Monte Carlo method and k -medoids) and problem-driven method (CVAE-SIPA) in terms of implementation error and total runtime. Initially, the original scenario set Ξ consisting of $N = 200$ scenarios was sampled uniformly at random from the demand distribution described in Equation 3.41. Subsequently, K scenarios were generated using each scenario-generation method: ICSSC, CVAE-SIPA, k -medoids ($K = 3, 5, 10, 20$), and the Monte Carlo method ($K = 3, 5, 10, 20, 50, 100$). The inventory placement optimization problem was then solved using CMA-ES with 50 populations using K scenarios. We conducted five trials for each problem set and evaluated the average objective values and runtime using Ξ . To evaluate the implementation error in Equation 3.4, we solved the problem using all 200 scenarios in Ξ five times. The best solution was selected as the reference solution, shown in Table 3.7 and Table 3.8.

Optimization Performances and Runtimes

Table 3.7 presents the evaluation results for each scenario-generation method. The means and standard deviations of the decision variables obtained in five optimization trials are shown for each method. The table includes the average objective function values consisting of the IHC of Warehouse 1 (IHC_1), Warehouse 2 (IHC_2), and SPC . Figure 3.7 presents the evaluation results for the implementation error. Table 3.8 shows the average runtimes for scenario generation, optimization, and total runtime. These results indicate that problem-driven methods, such as ICSSC and CVAE-SIPA, generate scenarios that result in more accurate optimal solutions compared with distribution-driven methods such as k -medoids and Monte Carlo method, particularly when K is small. Notably, ICSSC can obtain solutions closer to the optimal with fewer scenarios than the other methods. For example, ICSSC with $K = 3$ obtained the solution with a mean total cost similar to that of Monte Carlo method with $K = 10$, while ICSSC's total runtime was 7.6 times shorter than that of Monte Carlo method. Moreover, ICSSC with $K = 20$ obtained a solution very close to the reference one in 18 times shorter runtime. This solution is almost as accurate as that with Monte Carlo method with $K = 100$, but the computational time was 7.2 times shorter. Note that ICSSC requires longer scenario-generation time than the others, but such computational time is sufficiently short for planning tasks such as inventory placement, enabling ICSSC to be applied in planning tasks in practical situations. These results indicate that ICSSC can reduce the total runtime to find a near-optimal solution compared with the other methods.

Table 3.7. Objective function value comparison of inventory placement optimization problem.

Method	K	Decision variables		Objective function value			
		x_1	x_2	IHC_1	IHC_2	SPC	Total
ICSSC	3	2.88 ± 0.05	3.12 ± 0.04	31.6	190.2	122.1	343.9
	5	2.92 ± 0.01	3.08 ± 0.01	33.0	186.1	116.9	336.0
	10	3.00 ± 0.06	3.00 ± 0.06	36.0	175.1	117.9	329.0
	20	3.00 ± 0.03	3.00 ± 0.03	35.9	175.6	114.7	326.2
CVAE-SIPA	3	2.85 ± 0.05	3.08 ± 0.13	30.6	182.1	154.1	366.9
	5	3.00 ± 0.10	2.92 ± 0.03	35.9	160.9	159.9	356.7
	10	2.96 ± 0.14	3.04 ± 0.14	34.4	179.5	130.4	344.4
	20	2.94 ± 0.06	3.06 ± 0.06	33.7	183.4	117.0	334.1
k -medoids	3	2.88 ± 0.16	2.88 ± 0.08	31.8	150.4	247.4	429.6
	5	2.90 ± 0.02	2.99 ± 0.12	32.4	170.0	165.4	367.7
	10	3.00 ± 0.02	2.96 ± 0.04	36.1	169.2	128.7	334.0
	20	2.98 ± 0.08	3.02 ± 0.08	35.3	177.4	119.0	331.7
Monte Carlo	3	2.86 ± 0.16	2.99 ± 0.22	31.1	167.3	264.1	462.5
	5	2.92 ± 0.17	2.96 ± 0.14	33.1	162.7	194.6	390.5
	10	2.87 ± 0.03	3.13 ± 0.03	31.5	190.8	122.1	344.4
	20	3.04 ± 0.09	2.96 ± 0.09	37.3	169.9	126.2	333.4
	50	2.99 ± 0.04	3.01 ± 0.04	35.7	176.1	115.8	327.6
	100	3.01 ± 0.04	2.99 ± 0.04	36.3	173.9	116.2	326.4
Reference	200	3.03	2.97	37.0	171.7	115.5	324.2

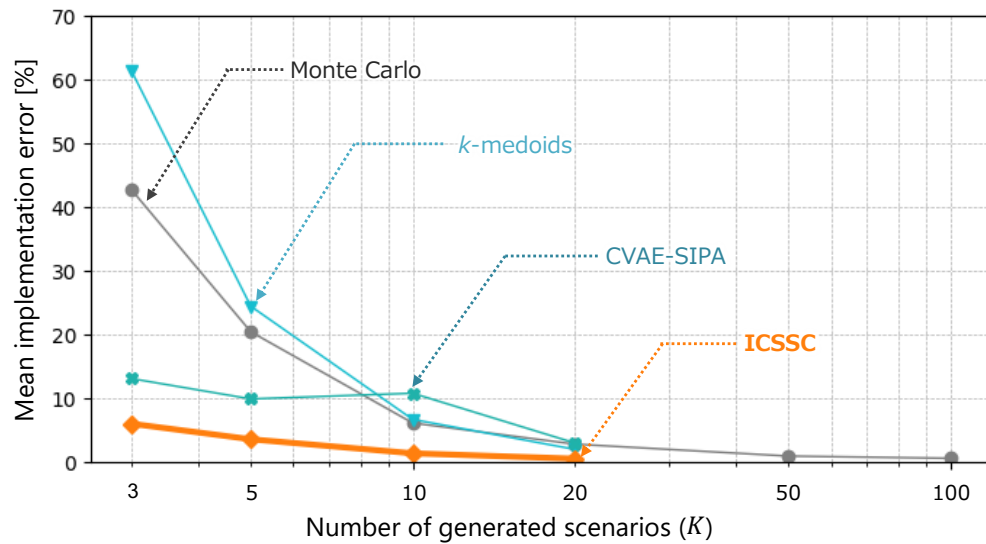
**Figure 3.7.** Mean implementation errors in inventory placement optimization problem.

Table 3.8. Runtime comparison of inventory placement optimization problem.

Method	K	Scenario-gen. [sec.]	Optimization [sec.]	Total runtime [sec.]	Ratio
ICSSC	3	4.28	44.31	48.60	$\times 1.0$
	5	4.28	80.13	84.41	$\times 1.7$
	10	5.20	183.22	188.41	$\times 3.9$
	20	5.25	392.42	397.66	$\times 8.2$
CVAE-SIPA	3	0.04	96.65	96.69	$\times 2.0$
	5	0.01	127.68	127.68	$\times 2.6$
	10	0.01	219.95	219.95	$\times 4.5$
	20	0.01	482.85	482.85	$\times 9.9$
k -medoids	3	0.00	41.09	41.10	$\times 0.8$
	5	0.00	73.70	73.71	$\times 1.5$
	10	0.00	128.14	128.15	$\times 2.6$
	20	0.00	255.66	255.66	$\times 5.3$
Monte Carlo	3	0.00	116.36	116.36	$\times 2.4$
	5	0.00	156.10	156.10	$\times 3.2$
	10	0.00	368.28	368.28	$\times 7.6$
	20	0.00	684.59	684.59	$\times 14.1$
	50	0.00	1738.48	1738.48	$\times 35.8$
	100	0.00	2874.70	2874.70	$\times 59.2$
Reference	200	0.00	7175.89	7175.89	$\times 144.0$

Data Generation and Training Runtimes

Next, we compared the computation time for data generation and training between ICSSC and CVAE-SIPA, as shown in Table 3.9. As mentioned above, CVAE-SIPA takes a long time to generate training data because it requires the results of various single-scenario subproblems. ICSSC, however, requires only the simulation results for random inputs as the training data, making the data-generation time about three times shorter than that of CVAE-SIPA. CVAE-SIPA takes a long time to train because the CVAE has a complex encoder-decoder structure and stochastic elements. Conversely, ICSSC's training time is about 200 times shorter than CVAE-SIPA since it uses a simple DNN and can easily reduce the training time through methods such as a one-cycle policy [92]. This indicates that ICSSC can reduce the total data-generation and training runtime by a factor of 6.89 compared with CVAE-SIPA.

Table 3.9. Computation times for data-generation and training in inventory placement problem.

Method	Data gen. [sec.]	Training [sec.]	Total runtime [sec.]	Ratio
ICSSC	1643	32	1675	$\times 1$
CVAE-SIPA	4709	6840	11549	$\times 6.89$

Characteristics of the Generated Scenarios

Finally, we discuss the characteristics of the obtained solutions from the generated scenarios and the inventory placement results. Figure 3.8 presents the representative scenarios generated with each method in $K = 3$, and Figure 3.9 shows the simulation results with the base stocks obtained as solutions using each method. This problem is critical to avoid shortages because the SPC is higher than the IHC. Comparing the generated scenarios in Figure 3.8, ICSSC produced more varied demand scenarios than the other methods. Figure 3.9 shows that the base stocks obtained using ICSSC were higher than those of the other methods. This leads to a more robust solution to avoid shortages against larger demand fluctuations. This is also confirmed in Table 3.7, where the results of ICSSC indicate a lower SPC than those of the other methods with the same K .

These experimental results confirm that ICSSC can select a set of necessary and sufficient scenarios even when K is small, leading to quickly obtaining a more robust optimal solution compared with the conventional methods.

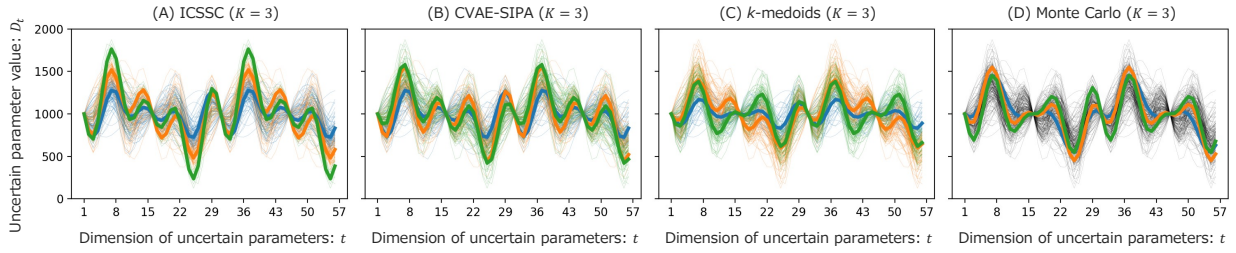


Figure 3.8. Visualization of representative scenarios in $K = 3$. Each line means a scenario, and its color indicates cluster it belongs to. Thick line is representative scenario of each cluster.

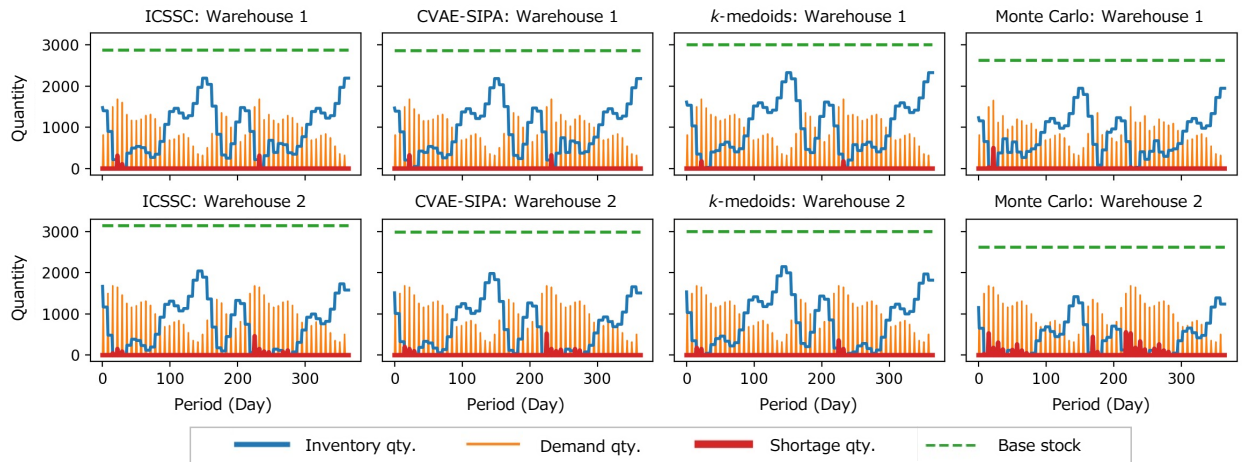


Figure 3.9. Simulation results with base stocks obtained as solutions using each scenario-generation method in $K = 3$. Each figure shows time series of inventory (blue), demand (orange), shortage (red), and basestock (green) quantities of Warehouses 1 and 2, respectively.

3.8 Conclusion

We presented a scenario-generation method, ICSSC, applicable to SBSPPs. ICSSC uses a surrogate model pre-trained with offline simulation datasets and selects K representative scenarios using k -medoids clustering based on the cost-space distance evaluated with the surrogate model. Three types of numerical experiments – Markowitz portfolio optimization, SSLP, and inventory placement optimization as SBSPP – demonstrated that ICSSC can generate representative scenarios that yield more robust and optimal solutions than conventional methods. Notably, it achieved a solution equivalent to the Monte Carlo method in 7.2 times less computational time.

From these results, we can draw the following insights:

- Problem-driven scenario-generation methods can generate more appropriate representative scenarios than distribution-driven methods, even in SBSPPs.
- CSED-based clustering is more scalable than MIPP and can execute fast and accurate scenario clustering even when the numbers of representative scenarios and original scenarios are large.
- Our experimental results suggest that higher prediction accuracy leads to higher-quality generated scenarios in problem-driven approaches leveraging ML techniques.

We plan to extend ICSSC to generate high-dimensional scenarios, such as future demands for multiple products, and apply it to SBSPPs that require highly robust solutions such as power distribution management and mechanical design optimization.

Integration of Sales, Inventory, and Transportation Resource Planning by Dynamic-demand Joint Replenishment Problem with Time-varying Costs

4.1 Introduction

Sales and operations planning (S&OP) serves as a pivotal integrated tactical planning process, ensuring a harmonious balance between demand and various supply capabilities, including production, distribution, procurement, and finance. This alignment with the sales plan empowers companies to enhance customer service, streamline inventory, reduce lead times, and provide top management with a comprehensive overview of the business [96]. Notably, *distribution planning* emerges as a key S&OP process, bridging the divide between production and distribution capacity and sales objectives. Its core aim is to set precise inventory targets and maximize the utilization of warehousing and transportation resources, both internally and externally, to meet the demands outlined by the sales plan [97][98]. Especially, the rise of e-commerce has presented new challenges in distribution planning, such as handling limited logistics resource capacities, such as storage spaces, trucks, and warehouse workers. Additionally, the unit cost of transportation vehicles, especially those owned by third-party logistics (3PL) providers, surges during high-demand seasons like national holidays. In such a situation, while replenishing inventories during periods of low unit transportation costs can reduce expenses, it can also lead to increased inventory holding costs. Thus, an optimal distribution plan should strike a balance between these costs, taking into account the fluctuating unit transportation costs and capacity constraints.

Operations research technologies, including mathematical optimization, constraint programming, and simulation, play a pivotal role in bolstering S&OP decision-making [99]. Notably, distribution planning has often been approached as a *joint replenishment problem* (JRP) [100][101]. The JRP seeks to optimize the replenishment plan, defining both the order quantity and timing for each item, to minimize overall costs while meeting specific demands [102]. This plan is typically articu-

lated through an inventory policy, which sets guidelines on order timings and quantities, tailored by specific *policy variables* for each item. Recognized as a foundational multi-item lot-sizing problem, the JRP has garnered significant attention, especially since multi-item replenishment is a prevalent scenario in supply chain management (SCM) [103]. Various JRP models exist, differentiated by demand type: deterministic-static, stochastic-static, and deterministic-dynamic. Among these, the deterministic-dynamic demand model, also known as dynamic-demand JRP (DJRP), accounts for demands that are pre-known but fluctuate throughout the planning horizon [104][105][106].

This research addresses the distribution planning through the lens of DJRP, given that the sales plan is characterized by deterministic-dynamic demand. Our primary objective is to devise a distribution plan that minimizes overall costs, factoring in time-varying unit transportation costs and capacity constraints, by optimizing inventory policy variables for each item and period. This necessitates solving the capacitated DJRP with time-varying costs (CDJRP-TC). Notably, most existing JRP and DJRP models assume constant unit costs throughout the planning horizon. There's a noticeable gap in research addressing DJRPs that consider both time-varying unit costs and capacity constraints. Hence, this paper introduces a mathematical formulation for CDJRP-TC and presents a novel solution methodology.

This problem can be formulated as a mixed integer linear programming (MILP) model. However, it contains enormous binary variables to express the discrete behaviors of inventory policies and, therefore, only small-scale CDJRP-TC can be solved with commercial MILP solvers. Conversely, simulation-based optimization (SBO) techniques, frequently combined with metaheuristics such as genetic algorithms (GA), are often used to solve complex systems' optimization problems. However, these approaches demand significant computational resources given the numerous iterative simulation cycles involved, thus making them unsuitable for large-scale CDJRP-TC. Therefore, to quickly solve the large-scale CDJRP-TC, we propose a three-phase approximate solution approach that combines MILP and SBO. The contributions of this paper are as follows: (1) formulation of distribution planning with outsourced transportation resources as CDJRP-TC, (2) proposal of the three-phase approach method for solving CDJRP-TC, (3) evaluation of the proposed method demonstrating its effectiveness in solving CDJRP-TC over MILP and GA, and (4) investigation of the proposed method's capability to obtain the appropriate solution according to the cost increase rate.

In Section 2, we introduce works related to this study. Section 3 explains JRPs and describes the assumptions of the CDJRP-TC. Section 4 formulates the CDJRP-TC as both an analytical and a simulation model. Section 4 presents the proposed three-phase solution approach. In Section 5, we evaluate the performance of the proposed method. We conclude in Section 6 with a summary and future work.

4.2 Related Works

4.2.1 Joint Replenishment Problem

The joint replenishment problem (JRP) is one of the most fundamental multi-item lot-sizing models [102]. The JRP determines the replenishment amounts and timings of items in the same order, generally provided by the same supplier, and corresponding warehousing and transportation resources to reduce the total cost.

In terms of demand type, JRPs are categorized into (1) deterministic-static, (2) stochastic-static, and (3) deterministic-dynamic demand [107]. Here, JRPs with deterministic-dynamic demand, i.e., dynamic-demand JRP (DJRP), consider the demands that are known in advance but vary over the planning horizon. It is difficult to solve DJRPs analytically because the dynamic demand increases the problem's complexity. DJRPs have been studied extensively because of their importance in industry and mathematical complexity. Federgruen and Tzur [108] proposed an efficient branch-and-bound-based heuristic approach for an uncapacitated DJRP with time-varying costs and optimized order quantities and timings for those with 20–30 items and 20–30 periods within a reasonable time. Robinson, Narayanan, and Gao [109] studied the effective solution approaches for the DJRP and determined that a two-phase heuristic and simulated annealing (SA) outperform existing methods. Gao, Altay, and Robinson [110] studied the unconstrained DJRPs and proposed two MILP formulations with tight LP relaxations that can be solved with the primal-dual algorithms.

Another important classification aspect of the JRP model is the capacity constraint; for example, production, transportation, and storage capacity. The capacitated JRP is difficult to solve within a reasonable time because even a two-item JRP with constrained capacity is NP-hard [111]. Kiesmüller [112] studied the stochastic-static JRP with full truckload constraints and provided approximated formulas to compute the policy variables of the (Q, S) policy, which are the variants of (s, S) policy that always satisfies the full truckload constraints. Singha, Buddhakulsomsiri, and Parthanadee [113] proposed a mathematical formulation of the JRP based on the (s, q) policy with the stochastic-static demand and the storage capacity constraint for one item and one period, which they solved using an iterative heuristic approach. Noh, Kim, and Hwang [114] studied the deterministic-dynamic JRP with a can-order policy with storage capacity and carbon cap constraints. They formulated this DJRP as a deterministic and a fuzzy MILP model, respectively, and solved these models, including ten items and 20 periods, using a commercial MILP solver.

Metaheuristics such as GA, SA, and other black-box optimization methods are also used to solve constrained JRPs. Chen, Yang, Jiang, *et al.* [115] developed a MILP model for JRPs with the shortage and partial demand substitution under budget, transportation capacity, and shipment

requirement constraints. They showed that DE outperformed other GA-based heuristic approaches. Lorenzo-Espejo, Muñuzuri, Guadix, *et al.* [116] considered the JRP of the omnichannel replenishment process under product devaluation. They solved this with the combined metaheuristic approach of particle swarm optimization (PSO) and SA. Liu, Wang, Pi, *et al.* [117] considered a coordinated capacitated dynamic lot-size and delivery problem and developed the six-phase heuristic (SPE) and the evolutionary algorithms (GA and DE) incorporating SPE (GA-S and DE-S, respectively). They reported that the DE-S provided more suitable solutions than the SPE and the GA-S for most problem instances.

4.2.2 Simulation-based Optimization Approaches

Recent studies dealing with complex inventory systems have widely utilized simulation models as an alternative to analytical models such as MILP. Simulation can model non-linear, discrete, and uncertain behaviors such as inventory policy and capacity constraints.

Simulation-based optimization (SBO) is one of the approaches to solving optimization problems whose objective functions are modeled by simulation. In this approach, to perceive the landscape of the objective function, solutions are evaluated by simulation, and black-box optimization methods such as metaheuristics are widely utilized as search algorithms [58]. Baharom and Hamzah [65] developed a simulation model mimicking the inventory system of a real-world electronics company. By Monte Carlo simulation, they determined the optimal reorder policy that results in the lowest average total inventory cost under uncertain demand and lead times. Pacheco, Cannella, Lüders, *et al.* [66] utilized a discrete event simulation to model a supply chain subject to market demand uncertainty and proposed a new reorder point update procedure for order-up-to-level policies in continuous review systems. Nagasawa, Irohara, Matoba, *et al.* [118] built the simulation model of JRP with the can-order policy to evaluate ordering penalty, shortage penalty, and inventory penalty. They solved this JPR with 200 products and one period using a multi-objective genetic algorithm (MOGA). Chu and You [119] used simulation to model real-world multi-echelon inventory systems under uncertainties. They propose an SBO framework for optimizing distribution inventory systems where each facility is operated with the (s, q) inventory policy by using a cutting plane algorithm. Lim, Alpan, and Penz [120] proposes a multi-objective simulation-optimization model for solving a S&OP problem in build-to-order industries with flexibility, uncertain demand, and impatient customers. They modeled this S&OP problem by using simulation with ϵ -constraints and solved it using SBO approaches based on random local search and SA. Grewal, Enns, and Rogers [121] proposed a framework to adjust reorder points and lot sizes based on a seasonal demand cycle. In this framework, a discrete-event simulation model of a capacity-constrained supply chain is developed. Then SBO experiments are performed to obtain optimal timing of adjustments, lot

sizes, and reorder points.

SBO approaches have been applied to various industrial engineering problems, such as scheduling, industrial processes, logistics, and SCM [122][123]. Hochmuth, Lässig, and Thiem [124] developed a metaheuristic SBO method for a complex multi-location inventory model consisting of (s, S) policy with stochastic static demand. They solved this problem with PSO and GA, respectively. Suemitsu, Bhamgara, Utsugi, *et al.* [88] developed an order sequence optimization method for complex automated picking systems modeled by simulation. SA assisted by a pretrained Bayesian recurrent neural network (BRNN) was proposed to quickly solve simulation-based sequence optimization problems by using the approximated simulation evaluation with BRNN. Yegul, Erenay, Striepe, *et al.* [125] tackled the production line configuration with SBO and proposed ant-colony optimization combined with a myopic search.

4.2.3 Hybrid Simulation and Optimization Approaches

Optimization approaches that utilize the output of a simulation model to compute performance measures are referred to as *hybrid simulation and optimization* [126]. Hybrid simulation and optimization approaches can be classified into three categories, *solution evaluation (SE)* referred to as SBO, *solution generation (SG)* and *analytical model enhancement (AME)* [127]. Specifically, SG and AME use analytical and simulation models, whereas SE (SBO) approaches generally use only a simulation model. In the SG approach, the simulation model is utilized to simulate the solutions of the analytical model to compute variables as part of the overall solution. While the SG approach does not use the simulation results for feedback to the analytical model, the AME approach updates the analytical model using the simulation results.

Nikolopoulou and Ierapetritou [128] proposed a hybrid approach combining mathematical programming and a simulation model that addresses the planning and scheduling in SCM. In this approach, a simulation is used to investigate the quality of the plans generated by the analytical model. Hrušovský, Demir, Jammerneegg, *et al.* [129] presented a solution framework that combines an analytical optimization model and simulation to determine intermodal transportation networks. They used the simulation model to evaluate uncertain travel times, whereas the optimization model provided the deterministic solution quickly. Keizer, Haijema, Bloemhof, *et al.* [130] developed a MILP model and a hybrid approach to identify a cost-optimal network design for distributing perishable products. In this method, the MILP network design solution is checked by a discrete event simulation to account for uncertainties in supply, processing, and transport. Then, the simulation results are used to update the product quality constraints in the MILP model iteratively.

Zhang, Matta, Alfieri, *et al.* [131] proposed the simulation-based Benders decomposition (BD), which is a variant of the generalized BD [132], for efficiently solving a large-scale MILP.

They proposed an algorithm to generate cuts using the simulation trajectory that utilizes simulation results as feasible solutions to the analytical model for solving joint workstation, workload, and buffer allocation problems. Forbes, Harris, Jansen, *et al.* [133] proposed Branch-and-Simulate that combined simulation and logic-based BD where a simulation is executed at each incumbent solution to the master problem. Using the proposed method, they solved an airport check-in counter allocation problem and a nursing home shift scheduling problem within a reasonable runtime.

Moreover, Table 4.1 is generated to present a detailed review of the literature regarding JRP and inventory control. As mentioned above, solving CDJRP-TC is critical to distribution planning in the recent logistics environment. However, to the best of our knowledge, the existing DJRP models do not consider the time-varying unit cost and the capacity constraints simultaneously. Thus, the primary objective of this study is to formulate CDJRP-TC mathematically and propose a new solution method based on the AME approach.

4.3 Problem Description and Assumptions

This section provides an overview of the capacitated DJRP with time-varying unit costs (CDJRP-TC) based on the (s, S) policy. In this section, Subsection 4.3.1 details the inventory policies, Subsection 4.3.2 delves into the JRP, and Subsection 4.3.3 outlines the assumptions of CDJRP-TC. Our discussion references the supply chain (SC) network shown in Figure 4.1, which illustrates the three-stage SC consists of suppliers, warehouses, and markets. In this structure, three suppliers, which have enough inventory to meet every shipment demand, replenish items in a warehouse based on a specific inventory policy. Subsequently, this warehouse distributes the items to markets based on a predefined sales plan. Note that these settings to simplify the SC model are adapted from the models presented by [121], [117], and [118]. For the following discussion, we categorize warehouse storage and workload as warehousing resources, while the vehicles from each supplier are considered transportation resources.

4.3.1 Inventory Policy

Inventory replenishment based on inventory policy is one of the most frequently used operations in real-world SCM because it enables efficient operations with a few parameters. The typical inventory policies are shown in Table 4.2. The goal is to optimize the policy variables of each item, site, and period to determine the appropriate inventory target.

In this study, the reorder point and base stock (order-up-to-level) (s, S) policy is used for the inventory replenishments from suppliers to a warehouse. Figure 4.2 illustrates the replenishment with the (s, S) policy. The inventory level z is defined as the warehouse inventory quantity plus

Table 4.1. Detailed investigation of literature related to JRP and inventory control

Author	Problem	Demand type	Inventory policy variable	Multiple products	Time-varying unit cost	Capacity consideration		Model type	Solution approach
						Transport	Storage		
Federgruen and Tzur [108]	JRP	Dynamic-deterministic	Order qty. (q)	✓	✓			Analytical	Branch-and-bound-based heuristic
Robinson, Narayanan, and Gao [109]	JRP	Dynamic-deterministic	Order qty. (q)	✓				Analytical	Two-phase heuristic, SA
Gao, Altay, and Robinson [110]	JRP	Dynamic-deterministic	Order qty. (q)	✓				Analytical	Primal-dual algorithms
Kiesmüller [112]	JRP	Stochastic-Static	(s, S) variant	✓		✓		Analytical	Heuristic
Singha, Buddhakulsomsiri, and Parthanadee [113]	JRP	Stochastic-static	(s, q)	✓			✓	Analytical	Iterative heuristic
Noh, Kim, and Hwang [114]	JRP	Deterministic-dynamic	Can-order (s, c, S)	✓			✓	Analytical	MILP
Chen, Yang, Jiang, <i>et al.</i> [115]	JRP	Deterministic-static	(t, q)	✓		✓		Analytical	DE, GA
Liu, Wang, Pi, <i>et al.</i> [117]	JRP	Deterministic-dynamic	Order qty. (q)	✓		✓		Analytical	Heuristic-DE hybrid
Lorenzo-Espejo, Muñozuri, Guadix, <i>et al.</i> [116]	Dynamic lot-sizing & delivery	Deterministic-static	Order qty. (q)	✓		✓		Analytical	PSO-GA hybrid
Baharom and Hamzah [65]	Inventory control	Stochastic-static	(s, q)					Simulation	Bruteforce search with Monte Carlo simulation
Pacheco, Cannella, Lüders, <i>et al.</i> [66]	Inventory control	Stochastic-static	(s, S)					Simulation	Heuristic
Nagasawa, Irohara, Matoba, <i>et al.</i> [118]	JRP	Deterministic-dynamic	Can-order (s, c, S)	✓				Simulation	MOGA
Chu and You [119]	Inventory control	Stochastic-static	(s, q)					Simulation	Cutting plane algorithm
Lim, Alpan, and Penz [120]	Inventory control	Stochastic-static	Order qty. (q)					Simulation	Local search-SA hybrid
Grewal, Enns, and Rogers [121]	Inventory control	Deterministic-dynamic	(s, q)	✓			✓	Simulation	Metaheuristics (OptQuest)
Hochmuth, Lässig, and Thiem [124]	Inventory control	Stochastic-static	(s, S)			✓	✓	Simulation	PSO, GA
Proposed method	JRP	Deterministic-dynamic	(s, S)	✓	✓	✓	✓	Analytical & simulation	Hybrid simulation & optimization approach

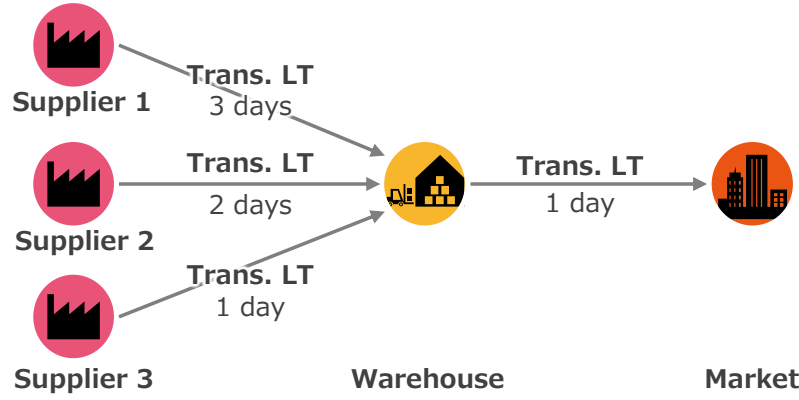


Figure 4.1. Target supply chain with three suppliers, one warehouse, and one market.

the quantity of orders that have been placed but have not arrived yet. In the (s, S) policy, if z has fallen below the reorder point s at the beginning of each period, the $S - z$ unit of the item is ordered to the corresponding supplier up to where the base stock is S . Note that our proposed method can be applied independently of the specific inventory policy. We adopted the (s, S) policy as a representative example in this study since this policy is one of the most extensively investigated inventory policies, as shown in the literature review.

Table 4.2. Classification of general inventory policies

Policy	Policy variables	Logic (Inventory level: z)
(t, q)	Review cycle: t Reorder quantity: q	A quantity q is ordered every t days.
(t, S)	Review cycle: t Base stock position: S	A quantity $\max(0, z - S)$ is ordered every t days.
(s, q)	Reorder point: s Reorder quantity: q	A quantity q is ordered if z is lower than s .
(s, S)	Reorder point: s Base stock position: S	A quantity $\max(0, z - S)$ is ordered if z is lower than s .

4.3.2 Description of Joint Replenishment Problem

The joint replenishment problem (JRP) considers N items and the planning horizon with T periods. For each item i , period t , and market k , demands D_{tik} are obtained from a sales plan. To satisfy these demands, replenishment orders may be placed in each period. The concept of JRPs when $N = 3$ is shown in Figure 4.3. This figure shows the inventory system based on the (s, S) policy, which replenishes inventory up to the base stock (order-up-to-level) S when the inventory level has

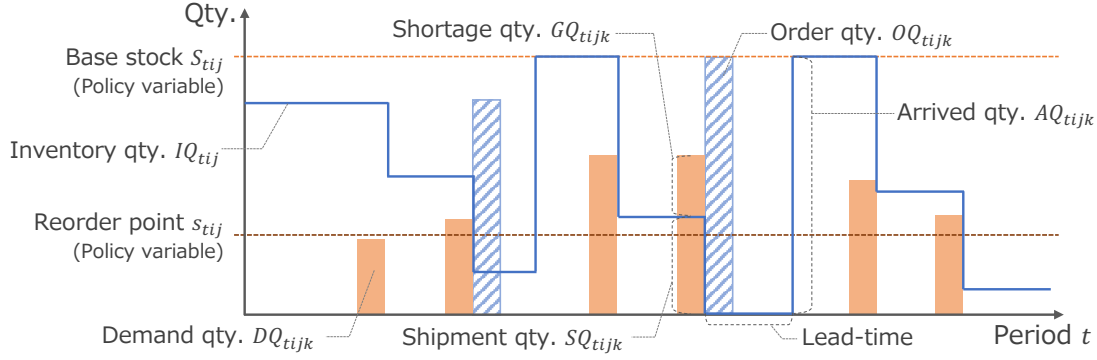


Figure 4.2. Reorder point and base stock (order-up-to-level) (s, S) policy.

fallen below the reorder point s . The left side of Figure 4.3 shows that the joint replenishment of three items is placed at period t_1 and two items at t_3 . The transportation is consolidated into two vehicles to minimize the number of vehicles because the transportation cost is determined by multiplying the number of vehicles and the unit transportation cost. The received inventories are stored in the warehouse, which consists of fixed and variable storage. All inventories are stored in the fixed storage until their total volume exceeds the fixed storage capacity. The inventory holding cost of the fixed storage is constant and incurred regardless of the inventory volume. The variable storage is used to store the inventory exceeding the fixed storage capacity. The variable inventory holding cost is proportional to the inventory volume in the variable storage.

The total cost of the JRP consists of two components: the fixed cost (or major ordering cost) and the variable cost (or minor ordering cost) ([110][134]). The fixed cost is incurred when at least one item is replenished regardless of the order quantity, such as the setup cost, the fixed inventory holding cost, and the transportation cost. Meanwhile, the variable cost depends on the replenishment order quantity, such as the variable inventory holding cost and the labor cost to receive replenished items in a warehouse. This paper defines the total cost as the summation of fixed and variable inventory holding, transportation, and labor costs.

4.3.3 Problem Assumptions

As mentioned above, this study discusses CDJRP-TCs in the supply chain network shown in Figure 4.1. The target CDJRP-TCs adhere to the following problem assumptions or characteristics:

1. The period t is defined as a day.
2. Each item is replenished with the given path between sites, referred to as the *supply route*.
3. The replenishment order arrives after transportation lead time of each supply route.
4. Two types of vehicles (*normal* and *extra* vehicle) are available, where the unit cost of the extra vehicle is higher than the normal vehicle.

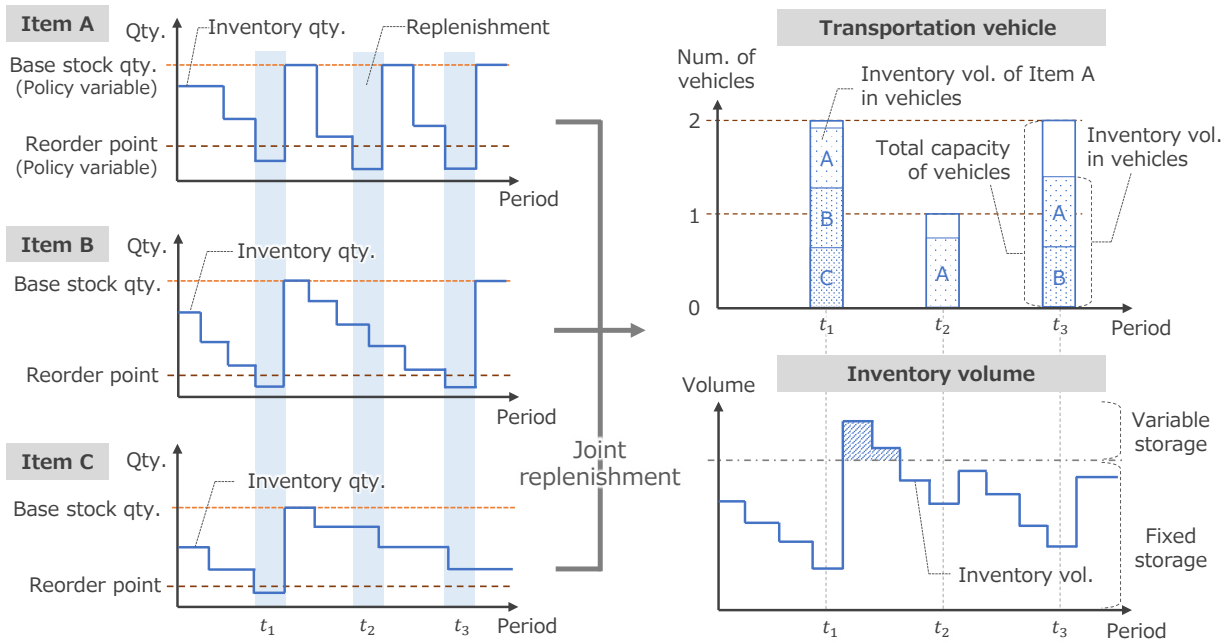


Figure 4.3. Overview of the joint replenishment problem with (s, S) policy. When the inventory quantity is less than the reorder point (s), the difference between the base stock quantity (S) and the inventory quantity is ordered for replenishment. The transportation of ordered items is consolidated to minimize the number of vehicles. The replenished inventories are stored in fixed storage when the stored volume does not exceed the storage capacity. After exceeding the capacity, inventories are stored in variable storage.

5. The maximum number of each type of vehicle per day is limited.
6. The maximum workload to receive and ship items in a warehouse per day is limited.
7. The inventory shortage is not backlogged.

4.4 Formulation of CDJRP-TC

This section presents a description and formulation of the CDJRP-TC based on the (s, S) policy. As mentioned above, we formulate DJRPs considering both time-varying unit costs and capacity constraints for addressing the distribution planning under recent logistics situations. To solve CDJRP-TC, we propose a three-phase solution method based on the AME approach shown in Figure 4.4. The solution process of the proposed method consists of the following three phases. In Phase 1, the simplified MILP model of CDJRP-TC is solved with a MILP solver, and *ideal inventory quantities (IIQ)* of each item are obtained as the inventory quantities of the MILP solution. Phase 2 estimates the policy variables that best fit the IIQs for each item using CMA-ES. In Phase 3, the shortage of each item is evaluated with the simulation model of CDJRP-TC, whose inputs are the estimated policy variables of all items. Then the IIQs are updated according to the shortages of each item. These procedures are iterated until no shortage occurs in the simulation, and the policy variables estimated in Phase 2 are returned as a solution. The proposed method utilizes the analytical MILP model and the simulation model of CDJRP-TC. Therefore, in this section, we formulate the CDJRP-TC as both the analytical and the simulation model.

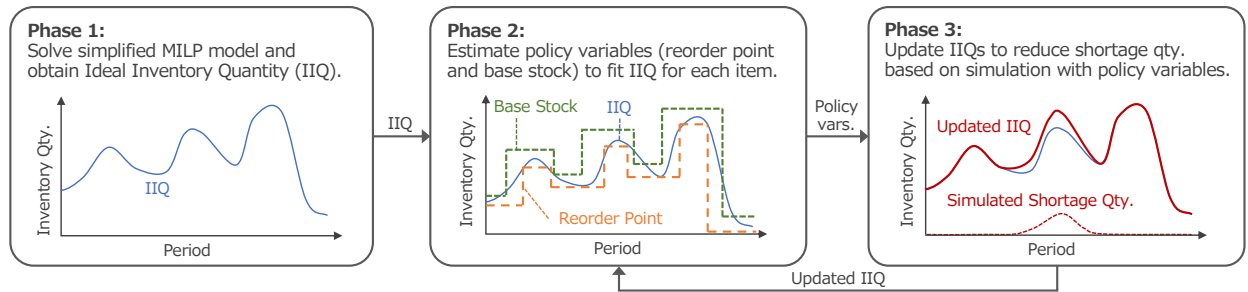


Figure 4.4. Concept of the proposed three-phase solution approach.

4.4.1 Notations

All notations required in this paper are defined below; the indexes and sets in Table 4.3, the input parameters in Table 4.4, and the decision variables in Table 4.5.

Table 4.3. Indexes and sets

Notation	Description
t	index of a period (day); $t \in \mathbf{T}$.
i	index of an item; $i \in \mathbf{I}$.
j, k	index of a site; $j, k \in \mathbf{J}$.
v	index of a vehicle; $v \in \mathbf{V}$.
$\mathbf{T}$	set of periods; $ \mathbf{T} = T$, $\mathbf{T} = \{t = 1, 2, 3, \dots, T\}$.
$\mathbf{I}$	set of items; $ \mathbf{I} = N$.
$\mathbf{J}$	set of sites.
$\mathbf{V}$	set of vehicles.
$\mathbf{J}_S$	set of supplier sites.
$\mathbf{J}_W$	set of warehouse sites.
$\mathbf{J}_M$	set of market sites.
$\mathbf{J}_{ij}^{Pre}$	set of the predecessor sites that can replenish item i to site j .
$\mathbf{J}_{ij}^{Suc}$	set of the successor sites that can replenish item i from site j .
$\mathbf{s}_{ij}$	set of reorder points of item i and site j ; $\mathbf{s}_{ij} = \{s_{tij} \mid \forall t \in \mathbf{T}\}$.
$\mathbf{S}_{ij}$	set of reorder points of item i and site j ; $\mathbf{S}_{ij} = \{S_{tij} \mid \forall t \in \mathbf{T}\}$.

Table 4.4. Input parameters

Notation	Description
D_{tik}	given demand of market k for item i at period t .
V_i^{Item}	unit volume of item i .
V_j^{Fix}	upper limit of the fixed storage volume of site j .
V_v^{Trans}	upper limit of transportation volume of vehicle v .
C_j^{Inv}	unit inventory holding cost of site j .
C_{tjkv}^{Trans}	unit transportation cost of vehicle v from site j to k at period t .
C_{tj}^{Labor}	unit labor cost of site j at period t .
T_{jk}	transportation lead-time from site j to k .
W_{ij}^{Labor}	unit labor workload for item i at site j .
W_{tj}^{Max}	upper limit of workload of site j at time t .
N_{tjkv}^{Trans}	upper limit of number of transportation vehicles v from site j to k at period t .
M	large positive value.
ω	positive value of shortage penalty weight.

Table 4.5. Decision variables

Notation	Description
s_{tij}	positive continuous variable for reorder point of item i in site j at period t .
S_{tij}	positive continuous variable for base stock of item i in site j at period t .
z_{tij}	positive continuous variable for inventory level of item i in site j at period t .
δ_{tij}	binary variable of the ordering status, which is 1 if replenishment order of item i in site j at period t is placed, 0 otherwise.
VQ_{tijv}	positive integer variable for vehicle quantity of type v from site j to k at period t .
IQ_{tij}	positive continuous variable for inventory quantity of item i in site j at the end of period t .
AQ_{tijk}	positive continuous variable for arrived quantity of item i from site j to k at period t .
OQ_{tijk}	positive continuous variable for the positive part of the order quantity of item i at period t placed from site j to k .
OQ'_{tijk}	positive continuous variable for the negative part of the order quantity of item i at period t placed from site j to k .
DQ_{tijk}	positive continuous variable for demand quantity of item i from site j to k at period t .
SQ_{tijk}	positive continuous variable for shipment quantity of item i from site j to k at period t .
GQ_{tijk}	positive continuous variable for shortage quantity of item i from site j to k at period t .
GR_{ij}	positive continuous variable for shortage rate of item i in site j .

Table 4.5. Decision variables (*Continued*)

Notation	Description
IV_{tj}	positive continuous variable for used variable storage volume in site j at period t .
WL_{tj}	positive continuous variable for used warehouse workload in site j at period t .
IQ_{tij}^{Ideal}	ideal inventory quantity of item i at site j in period t .
$IQ_{tij}^{Est}(\mathbf{s}_{ij}, \mathbf{S}_{ij})$	estimated inventory quantity of item i at site j in period t , which is the function of policy variables $(\mathbf{s}_{ij}, \mathbf{S}_{ij})$.

4.4.2 Relevant Costs

In this study, we consider the total cost of CDJRP-TC as the summation of the inventory holding cost, the labor cost, and the transportation cost. In the following discussion, $\{x\}$ denotes the set containing all elements of x .

The inventory holding cost for a planning horizon T and all warehouses $\mathbf{J}_W$ is defined as Eq. 4.1. In our CDJRP-TC model, two different storage types are considered: fixed storage (i.e., an in-house warehouse) and variable storage (i.e., an external warehouse). Inventories are stored in fixed storage unless the total inventory volume exceeds the fixed storage capacity. Then, if the total volume exceeds the capacity, the excess inventories are stored in variable storage. The inventory holding cost of the fixed storage (*fixed inventory holding cost*) is constant and incurred regardless of the inventory volume. The inventory holding cost of the variable storage (*variable inventory holding cost*) is proportional to the inventory volume in the variable storage (IV_{tj}). Therefore, the inventory holding cost is defined as the summation of the *fixed inventory holding cost* $C_j^{Inv} V_j^{Fix}$ and the *variable inventory holding cost* $C_j^{Inv} IV_{tj}$ for each period t and warehouse $j \in \mathbf{J}_W$, as follows:

$$f_{Inv}(\{IV_{tj}\}) = \sum_{t \in \mathbf{T}} \sum_{j \in \mathbf{J}_W} C_j^{Inv} (V_j^{Fix} + IV_{tj}) \quad (4.1)$$

The labor cost for a planning horizon T and all warehouses $\mathbf{J}_W$ is defined as Eq. 4.2. This is the summation of the warehouse workload WL_{tj} multiplied by the unit labor cost C_{tj}^{Labor} for each period t and warehouse $j \in \mathbf{J}_W$. Here, WL_{tj} is defined as the total workload to receive the arrived quantity and to fulfill the shipment quantity.

$$f_{Labor}(\{WL_{tj}\}) = \sum_{t \in \mathbf{T}} \sum_{j \in \mathbf{J}_W} C_{tj}^{Labor} WL_{tj} \quad (4.2)$$

The transportation cost for a planning horizon T and all supply routes are defined as Eq. 4.3, which indicates the summation of the number of used vehicles VQ_{tjkv} multiplied by the unit transportation

cost C_{tjkv}^{Trans} for each period t , supply route (j, k) and vehicle type v , as follows:

$$f_{Trans}(\{VQ_{tjkv}\}) = \sum_{t \in \mathbf{T}} \sum_{j \in \mathbf{J}_S} \sum_{k \in \mathbf{J}_W} \sum_{v \in \mathbf{V}} C_{tjkv}^{Trans} VQ_{tjkv} \quad (4.3)$$

4.4.3 Analytical Model Formulation: Mixed Integer Linear Programming

This subsection provides a MILP formulation as the analytical model of the CDJRP-TC. The MILP model is formulated as follows:

Objective function:

$$\min \{f_{Inv}(\{IV_{tj}\}) + f_{Labor}(\{WL_{tj}\}) + f_{Trans}(\{VQ_{tjkv}\})\} \quad (4.4)$$

Constraints:

$$GR_{ij} = \frac{\sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{J}_M} GQ_{tijk}}{\sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{J}_M} D_{tik}} \leq 0 \quad (\forall i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.5)$$

$$z_{tij} = IQ_{tij} + \sum_{\tau=t+1}^{t+T_{kj}+1} \sum_{k \in \mathbf{J}_{ij}^{Pre}} AQ_{\tau ikj} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.6)$$

$$-M \times (1 - \delta_{tij}) \leq s_{tij} - z_{tij} \leq M \times \delta_{tij} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.7)$$

$$0 \leq OQ_{tijk} \leq M \times \delta_{tij} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W, k \in \mathbf{J}_{ij}^{Pre}) \quad (4.8)$$

$$0 \leq OQ'_{tijk} \leq M \times (1 - \delta_{tij}) \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W, k \in \mathbf{J}_{ij}^{Pre}) \quad (4.9)$$

$$OQ_{tijk} - OQ'_{tijk} = S_{tij} - z_{tij} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W, k \in \mathbf{J}_{ij}^{Pre}) \quad (4.10)$$

$$IQ_{tij} = IQ_{t-1ij} + \sum_{k \in \mathbf{J}_{ij}^{Pre}} AQ_{tikj} - \sum_{k \in \mathbf{J}_{ij}^{Suc}} SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.11)$$

$$DQ_{tijk} = \begin{cases} D_{tik} & \text{if } k \in \mathbf{J}_M, \\ OQ_{t-1ikj} & \text{otherwise.} \end{cases} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (4.12)$$

$$WL_{tj} = \sum_{i \in \mathbf{I}} w_{ij}^{Labor} \left(\sum_{k \in \mathbf{J}_{ij}^{Pre}} AQ_{tikj} + \sum_{k \in \mathbf{J}_{ij}^{Suc}} SQ_{tijk} \right) \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (4.13)$$

$$AQ_{tijk} = SQ_{t-T_{jk}ijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (4.14)$$

$$IQ_{tij} \geq \sum_{k \in \mathbf{J}_{ij}^{Suc}} SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.15)$$

$$GQ_{tijk} = DQ_{tijk} - SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (4.16)$$

$$\sum_{i \in \mathbf{I}} V_i^{Item} IQ_{tij} - V_j^{Fix} \leq IV_{tj} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (4.17)$$

$$VQ_{tjvk} \leq N_{tjvk}^{Trans} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}, k \in \mathbf{J}, v \in \mathbf{V}) \quad (4.18)$$

$$\sum_{i \in \mathbf{I}} V_i^{Item} SQ_{tijk} \leq \sum_{v \in \mathbf{V}} V_v^{Trans} VQ_{tjvk} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (4.19)$$

$$WL_{tj} \leq W_{tj}^{Max} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (4.20)$$

Eq. 4.4 shows the objective function; the first term denotes the inventory holding cost, the second term is the labor cost, and the third is the transportation cost. Eq. 4.5 shows the shortage constraint. This defines the upper limit of shortage quantity, which must be zero in this report.

Eq. 4.6 – 4.10 are the MILP formulation of (s, S) policy defined in Table 4.2. Eq. 4.6 defines the inventory level as the warehouse inventory quantity plus the order quantities that have been placed but have not arrived yet. Eq. 4.7 shows that the replenishment order can be placed ($\delta_{tij} = 1$) only if the inventory level has fallen below the reorder point ($z_{tij} < s_{tij}$). Eq. 4.8 indicates that the positive part of the order quantity is zero ($OQ_{tijk} = 0$) if the replenishment order is not placed ($\delta_{tij} = 0$). Meanwhile, Eq. 4.9 defines the negative part of order quantity that is zero ($OQ'_{tijk} = 0$) if the replenishment order is placed ($\delta_{tij} = 1$). Eq. 4.10 means that the order quantity is equal to the difference between the base stock and inventory level ($OQ_{tijk} = S_{tij} - z_{tij}$) if the replenishment order is placed ($\delta_{tij} = 1$).

Eq. 4.11 – 4.16 model the inventory system. Eq. 4.11 is the inventory-balance constraint that ensures the change in the inventory quantity between period t and $t - 1$ is equal to the difference between total arrived quantities and total shipment quantity of period t for each item and site. Eq. 4.12 defines the demand for each item, site, and period according to the destination. Here, the first condition indicates that demands from markets are given as input values, and the second means that other demands are equal to the order quantity placed toward each site in the last period. Eq. 4.13 defines the warehouse workload as the summation of workloads to receive the arrived quantity and to fulfill the shipment quantity with the unit labor workload. Eq. 4.14 shows the relationship between the arrived quantity and shipment quantity, which indicates that the order arrives after the transportation lead time. Eq. 4.15 expresses the upper limit of shipping quantity and indicates that each item can be shipped no more than the inventory quantity of that period and site. Eq. 4.16 defines the shortage quantity as equal to the difference between demand quantity and shipment quantity.

Eq. 4.17 – 4.20 define the capacity constraints. Eq. 4.17 represents the usage of the variable inventory storage, which is defined as the inventory volume that exceeds the fixed storage volume. Eq. 4.18 defines the upper limit of vehicle quantities for each supply route and each period. Eq. 4.19 shows the transportation volume constraints for each supply route and each period. Eq. 4.20

expresses the upper limit of the total workloads of receiving and shipping operations in a warehouse.

4.4.4 Discrete Simulation Model Implementation

A simulation model of the CDJRP-TC was constructed as a time-driven simulation program to investigate the performances with respect to the policy variables. This simulation model is implemented to satisfy all assumptions defined in Subsection 4.3.3. The simulation model is expressed as Eq. 4.21.

$$[\{IV_{tj}\}, \{VQ_{tijv}\}, \{WL_{tj}\}, \{GQ_{tijk}\}] = \text{Simulator}(\{D_{tik}\}, \{s_{tij}\}, \{S_{tij}\}). \quad (4.21)$$

The input of the simulation is the set of demands $\{D_{tik}\}$ and the policy variables, such as reorder points $\{s_{tij}\}$ and base stock $\{S_{tij}\}$. In this simulation, each fulfillment operation, such as receiving, shipping, and replenishment order, is processed sequentially over the planning horizon ($t = 1, 2, \dots, T$). In each period t , the fulfillment operations of each site are simulated in the order of the suppliers, the warehouses, and the markets. In each operation, target items are selected to minimize the shortage quantity sequentially while satisfying all constraints except for the upper shortage rate constraint Eq.4.5. The outputs are the set of the used variable storage volumes $\{IV_{tj}\}$, the vehicle quantities $\{VQ_{tijv}\}$, the warehouse workloads $\{WL_{tj}\}$, and the shortage quantities $\{GQ_{tijk}\}$. These values are used to evaluate the total costs defined in Eq 4.4 and the shortage rate defined in Eq 4.5.

4.5 Solution Approach

4.5.1 Overview of Proposed Method

As described in Subsection 4.4.3, CDJRP-TC can be formulated as the MILP model. However, it is still difficult to solve since it contains a vast number of binary variables δ_{tij} for modeling the (s, S) inventory policy. This problem is also challenging to solve with SBO approaches in a practical computational time. The inventory fulfillment simulation defined in Subsection 4.4.3 needs a few to hundreds of seconds for one execution when the number of items N and the planning horizon T are large. As a result, SBO approaches, which search for the optimal solution with numerous iterations of solution evaluation with simulation, require a very long calculation time. Therefore, to solve the practical scale CDJRP-TC quickly, a more efficient solution method is necessary.

Figure 4.5 shows our proposed three-phase solution method. The solution process of the proposed method consists of the following three phases.

In Phase 1, we formulate the simplified MILP model of CDJRP-TC by ignoring the (s, S)

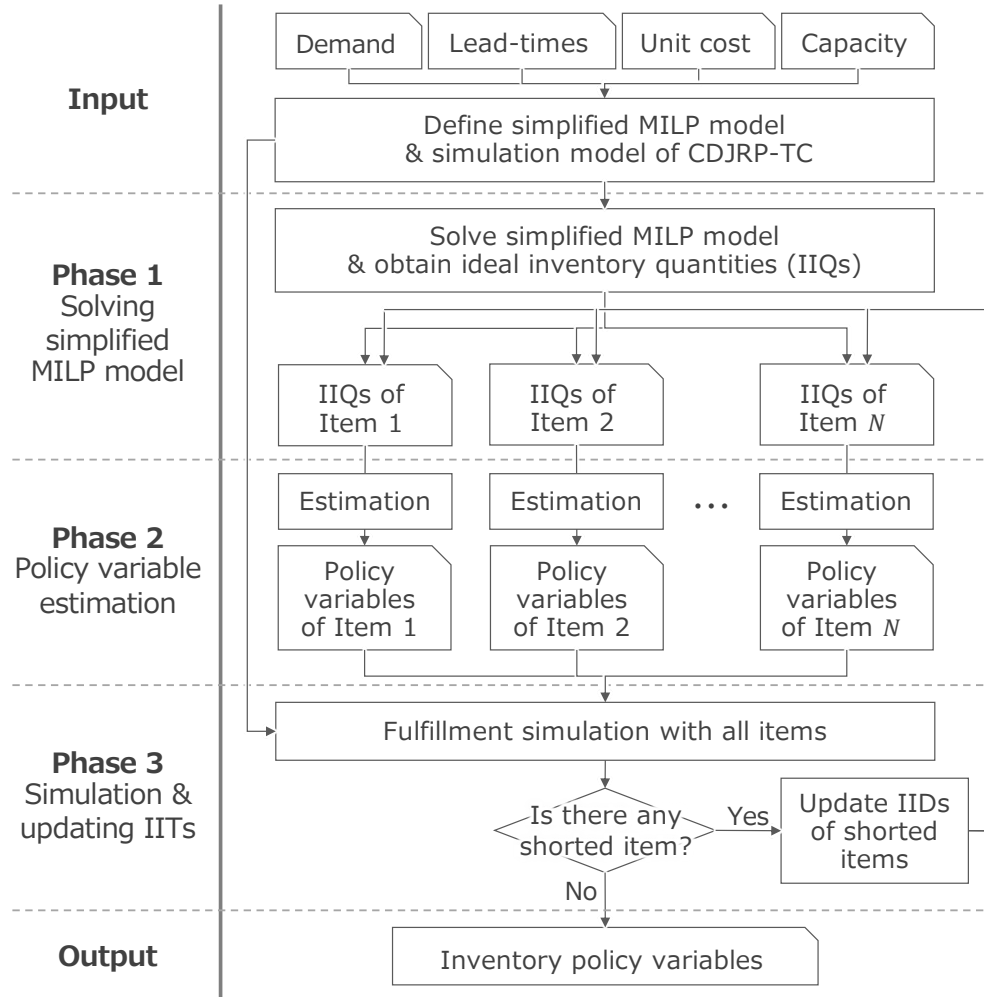


Figure 4.5. Flowchart of the proposed method. In Phase 1, the simplified MILP model that ignores inventory policies is solved, and ideal inventory quantities (IIQs) are obtained. In Phase 2, the policy variables that best fit the IIQs are estimated for each item in parallel. In Phase 3, the inventory fulfillment simulator with estimated policy variables is executed, and the IIQs of the shorted items are updated to reduce the shortage quantities.

inventory policy from the MILP formulation defined in Subsection 4.4.3. This simplified MILP model can be solved much shorter than the original MILP problem because the simplified model contains few binary variables δ_{tij} . Then the ideal inventory quantities (IIQs) of each item, which is the solution to the simplified MILP problem, are obtained. The IIQs may not satisfy the constraints of the inventory policy. However, the replenishment order quantities and timings of the IIQs are still valuable information as these two factors are essential for minimizing the total cost while satisfying constraints under the time-varying costs.

In Phase 2, therefore, we estimate the policy variables (s_{tij}, S_{tij}) that mimic the IIQs' replenishment order quantities and timings. In this policy variable estimation (PVE) process, the policy

variables of each item are estimated by CMA-ES to minimize the difference between the IIQs and the estimated inventory quantities (EIQs), which are the function of policy variables. Here, we ignored the capacity constraints, such as warehouse inventory, transportation, and workload capacities, because the PVE is executed for each item separately. Therefore, the policy variables obtained by PVE can violate the capacity constraints when used in operations simultaneously.

In Phase 3, the inventory fulfillment simulation, whose inputs are the policy variables of all items estimated in PVE, is executed to account for the capacity constraints. This simulation satisfies all capacity constraints automatically, as mentioned in Subsection 4.4.4. When these policy variables violate capacity constraints, the simulation outputs the positive shortage quantities. This means that these policy variables are a feasible solution to the CDJRP-TC if no shortage has occurred. Otherwise, the policy variables must be modified to reduce the shortage quantity to zero. Here, the policy variables are revised by executing PVE with the IIQs updated by the shortage quantities of each item and period.

Phase 2 and 3 are iterated until no shortage occurs in the simulation, and the estimated policy variables of the final iteration are returned as a feasible solution of the target CDJRP-TC. Additionally, Warehousing and transportation resources are obtained as the used variable storage, warehouse workloads, and vehicle quantities in the final solution. We note that our proposed method is a heuristic approach to provide a practical and efficient solution, especially for large-scale CDJRP-TCs where obtaining an exact solution might be computationally infeasible.

4.5.2 Phase 1: Simplified MILP Optimization

In this phase, the CDJRP-TC model is simplified by removing Eqs. 4.6 – 4.10, which models (s, S) inventory policies, as follows:

Objective function:

$$\min \{f_{Inv}(\{IV_{ij}\}) + f_{Labor}(\{WL_{ij}\}) + f_{Trans}(\{VQ_{tjkv}\})\} \quad (\text{Eq.4.4})$$

Constraints:

$$GR_{ij} = \frac{\sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{J}_M} GQ_{tijk}}{\sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{J}_M} D_{tik}} \leq 0 \quad (\forall i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.5})$$

$$IQ_{tij} = IQ_{t-1ij} + \sum_{k \in \mathbf{J}_{ij}^{Pre}} AQ_{tikj} - \sum_{k \in \mathbf{J}_{ij}^{Suc}} SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.11})$$

$$DQ_{tijk} = \begin{cases} D_{tik} & \text{if } k \in \mathbf{J}_M, \\ OQ_{t-1ikj} & \text{otherwise.} \end{cases} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (\text{Eq.4.12})$$

$$WL_{tj} = \sum_{i \in \mathbf{I}} W_{ij}^{Labor} \left(\sum_{k \in \mathbf{J}_{ij}^{Pre}} AQ_{tikj} + \sum_{k \in \mathbf{J}_{ij}^{Suc}} SQ_{tijk} \right) \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (\text{Eq.4.13})$$

$$AQ_{tijk} = SQ_{t-T_{jk}ijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (\text{Eq.4.14})$$

$$IQ_{tij} \geq \sum_{k \in \mathbf{J}_{ij}^{Suc}} SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.15})$$

$$GQ_{tijk} = DQ_{tijk} - SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (\text{Eq.4.16})$$

$$\sum_{i \in \mathbf{I}} V_i^{Item} IQ_{tij} - V_j^{Fix} \leq IV_{tj} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (\text{Eq.4.17})$$

$$VQ_{tjkv} \leq N_{tjkv}^{Trans} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}, k \in \mathbf{J}, v \in \mathbf{V}) \quad (\text{Eq.4.18})$$

$$\sum_{i \in \mathbf{I}} V_i^{Item} SQ_{tijk} \leq \sum_{v \in \mathbf{V}} V_v^{Trans} VQ_{tjkv} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (\text{Eq.4.19})$$

$$WL_{tj} \leq W_{tj}^{Max} \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (\text{Eq.4.20})$$

In this simplified model, any item can be replenished at any time with arbitrary quantities while other constraints (Eq. 4.5 and Eqs. 4.11 – 4.20) are satisfied. By solving this simplified model as MILP, the IIQs (IQ_{tij}^{Ideal}) are obtained as the solution's inventory quantities IQ_{tij} .

4.5.3 Phase 2: Policy Variable Estimation

In Phase 2, the policy variables (s_{tij}, S_{tij}) of each item and site are estimated to mimic the IIQs' replenishment order quantities and timings. This process is referred to as policy variable estimation (PVE), whose concept is shown in Figure 4.6. The reorder point is shown as the red line and the base stock as the orange line, which are the policy variables to be estimated. IIQs (IQ_{tij}^{Ideal}) obtained in Phase 1 are illustrated as the dotted cyan line. Here, we define *estimated inventory quantities* (EIQs), shown in the blue line. EIQs ($IQ_{tij}^{Est}(s_{ij}, S_{ij})$) are the inventory quantities obtained by replenishment based on the given policy variables (s_{ij}, S_{ij}).

To estimate the policy variables that mimic the IIQs' replenishment order quantities and timings, we formulate the PVE as an item-wise optimization problem that minimizes the difference between IIQs and EIQs. The PVE of item i and site j is formulated as follows.

Objective function:

$$\min \left\{ \sum_{t \in \mathbf{T}} \sum_{j \in \mathbf{J}_W} \left(IQ_{tij}^{Est}(s_{ij}, S_{ij}) - IQ_{tij}^{Ideal} \right)^2 + \omega \sum_{t \in \mathbf{T}} \sum_{j \in \mathbf{J}_W} \sum_{k \in \mathbf{J}_M} GQ_{tijk}^2 \right\} \quad (4.22)$$

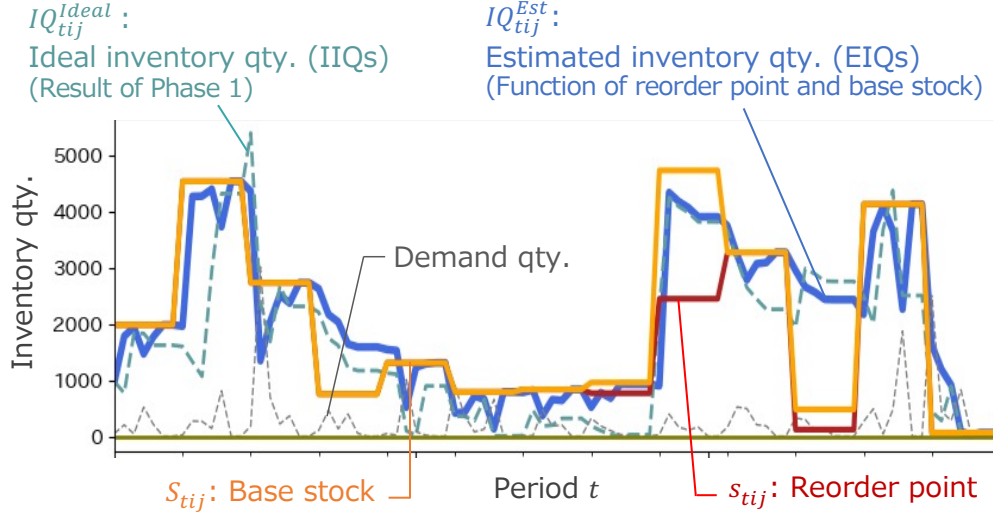


Figure 4.6. The concept of the policy variable estimation. The policy variables are estimated for each item to minimize the difference between EIQs and IIQs.

Constraints:

$$z_{tij} = IQ_{tij} + \sum_{\tau=t+1}^{t+T_{kj}+1} \sum_{k \in \mathbf{J}_{ij}^{Pre}} A Q_{\tau ikj} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.6})$$

$$-M \times (1 - \delta_{tij}) \leq s_{tij} - z_{tij} \leq M \times \delta_{tij} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.7})$$

$$0 \leq OQ_{tijk} \leq M \times \delta_{tij} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W, k \in \mathbf{J}_{ij}^{Pre}) \quad (\text{Eq.4.8})$$

$$0 \leq OQ'_{tijk} \leq M \times (1 - \delta_{tij}) \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W, k \in \mathbf{J}_{ij}^{Pre}) \quad (\text{Eq.4.9})$$

$$OQ_{tijk} - OQ'_{tijk} = s_{tij} - z_{tij} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W, k \in \mathbf{J}_{ij}^{Pre}) \quad (\text{Eq.4.10})$$

$$IQ_{tij} = IQ_{t-1ij} + \sum_{k \in \mathbf{J}_{ij}^{Pre}} A Q_{tikj} - \sum_{k \in \mathbf{J}_{ij}^{Suc}} S Q_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.11})$$

$$DQ_{tijk} = \begin{cases} D_{tik} & \text{if } k \in \mathbf{J}_M, \\ OQ_{t-1ijk} & \text{otherwise.} \end{cases} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (\text{Eq.4.12})$$

$$WL_{tj} = \sum_{i \in \mathbf{I}} W_{ij}^{Labor} \left(\sum_{k \in \mathbf{J}_{ij}^{Pre}} A Q_{tikj} + \sum_{k \in \mathbf{J}_{ij}^{Suc}} S Q_{tijk} \right) \quad (\forall t \in \mathbf{T}, j \in \mathbf{J}_W) \quad (\text{Eq.4.13})$$

$$A Q_{tijk} = S Q_{t-T_{jk}ijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (\text{Eq.4.14})$$

$$IQ_{tij} \geq \sum_{k \in \mathbf{J}_{ij}^{Suc}} S Q_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (\text{Eq.4.15})$$

$$GQ_{tijk} = DQ_{tijk} - SQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}, k \in \mathbf{J}_{ij}^{Suc}) \quad (\text{Eq.4.16})$$

Eq. 4.22 shows that the policy variables are optimized to minimize the squared errors between IQ_{tij}^{Est} and IQ_{tij}^{Ideal} with the shortage penalty, which is a relaxed formulation of Eq. 4.5. Additionally, Eqs. 4.6 – 4.16 describe the inventory system of item i . In item-wise PVE, we ignore the capacity constraints (Eqs. 4.17 – 4.20) since these evaluations require all items' information.

When a pair of the policy variables $(s_{ij}, \mathbf{S}_{ij})$ are given, its objective value (Eq. 4.22) can be easily evaluated because IQ_{tij}^{Est} and GQ_{tijk} are obtained by calculating the above one-item inventory model (Eqs. 4.6 – 4.16) from $t = 1$ to T sequentially. We, therefore, solved PVE by using the metaheuristics approach. In this study, the covariance matrix adaptation evolution strategy (CMA-ES), which is one of the metaheuristics for continuous optimization problems, is utilized ([57], [135]). CMA-ES facilitates optimization by updating the parameters of the multivariate Gaussian distribution and searches for the optimal policy variables by sampling candidate solutions from the distribution. Additionally, PVE can be separately solved for each item because of ignoring the capacity constraints to joint multi-item inventory models. This separation makes it possible to solve PVE quickly with reduced solution space and parallel computation.

4.5.4 Phase 3: Update of Ideal Inventory Quantities

The policy variables estimated in Phase 2 can violate the capacity constraints (Eqs. 4.17 – 4.20) because these are ignored in PVE. Thus, in Phase 3, we validate the shortage quantities (GQ_{tijk}) under the satisfaction of all capacity constraints by using the inventory simulation with all items defined in Section 3.3. If all items satisfy the shortage constraints (Eq. 4.5), the current policy variables are the solution to the target CDJRP-TC.

Otherwise, the current policy variables $(s_{ij}, \mathbf{S}_{ij})$ must be updated to satisfy the shortage constraint. However, it is difficult to modify the policy variables directly to reduce shortage because of their discrete and complex behaviors. Therefore, we consider updating the policy variables through PVE with the IIQs (IQ_{tij}^{Ideal}) revised with the simulation result. Here, Eq. 4.15 and Eq. 4.16 lead to the following inequation between the shortage quantity and inventory quantity:

$$IQ_{tij} + \sum_{k \in \mathbf{J}_{ij}^{Suc}} GQ_{tijk} \leq \sum_{k \in \mathbf{J}_{ij}^{Suc}} DQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.23)$$

In this inequation, the left-hand side is the inventory quantity and the total shortage quantity of item i in site j at period t , and the right-hand side is the total demand quantity at the same period. Note that the demand quantity DQ_{tijk} can not be controlled directly since the simulation calculates it. A simple way to reduce the total shortage quantity is to increase the inventory quantity of the shorted

item, site, and period as follows:

$$IQ_{tij} \leftarrow IQ_{tij} + \sum_{k \in \mathbf{J}_{ij}^{Suc}} GQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.24)$$

Now, we can assume that IQ_{tij} takes similar values to EIQ (IQ_{tij}^{Est}) since both are calculated based on the same policy variables. In addition, Eq. 4.22 indicates that EIQ s become close to IIQ s. Therefore, Eq. 4.24 can be rewritten with this approximation ($IQ_{tij} \approx IQ_{tij}^{Est} \approx IQ_{tij}^{Ideal}$) as follows:

$$IQ_{tij}^{Ideal} \leftarrow IQ_{tij}^{Ideal} + \sum_{k \in \mathbf{J}_{ij}^{Suc}} GQ_{tijk} \quad (\forall t \in \mathbf{T}, i \in \mathbf{I}, j \in \mathbf{J}_W) \quad (4.25)$$

In our proposed method, the IIQ s are updated as Eq. 4.25, and PVE is executed again for shorted items. Phase 2 – 3 are continued until all items satisfy the shortage constraint, and the policy variables are returned as the solution to the target CDJRP-TC when all items satisfy the shortage constraints.

4.6 Computational Experiments

We conducted numerical experiments to investigate the effectiveness of the proposed method and compared it with conventional optimization approaches MILP and SBO.

4.6.1 Conditions

In the following experiments, we consider the CDJRP-TC of the supply chain with three suppliers, one warehouse, and one market shown in Figure 4.1. In these problems, the policy variables of the (s, S) inventory policy for each item and period in a warehouse ($\{s_{tij}, S_{tij}\} \mid \forall t, i, j \in \mathbf{J}_W$) are optimized to minimize the total cost while satisfying all market demands. As described in Section 3.1, the policy variables can be updated for each item and each week. In a week, the policy variables of each item (s, S) take the same value, and the planning horizon T is 91 days (13 weeks or three months). Therefore, the number of policy variables to be optimized is $26 \times T \times N$, where N is the number of items.

Deterministic-dynamic demands and item properties were randomly generated as follows. For the sake of simplicity, we use $U[a, b]$ to denote the corresponding uniform distribution.

- Unit volume of item i : $V_i^{Item} \sim U[0.5, 4.0]$,
- Supplier of each item to the warehouse is randomly selected in the following ratio, [Supplier 1, Supplier 2, Supplier 3] = [15%, 25%, 60%],

- Demand quantity of item i in period t : D_{tik}
 - Demand basis of item i : $\bar{D}_i \sim U[0, 10000]$,
 - D_{tik} if period t is in the 1st or 2nd month: $D_{tik} \sim U[0, \bar{D}_i]$,
 - D_{tik} if period t is in the 3rd month: $D_{tik} \sim U[0, 1.3 \times \bar{D}_i]$.

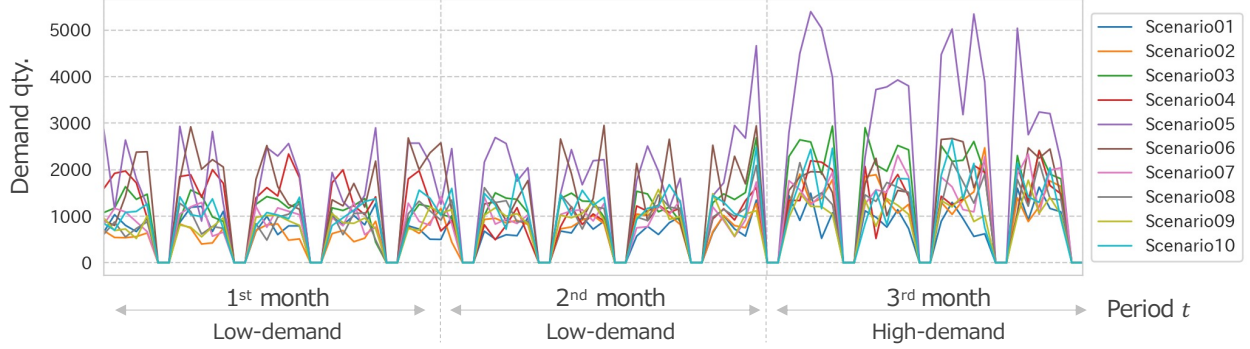


Figure 4.7. Ten random demand patterns, in which the first and second months are low-demand seasons and the third month high-demand.

Figure 4.7 shows the ten patterns of the time-series total demands of ten items generated as mentioned above. The other problem settings of CDJRP-TC are set as follows; $W_{ij}^{Labor}=0.003$, $W_{ij}^{Max}=150$, $C_{ij}^{Labor}=5$, $C_j^{Inv}=2$, $V_j^{Fix}=100,000$, $V_v^{Trans}=1,700$. Additionally, the proposed method's hyper-parameters were set as follows; The time limit of Phase 1 was 60 seconds. The CMA-ES population was 48, the upper limit of iteration was 200, and $\omega=20$. The computing environment was a Ryzen 3950X CPU (3.70GHz, 32threads) and 128GB RAM, and all MILP models were solved by CPLEX 12.8.

4.6.2 Optimization Result Comparison

In this experiment, we compared the optimization results of the CDJRP-TCs using the proposed and conventional methods. We evaluated the three problem sizes of CDJRP-TCs with 10, 50, and 100 items. Ten problem instances with different randomly generated demands were solved for each problem size, and the averages and standard deviations of the results were compared. Their time limits were set as follows: 100 sec. for 10-item problems, 1800 sec. for 50-item problems, and 3600 sec. for 100-item problems. Table 4.6 shows the transportation conditions; unit transportation costs C_{tjkv}^{Trans} and the upper limit number of the vehicles N_{tjkv}^{Trans} . All problems must satisfy the shortage constraint Eq.4.5. Therefore, there is no feasible solution when the shortage rate exceeds zero.

As conventional methods, we used the SBO approach with GA and MILP, respectively. In the SBO approach, the problems were formulated with the simulator described in Subsection 4.4.4 and solved by GA with the following settings: the population size is set to 100, and uniform crossover

Table 4.6. Transportation conditions of optimization runtimes comparison

From	To	Vehicle	Month	Upper num. of vehicles	Unit trans. cost
Supplier 1	Warehouse	Normal	1st – 3rd	1	200
		Extra	1st – 2nd	4	240
			3rd	4	360
Supplier 2	Warehouse	Normal	1st – 3rd	3	150
		Extra	1st – 2nd	7	180
			3rd	7	270
Supplier 3	Warehouse	Normal	1st – 3rd	3	100
		Extra	1st – 2nd	9	120
			3rd	9	180
Warehouse	Market	Normal	1st – 3rd	Unlimited	0

and Gaussian mutation are set as genetic operators as proposed by Hochmuth, Lässig, and Thiem [124]. In the MILP approach, the problems formulated as the MILP model in Subsection 4.4.3 were directly solved with the commercial MILP solver. In addition, we investigated the optimality gap by comparing each solution's total cost with the analytical optimal solution obtained by MILP run for 3600 seconds. Due to the problem size, we could obtain the analytically optimal solution and report the optimality gap successfully only for Experiment 1.

The computational results are shown in Table 4.7. First, we discuss the result of Experiment 1, which contains ten items. Table 4.7 shows that the proposed method solved all problem instances, whereas GA solved seven instances, and MILP only solved two instances. The total cost of the MILP solutions was the lowest of all methods, but its shortage rate was the worst.

Table 4.7. Optimization results of CDJRP-TC with 10, 50, and 100 items

Setting	Method	Solved instance	Shortage rate [%]	Total cost [$\times 10^3$]	Optimality gap[%]	Runtime [sec]
Experiment 1 10 items ≤ 100 sec.	Proposed	10/10	0.0 ± 0.0	49.7 ± 11.0	11.9 ± 12.9	72 ± 26
	GA	7/10	0.02 ± 0.04	52.4 ± 12.7	16.2 ± 12.6	100
	MILP	2/10	8.3 ± 7.8	40.4 ± 9.3	24.8 ± 29.0	100
Experiment 2 50 items ≤ 1800 sec.	Proposed	10/10	0.0 ± 0.0	145.3 ± 20.2	-	75 ± 11
	GA	0/10	0.09 ± 0.04	169.1 ± 28.0	-	1800
	MILP	0/10	56.7 ± 9.2	63.7 ± 13.4	-	1800
Experiment 3 100 items ≤ 3600 sec.	Proposed	10/10	0.0 ± 0.0	298.9 ± 18.2	-	295 ± 88
	GA	0/10	0.41 ± 0.01	342.2 ± 20.0	-	3600
	MILP	0/10	58.8 ± 3.2	109.7 ± 8.6	-	3600

The total costs and the shortage rates of each instance are shown in Figure 4.8. In these figures, the x -axis means the problem instance ID and the y -axis of the upper chart represents the total cost, and that of the lower chart is the shortage rate in the log scale. In these figures, when no feasible solution was obtained in the problem instance, the result is illustrated as a hatched bar, which indicates the result of the best infeasible solution. Figure 4.8 shows that GA obtained six feasible solutions. MILP obtained two feasible solutions with the lowest objective values, while it returned infeasible solutions for the other eight instances. As shown in Figure 4.8, MILP infeasible solutions have larger shortage rates than those of the other two methods. These results indicate that the MILP is an unstable approach to solving CDJRP-TCs. On the other hand, the proposed method obtained feasible solutions for all problem instances, as well as feasible solutions with the lowest objective values of all methods in the seven instances.

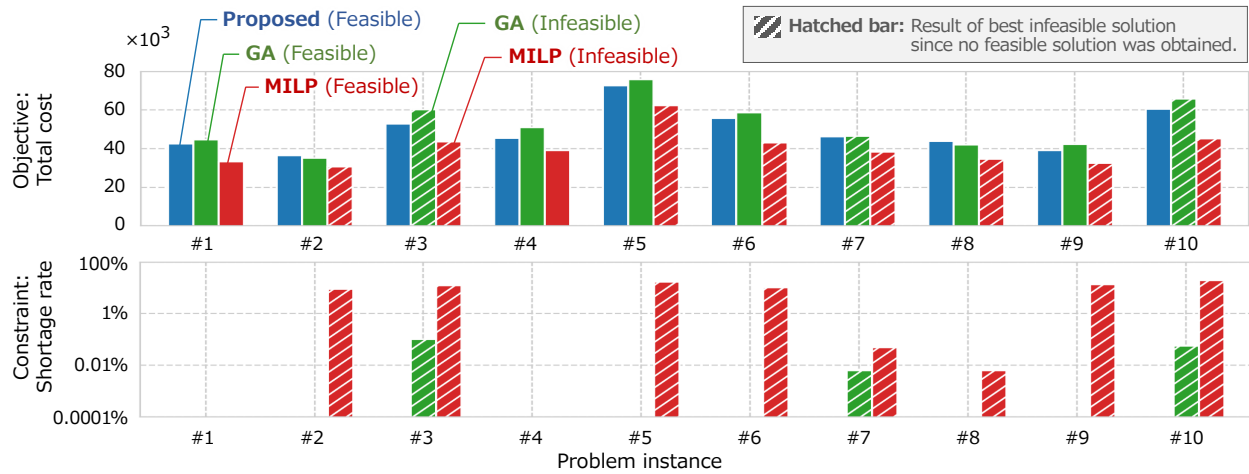


Figure 4.8. Objective and constraint value comparison of 10-item CDJRP-TC.

Next, we analyzed problem instance #1 in detail. The time-series total inventory quantities obtained by each method are shown in Figure 4.9. In this figure, the x -axis indicates period t , and the y -axis represents each date's total inventory and demand quantities. The dotted line is the demand quantities. The blue line is the total inventory quantities obtained by the proposed method, the green line is that of GA, and the red line is that of MILP. This figure shows that the inventory quantities take the largest values for MILP, the proposed method, and GA in that order, which is the same order as their objective values shown in Table 4.7. This indicates that the more inventories are replenished in the first and second months (when unit transportation costs are low), the fewer inventories will need to be replenished in the third month (when unit transportation costs are high), leading to a lower total cost. This relationship can also be seen in Figure 4.10, which shows the total number of monthly used vehicles of each method's solution. This figure shows that only the GA solution used the extra vehicle in the third month, and the fourth chart, which illustrates the

total vehicle quantities for all periods, shows that GA used more vehicles. These results suggest that MILP and the proposed method can obtain solutions that control inventory replenishment according to the change in unit transportation costs.

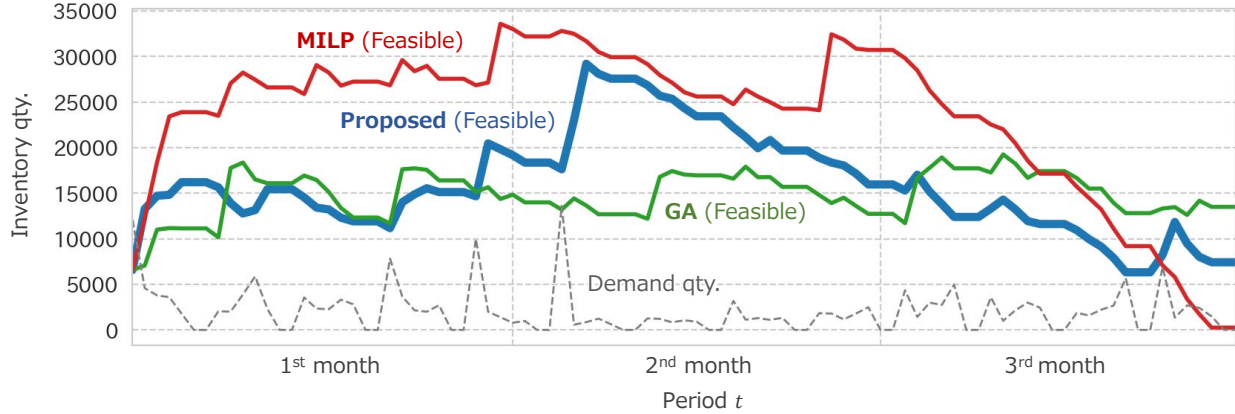


Figure 4.9. Time-series total inventory quantities in 10-item CDJRP-TC instance #1.

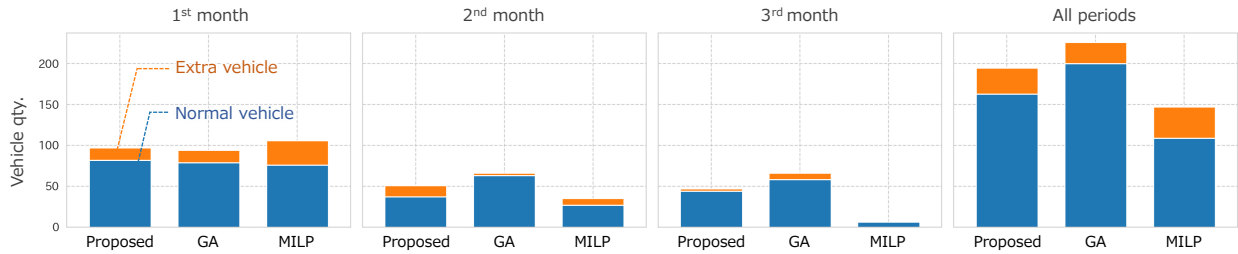


Figure 4.10. Monthly used vehicle quantities in 10 item-CDJRP-TC instance #1.

Next, we discuss the results of Experiments 2 and 3, which contain 50 and 100 items, respectively. Table 4.7 shows that the only proposed method obtained feasible solutions for all problem instances, whereas GA and MILP could not find any feasible solutions within the time limits. The total costs and shortage rates of each instance in 50-item and 100-item CDJRP-TCs are shown in Figure 4.11 and Figure 4.12, respectively. These figures show that GA could not obtain feasible solutions in any instance; moreover, the objective values of GA's infeasible solutions were larger than those of the feasible solutions obtained by the proposed method. Additionally, the infeasible solutions obtained by MILP have much higher shortage rates than those of the proposed method and GA. It indicates that the MILP approach can only be used for small-scale problems. In terms of the runtime to obtain feasible solutions, the proposed method could find feasible solutions within 75 seconds for 50-item cases and 295 seconds for 100-item cases on average, while GA and MILP could not find any feasible solutions within 1800 and 3600 seconds, respectively. From this result, it can be seen that the proposed method obtains feasible solutions in at least 12 times shorter

runtimes than GA and MILP for large-scale CDJRP-TCs. It indicates that the heuristic approach in the proposed method is effective for finding feasible solutions for large-scale CDJRP-TCs. These results suggest that the proposed method is the most scalable approach to solving CDJRP-TCs of the conventional methods.

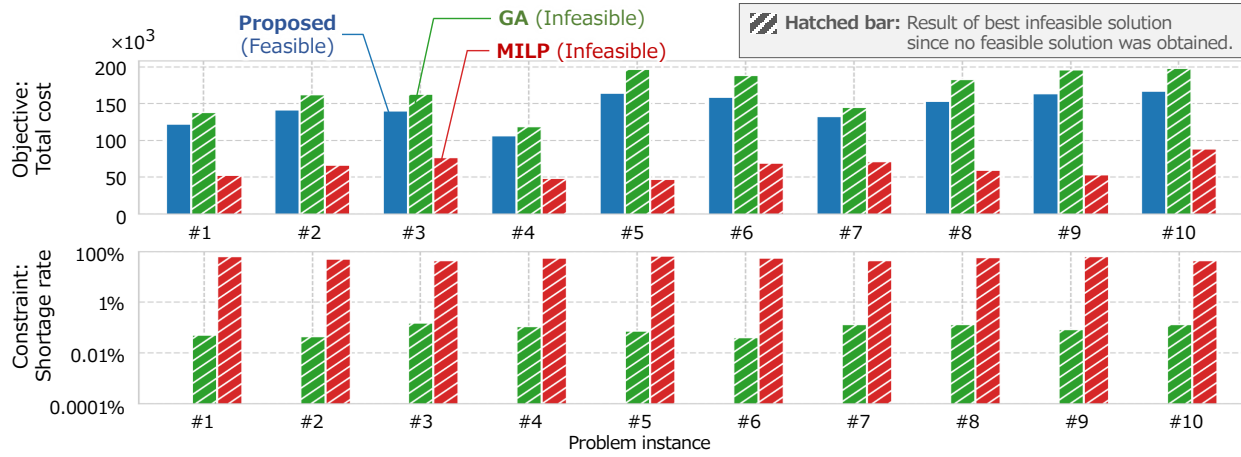


Figure 4.11. Objective and constraint value comparison of 50-item CDJRP-TC.

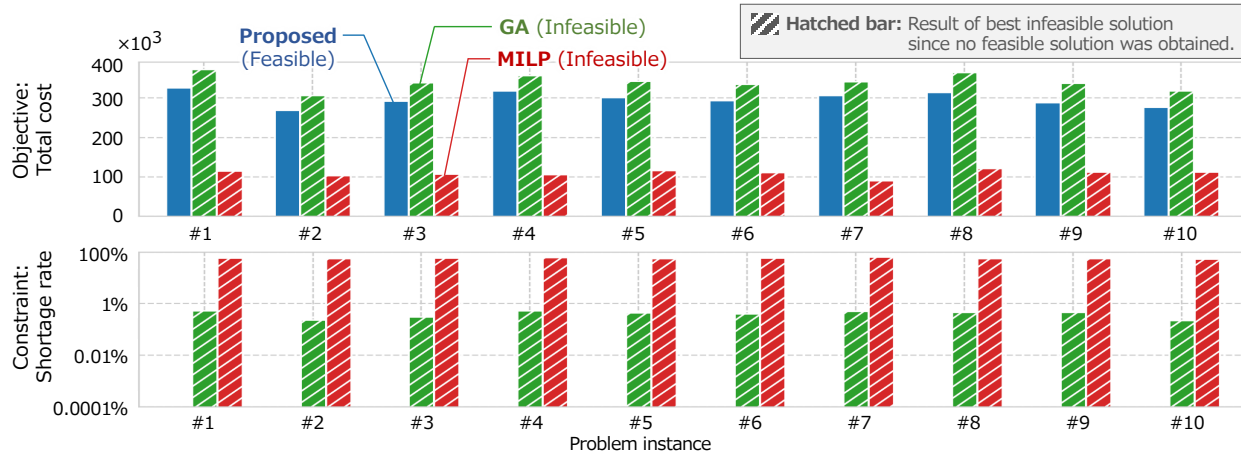


Figure 4.12. Objective and constraint value comparison of 100-item CDJRP-TC.

4.6.3 Planning Capability for Different Unit Transportation Costs

In this section, we investigate the proposed method's capability to obtain the appropriate solution according to the increase in unit transportation costs. We conducted the 100-item CDJRP-TCs with different patterns of the cost increase rate. Table 4.8 shows the three patterns of transportation conditions. Each pattern has a different cost increase rate of the unit transportation costs. + 0% means that unit costs do not increase in the third month. + 50% means that unit costs increase

50% in the third month from the first and second month. In the same way, + 100% means that unit costs increase 100% in the third month. The other problem settings are the same as the 100-item CDJRP-TCs described in Subsection 4.6.2. Using the proposed method, we randomly generated ten demands and solved the 100-item CDJRP-TCs with these demands and the transportation conditions with each cost increase rate.

Table 4.8. Transportation conditions of different cost increase patterns.

From	To	Vehicle	Month	Upper num. of vehicles	Unit trans. cost increase		
					+ 0%	+ 50%	+ 100%
Supplier 1	Warehouse	Normal	1st – 3rd	1	200	200	200
		Extra	1st – 2nd	4	240	240	240
			3rd	4	240	360	480
Supplier 2	Warehouse	Normal	1st – 3rd	3	150	150	150
		Extra	1st – 2nd	7	180	180	180
			3rd	7	180	270	360
Supplier 3	Warehouse	Normal	1st – 3rd	3	100	100	100
		Extra	1st – 2nd	9	120	120	120
			3rd	9	120	180	240

Table 4.9 shows the normalized results of the 100-item CDJRP-TCs with each cost increase rate. This table shows the average inventory quantity (AIQ) and the total vehicle quantity (TVQ), normalized by each result of the first month with + 0% cost increase rate. This table shows that when the cost increase rate is higher, the AIQs of the second and third months become larger, and the TVQs of those months become smaller. For example, the third month's AIQs with + 50% and + 100% cost increase rates are 17.0% and 62.3% larger than that with + 0% cost increase rate, respectively. In the same period, the TVQs are reduced by 5.1% in + 50% case and by 30.3% in + 100% compared to + 0% case. These results indicate that the unit transportation cost has a significant impact on distribution planning.

Table 4.9. Optimization result comparison with different unit cost increase rates.

Unit trans. cost increase	Normalized avg. inventory qty.				Normalized total vehicle qty.			
	Month:	1st	2nd	3rd	Month:	1st	2nd	3rd
+ 0%		1.00	1.06	1.06		1.00	0.93	0.99
+ 50%		1.10	1.24	1.24		1.10	0.91	0.94
+ 100%		1.10	1.56	1.72		1.04	1.19	0.69

Next, we focused on one problem instance and analyzed it in detail. The time-series total inventory quantities obtained by each cost increase rate are shown in Figure 4.13. In this figure, the

x -axis indicates period t , and the y -axis indicates each date's total inventory and demand quantities. The dotted line is the demand quantities. The green, blue, and red lines are the inventory quantities of $+0\%$, $+50\%$, $+100\%$ cost increase cases. This figure shows that when the cost increase rate is higher, more inventories are stored at the end of the second month to reduce replenishment in the third month. Figure 4.14 also illustrates that the more extra vehicles are used in the first and second months, the fewer extra vehicles are used in the third month when the cost increase rate is higher.

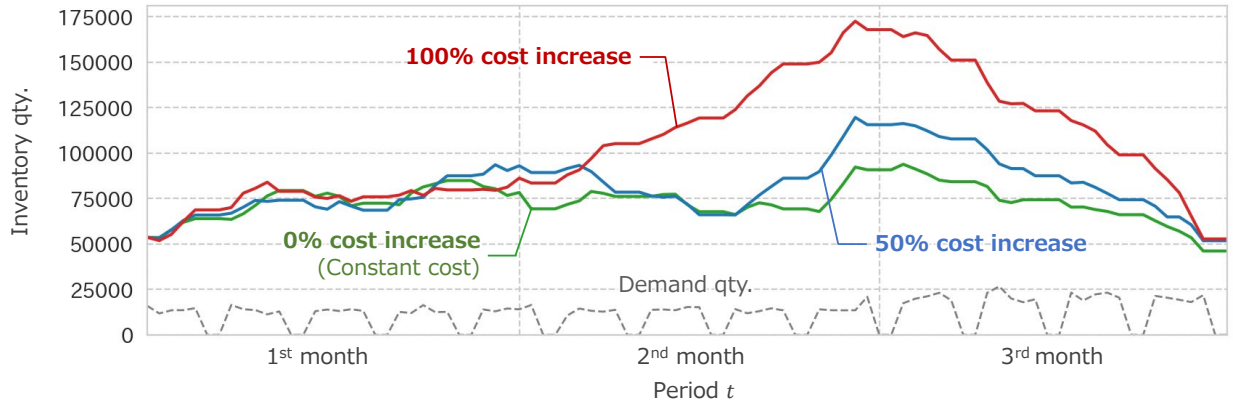


Figure 4.13. Time-series total inventory quantities for each cost increase rate.

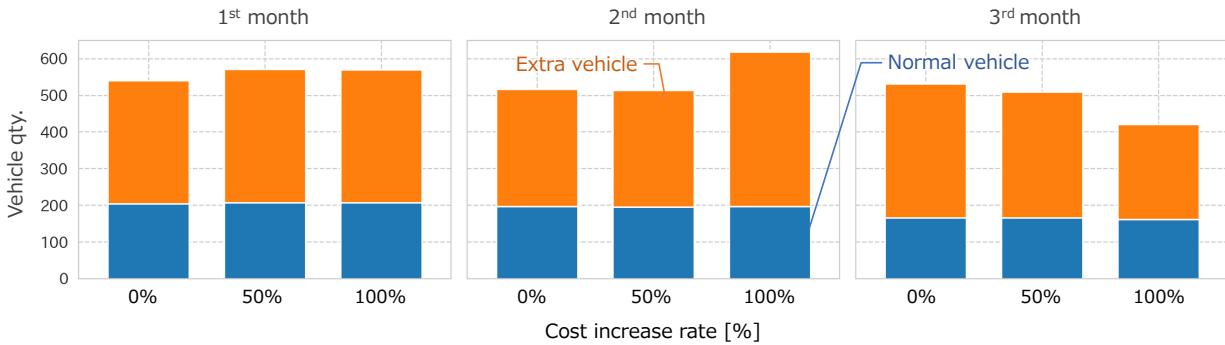


Figure 4.14. Monthly used vehicle quantities from all suppliers for each cost increase rate.

Figure 4.15 illustrates the monthly vehicle quantities used to replenish from Supplier 1, 2, and 3 with each cost increase rate. From this figure, the above relationship can be seen more clearly in the increased unit transportation cost from each supplier. Replenishment from Supplier 1, whose unit transportation cost is the highest of all suppliers, changes most drastically with the cost increase rate. This is because the transportation cost from such a supplier has a larger impact on the total cost than other suppliers. These results demonstrate that the proposed method can obtain the appropriate solution according to increases in unit transportation costs.

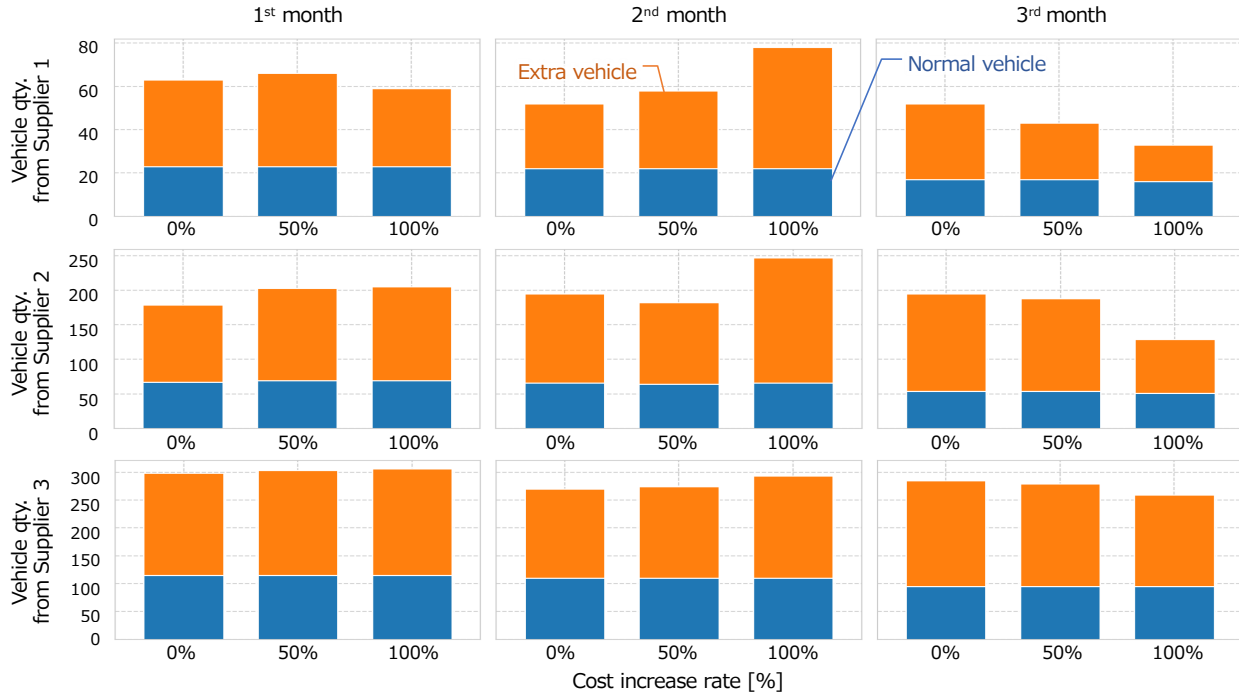


Figure 4.15. Monthly used vehicle quantities from supplier 1 – 3 for each cost increase rate.

4.7 Conclusion

In this paper, we investigated the formulation and solution approach to support distribution planning in the sales and operations planning (S&OP) process. To consider dynamic-deterministic demands obtained as the sales plan and time-varying unit costs of vehicles owned by third-party logistics providers, we formulate this problem as the capacitated dynamic-demand JRP with time-varying costs (CDJRP-TC), and proposed a three-phase solution approach based on the hybrid simulation & optimization. Numerical experiments with up to 100 items demonstrate that the proposed method could obtain feasible solutions within at least 12 times shorter runtimes than the conventional methods (GA, MILP) through computational experiments. Additionally, we investigated the capability of the proposed method to obtain the appropriate solutions according to the cost increase rate of unit transportation cost.

From these computational results, we can obtain the following managerial insights for the distribution planning under the unit cost increase:

1. For the distribution planning to minimize the total cost, inventory replenishment in the period with the low unit transportation cost should be increased to reduce the number of transportation vehicles in the period with the high unit cost and suppress the transportation cost.
2. The appropriate amount of inventory replenishment in the low unit cost periods depends

on the cost increase rate, and the higher the rate, the larger amount of replenishment is recommended while satisfying the capacity constraints and increase of other costs.

In future work, we can consider improving the policy variable estimation in Phase 2 and updating IIQs in Phase 3 to solve larger CDJRP-TCs, for example, by using machine learning techniques.

Fast Simulation-based Order Sequence Optimization Assisted by Pre-trained Bayesian Recurrent Neural Network

5.1 Introduction

Sequence optimization problem (SOP) or sequencing problem is a combinatorial optimization problem (COP) to find an appropriate sequence of tasks to minimize or maximize the given objective function. SOP can be seen in various practical fields such as traveling salesman problem (TSP), vehicle routing problem (VRP), and other job scheduling problems. In logistics warehouses, SOP can take the form of the scheduling of order picking activity, where the largest workforce is allocated. For example, in the most common order picking system (picker-to-parts), order pickers physically walk into a warehouse to pick items [136], and the associated labor costs account for about 55% of the total fulfillment operation cost [137]. Recently, various robotic systems have been developed to automate order picking. Among these, automated storage and retrieval systems (AS/RS) [138], automated guided vehicle (AGV) picking systems [139], [140] and piece picking robots [141], [142] are commonly employed.

In this paper, we consider the AGV picking system with robot arms and conveyor modules shown in Figure 5.1, which is entitled Pick-Place-Carry-Pick-Place (PPCPP) system. Figure 5.2 shows the detailed layout. The PPCPP system contains multiple piece-picking robots that can deal with various picking orders in parallel. When the warehouse management system (WMS) sends picking order information to the PPCPP system, the picking order is allocated to one of the available piece-picking robots. Then AGVs start transferring the movable storage shelves where ordered items are stored inside the storage boxes. The feeding robots *pick* the transferred storage boxes from the shelves and *place* them on the feeding conveyor. Then the feeding conveyor *carries* the storage boxes in front of the piece-picking robots. The piece-picking robots *pick* items from storage boxes on the feeding conveyor and *place* them into shipping boxes on the shipping conveyor. When a shipping box contains all the requested items, the shipping conveyor transfers the box to the following process, and a new picking order is immediately assigned to the piece-picking robot.

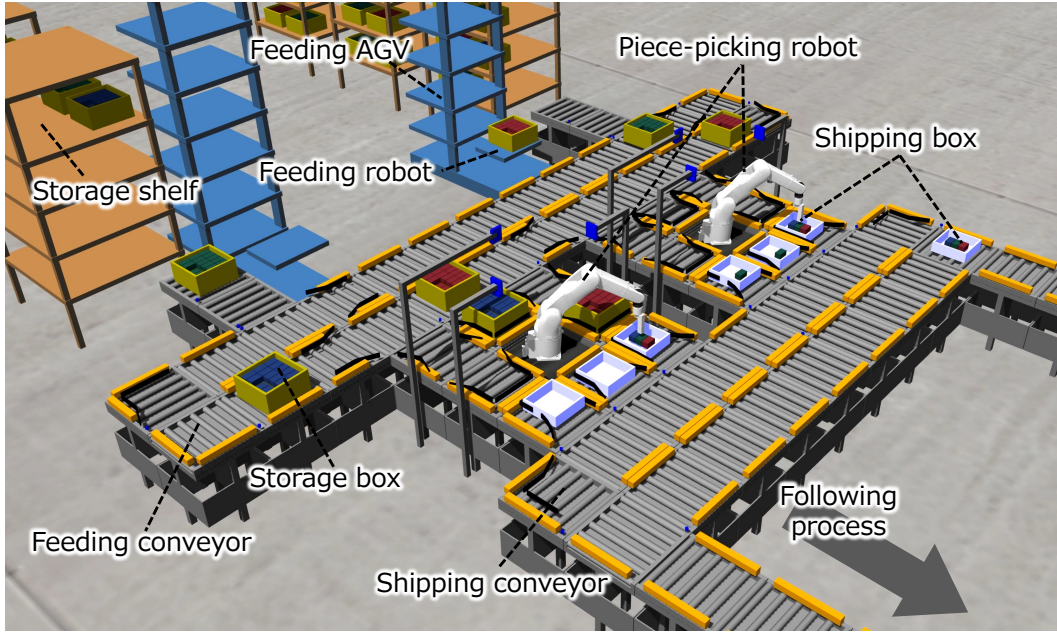


Figure 5.1. Pick-Place-Carry-Pick-Place (PPCPP) order picking system

PPCPP systems can achieve high scalability since the number of AGVs, robot arms, and other components can be customized easily.

For achieving high productivity, appropriate job scheduling is vital [143]. In the PPCPP system, the operation sequence of the picking orders from the WMS strongly affects the *makespan*, defined as the duration from the initial order placement to the end of the final order shipping. Generally, the makespan can be reduced by continuously picking similar orders with overlapped items since transportation of these items' storages by AGVs and conveyors can be aggregated. The order sequence can also affect the occurrence of *deadlock*, which is a situation where no task can be achieved because of stacked system modules. An example situation is when inputting picking orders that require more kinds of items than the feeding conveyor capacity at the same time. All feeding conveyors can be occupied before completing these orders in this situation. Once such a situation happens, it increases the operation time due to system stoppage and is hard to restore automatically. Therefore, the order sequence that can lead to deadlock must be avoided.

In this paper, we focus on the order sequencing problem (OSP) of PPCPP systems that determines the optimal order sequence $\mathbf{X}$ of N picking orders $\mathbf{O} = \{O_n | n = 1, 2, \dots, N\}$ for minimizing the makespan under the deadlock avoidance constraint. For example, if $N = 3$, the possible $\mathbf{X}$ is $\{O_1, O_2, O_3\}, \{O_1, O_3, O_2\}, \dots, \{O_3, O_2, O_1\}$. Simulation is essential for solving the real SOPs (such as the OSP of PPCPP systems) because the performance measures are hard to formulate mathematically. Therefore, many researchers and practitioners have solved SOPs with simulation-based optimization (SBO) methods using metaheuristics[144]–[148]. However, these

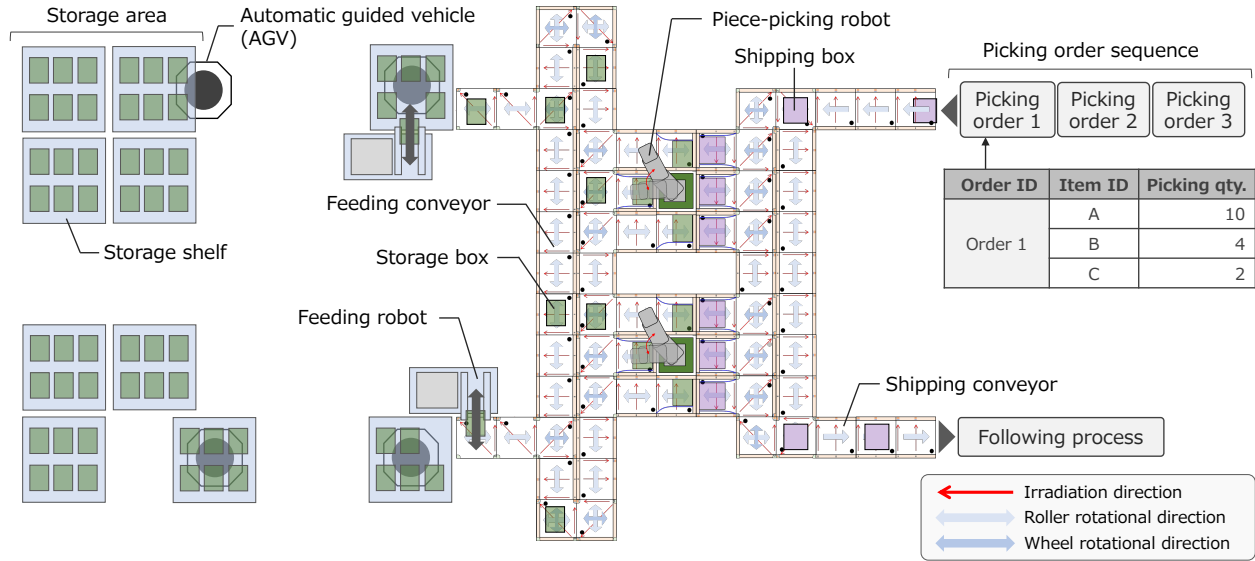


Figure 5.2. PPCPP system layout. The system consists of AGVs, feeding robots, piece-picking robots, and conveyors. The system input (picking orders sequence) strongly affects the performances, such as the makespan and the deadlock occurrence.

approaches are usually computationally expensive because they require iterative simulation runs, which generally take a few seconds to several minutes for one execution.

Surrogate-assisted optimization (SAO) is an alternative approach to solving computationally expensive SBO problems. Figure 5.3 shows the flowchart of SAO, where the fundamental idea is to approximate the performance measures using a *surrogate model* or *metamodel* whose computational cost is relatively lower than simulation. Especially, Bayesian optimization (BO) is the most popular one, and several BO variants for binary optimization problems have been developed recently [149], [150]. However, it remains challenging for these methods to solve SOPs because enormous numbers of binary variables and constraints must be considered. Additionally, BO requires many simulations to train an accurate surrogate model *during optimization*. Thus, since conventional approaches cannot solve simulation-based SOPs in a short amount of time, flexible updating of picking schedules is impossible to achieve.

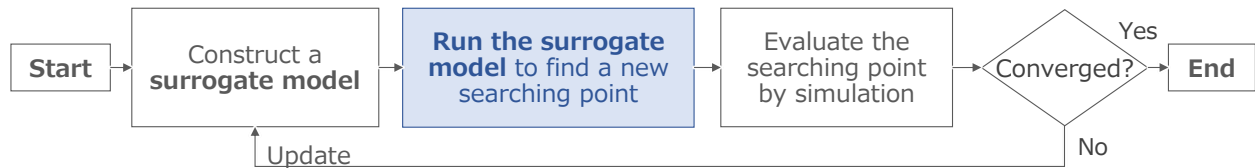


Figure 5.3. Flowchart of surrogate-assisted optimization. It can solve the simulation-based optimization problem efficiently by approximating simulation with a low-computational-cost surrogate model.

For quickly solving simulation-based SOPs, we have developed a new sequence optimization method that uses a surrogate model trained *before optimization* (*pre-trained*). In practical OSPs, most picking system configurations cannot be changed frequently, and the distribution of future picking orders can be forecasted using historical data. Therefore it is possible to train the surrogate model with a dataset randomly generated from the forecasted distribution. We utilize a Bayesian recurrent neural network (BRNN) as a surrogate model to learn the sequential nature of OSPs. Our proposed method, BRNN-SA, combines simulated annealing (SA) as the basic optimization strategy and BRNN as the surrogate model in the optimization.

The purpose of this paper is to answer the following research questions.

- RQ1: Which surrogate model is most suitable for the OSPs of PPCPP systems?
- RQ2: How quickly can the proposed optimization method (BRNN-SA) solve the OSPs of PPCPP systems compared to conventional metaheuristics?

The contributions of this paper are as follows.

- Proposal of the Bayesian recurrent neural network-assisted simulated annealing (BRNN-SA) method.
- Investigation and selection of the proper surrogate model for the OSPs of PPCPP systems.
- Evaluation of BRNN-SA demonstrating its superiority in solving the OSPs of PPCPP systems compared with conventional metaheuristics.

In Section II, we explain the background of this study. Section III describes the BRNN-SA for OSPs. In Section IV, we evaluate the performance of the proposed BRNN-SA. We conclude in Section V with a brief summary and mention of future work.

5.2 Related Works

5.2.1 Order Sequencing Problem

The order sequencing problem (OSP) is a job scheduling process[151] that is categorized as a sequence optimization problem (SOP), which is a type of combinatorial optimization problem (COP). As OSP is a vital decision-making task for efficient picking operations in a logistics warehouse, many studies have investigated OSP for order fulfillment systems. Boysen et al. [152] formulated the OSP of the consolidation and packing area as a mixed-integer linear programming (MILP) model and solved it with a heuristic approach and dynamic programming. Pinto et al. [153] solved the order batching and sequencing problem by an iterative method of two Genetic algorithms (GA) that solve order batching and order sequencing separately.

In these OSPs and other ordinary job scheduling problems, the time needed for each operation and transportation is given to the planner as constant input values. However, in our problem setting, these values are difficult to determine mathematically since they can change depending on the AGV and conveyor status at a given time. In PPCPP systems, storage feeding for multiple picking orders can be aggregated in one AGV transportation, and it is nearly impossible to calculate the operation and transportation times in all possible situations. Therefore, instead of defining them as constant input values, we evaluate the makespan and the deadlock occurrence with the PPCPP simulator, whose input is an order sequence $\mathbf{X}$.

5.2.2 Simulation-based Combinatorial Optimization

SOPs like those above have been solved as simulation-based optimization (SBO) problems, where simulation models evaluate the system performance under a given configuration. Metaheuristic methods are widely utilized in this context, especially for solving simulation-based combinatorial optimization problems (SBCOP). Rouky et al. [145] proposed an ant colony optimization (ACO) hybridized with a variable neighborhood descent local search to solve the task sequences of quay cranes. Uman et al. [146] tackled the flow shop scheduling problem and proposed the tabu search (TS) process with a genetic algorithm (GA) to minimize makespan. Another popular metaheuristic to solve SOPs is simulated annealing (SA). Cheng et al. [147] proposed an extended SA algorithm to minimize the makespan of a permutation flowshop scheduling problem (PFSSP). Khurshid et al. [148] also tackled PFSSP by using an improved evolution strategy that harnesses local search abilities of SA.

While these SBO methods may effectively solve analytically intractable SOPs, they are usually computationally expensive because iterative simulation runs (generally at least 100 or more) are essential to calculate the performance measures of each set of parameters. Realistic simulations like the PPCPP simulator take anywhere from a few seconds to several minutes for one execution, which limits the application of metaheuristics for OSPs to situations with enough calculation time.

5.2.3 Surrogate-assisted Optimization

Surrogate-assisted optimization (SAO) is an alternative approach to solving computationally expensive SBO. The fundamental idea of SAO is to approximate the performance measures by a *surrogate model* whose computational cost is relatively lower than simulation. As surrogate models, linear regression (LR)[61], support vector machine (SVM)[154], gradient boosting trees (GBT)[63], artificial neural networks (NN)[155], and Gaussian process (GP)[95] are utilized. Bayesian optimization (BO), which uses GP as a surrogate model, is one of the most popular SAO methods. Although the original BO can solve only continuous optimization problems, more recent BO variants for solving

SBCOP have been developed[156]. For example, COMBO (Oh et al.[149]) and MerCBO (Deshwal et al.[150]) are state-of-the-art BO methods to solve binary optimization problems.

However, it is still challenging to use these methods for solving simulation-based SOPs. SOP can be reformulated as a binary optimization problem, but the reformulated problem typically contains an enormous number of binary variables and constraints to eliminate overlaps or sub-tours. Generally, such problems are difficult to solve with metaheuristics and BO. Additionally, when solving a large-scale problem, BO requires many simulations to train the surrogate model accurately *during optimization*. Thus, conventional approaches cannot quickly solve simulation-based SOPs enough to update picking schedules flexibly. For reducing optimization runtime, we propose a new SAO approach that uses a surrogate model trained *before optimization*.

5.3 Methodology

5.3.1 Problem Setting and Formulation

We investigate the OSP of PPCPP systems, which determines the optimal sequence of the given N picking orders $\mathbf{O} = \{O_n | n = 1, 2, \dots, N\}$ for minimizing the makespan without system deadlock. The makespan and deadlock occurrence are obtained as simulation results. The notations and decision variables are listed in Table 5.1.

Table 5.1. Notations and sets.

Notation	Description
$\mathbf{X}$	Decision variables, which indicate an order sequence.
x_i	i -th picking order in order sequence ($i = 1, 2, \dots, N$).
O_n	n -th customer's picking order ($n = 1, 2, \dots, N$).
$F(\mathbf{X})$	Makespan to finish given order sequence $\mathbf{X}$, where F is implemented as a simulator.
$G(\mathbf{X})$	Deadlock occurrence of order sequence $\mathbf{X}$ (1 if a deadlock occurs, 0 otherwise), where G is implemented as a simulator.
$f(\mathbf{X})$	Predicted makespan of order sequence $\mathbf{X}$ by surrogate model f .
$g(\mathbf{X})$	Predicted deadlock occurrence of order sequence $\mathbf{X}$ by the surrogate model g (1 if a deadlock occurs, 0 otherwise).

The mathematical formulation of the OSP is as follows:

$$\min_{\mathbf{X}} F(\mathbf{X}) \quad (5.1)$$

$$\text{s.t. } G(\mathbf{X}) = 0 \quad (5.2)$$

$$\mathbf{X} = \{x_1, x_2, \dots, x_N\} \quad (5.3)$$

$$x_i \in \{O_1, O_2, \dots, O_N\} \quad \forall i \quad (5.4)$$

$$x_i \neq x_j \quad \forall i \neq j \quad (5.5)$$

Equation 5.1 shows the objective function $F(\mathbf{X})$, which is the minimization of the makespan to finish the given order sequence $\mathbf{X}$. Equation 5.2 describes the constraint to avoid deadlock $G(\mathbf{X})$. Equation 5.3, 5.4, and 5.5 ensure that the decision variable $\mathbf{X}$ is a non-overlapped sequence of each customer's picking order $\mathbf{O}$. As described in Section 1, $F(\mathbf{X})$ and $G(\mathbf{X})$ are difficult to define as mathematical equations. We, therefore, calculate them with a PPCPP simulator.

In this paper, we investigate practical OSPs in logistics warehouses. In general, OSPs are solved with only the target OSP information, which is available just before solving the problem. In this case, the target problem information is the picking orders from customers (shipping orders) and the PPCPP system configuration. However, in real situations, we can utilize other information to solve the target OSP, such as the historical picking order information and the current system configuration. In this work, we assume the following two conditions;

- Rather than assuming that the picking orders from customers are only available just before the picking operation begins, we assume that the distribution of historical picking orders is always available.
- The system configurations do not change from the preparation to the picking operations because the configurations of practical industrial systems cannot be changed so frequently.

In the following sections, we discuss how to solve the OSPs of PPCPP systems under these assumptions.

5.3.2 Surrogate Pruning Optimization Framework

The overall flowchart of the proposed surrogate pruning optimization framework is shown in Figure 5.4. This method consists of two phases: pre-training and optimization. In the pre-training phase, random picking orders are generated in accordance with the distribution obtained by the historical picking orders. Then, the surrogate models of the performance measures, namely, the makespan $f(\mathbf{X})$ and the deadlock $g(\mathbf{X})$, are trained to predict the simulated makespan $F(\mathbf{X})$ and deadlock $G(\mathbf{X})$ respectively. In the optimization phase, surrogate models are used to choose

good candidate solutions quickly without simulations. The selected candidate solutions are then evaluated by simulation. This *surrogate pruning* enables us to reduce computational time to evaluate performance measures by simulation. These steps are repeated until the runtime reaches the time limit.

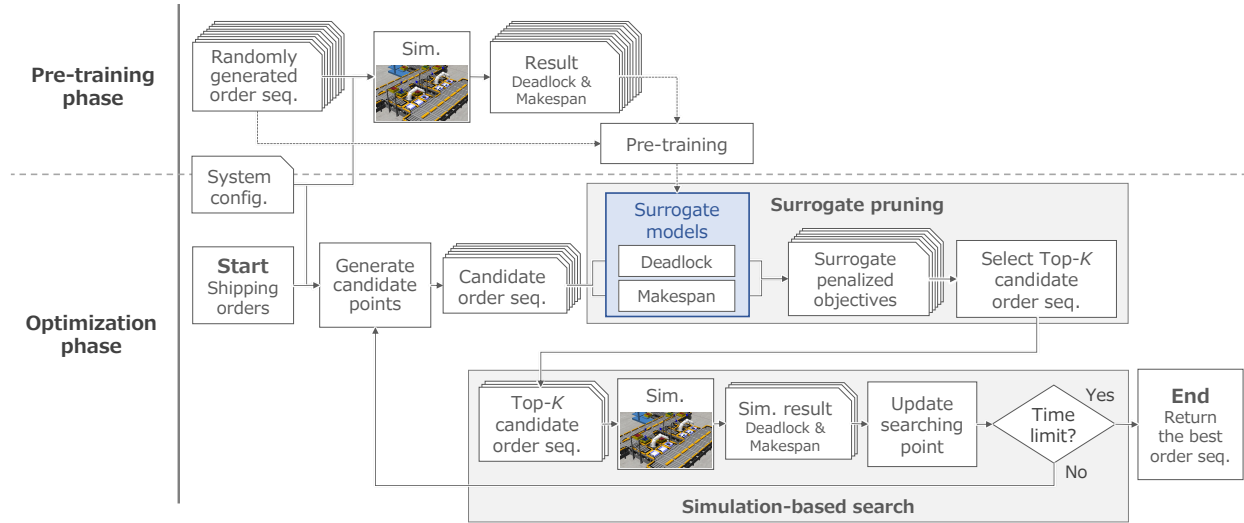


Figure 5.4. Flowchart of surrogate pruning optimization framework. Surrogate models of the performance measures are trained in the pre-training phase and utilized to choose the top- K candidate solutions in the surrogate pruning of the optimization phase.

5.3.3 Selection of Surrogate Models

There are two requirements for the surrogate models for OSPs ($f(\mathbf{X})$ and $g(\mathbf{X})$) as follows.

Requirement 1: The surrogate models must find the appropriate features to infer the performance measures automatically since manual feature engineering of a complex system such as PPCPP systems is complicated.

Requirement 2: For optimizing the order sequence, the surrogate models must learn how the performance measures change depending on the order sequence.

To meet these requirements, we opted to use a multi-layer perceptron (MLP), which is a powerful machine learning method that automatically extracts features from input, which meets Requirement 1. For dealing with Requirement 2, we use a recurrent neural network (RNN), one of the MLPs that utilizes a recurrent layer to learn the sequential relationship in input features, such as order sequences. We also utilize a Bayesian neural network (BNN) to improve the model generalization. BNN is an MLP model that imposes a prior distribution on the weight parameters and aims to infer a posterior distribution instead of point estimates. Although the output of the PPCPP simulator is deterministic, we utilize BNN because it performs ensemble learning by

marginalizing over this posterior for prediction.

Based on the above discussion, we select a Bayesian recurrent neural network (BRNN), which is a BNN consisting of a recurrent layer, for the surrogate models, as shown in Figure 5.5. However, precisely computing the posterior of non-linear BRNNs is infeasible, so we utilized the Dropout variational inference [157] to train BRNN. It interprets Dropout regularization as approximate inference in BRNN models and learns a variational Dropout posterior over the weight parameters. Prediction results of BRNN are obtained by averaging N_{pred} time inferences. In this work, we use $N_{pred} = 1$ for training and $N_{pred} = 10$ for prediction and optimization. In addition, we use the Adam optimizer [91] to train BRNN. The bidirectional long short-term memory (LSTM) architecture with 64 neurons (n_r) is utilized as the recurrent layer and connected to two dense layers, whose neurons are connected to all neurons in the next layer. Each layer consists of 192 neurons (n_d) with Leaky ReLU as the activation function and has a Dropout layer whose Dropout rate is set to 10% (r_d). All mentioned parameters ($n_r \in [32, 48, 64, 128]$, $n_d \in [64, 128, 192, 256]$, and $r_d \in [5\%, 10\%, 30\%]$) were analyzed and the most accurate model on the test set was picked.

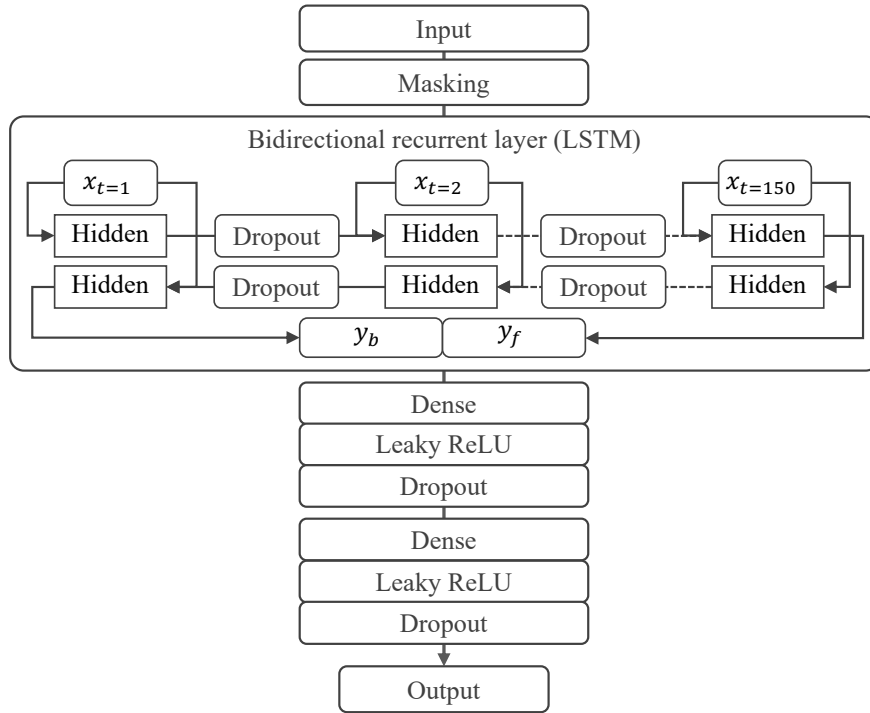


Figure 5.5. The proposed architecture of the Bayesian recurrent neural network. It contains a bidirectional recurrent layer to learn the sequential relationship in input and Dropout layers for the Dropout variational inference.

5.3.4 Proposed Optimization Method: BRNN-SA

In this section, we propose BRNN-SA, which implements the surrogate pruning optimization by combining the simulated annealing (SA) strategy and BRNN as surrogate models. The detailed algorithm is shown in Algorithm 5-1.

Algorithm 5-1 Optimization procedure of BRNN-SA

Input: Picking orders $\mathbf{O} = \{O_1, O_2, \dots, O_N\}$, Penalty ω .

Output: Best order sequence $\mathbf{X}_{\text{best}} = \{x_i\}$.

- 1: Initialize $t \leftarrow 1$ and temperature $T \leftarrow T_0$.
 - 2: Generate M_0 candidate points $\mathbf{X}_{0j}$ ($j = 1, 2, \dots, M_0$)
 - 3: Select an initial searching point $\mathbf{X}_1$ from $\mathbf{X}_{0j}$ for minimizing $h(\mathbf{X}_{0j}) = f(\mathbf{X}_{0j}) + \omega g(\mathbf{X}_{0j})$.
 - 4: **while** Until reaching the time limit **do**
 - 5: Select M candidate points $\mathbf{X}_{tj}$ ($j = 1, 2, \dots, M$) from the neighborhood $\mathbf{S}(\mathbf{X}_t)$ uniformly at random.
 - 6: Evaluate $h(\mathbf{X}_{tj}) = f(\mathbf{X}_{tj}) + \omega g(\mathbf{X}_{tj})$.
 - 7: Select the top- K candidate points $\hat{\mathbf{X}}_{tk}$, ($k = 1, 2, \dots, K$) from $\mathbf{X}_{tj}$ according to $h(\mathbf{X}_{tj})$.
 - 8: Evaluate $H(\hat{\mathbf{X}}_{tk}) = F(\hat{\mathbf{X}}_{tk}) + \omega G(\hat{\mathbf{X}}_{tk})$.
 - 9: $\hat{\mathbf{X}}_t \leftarrow \hat{\mathbf{X}}_{tk^*}$, where $k^* \leftarrow \underset{k}{\operatorname{argmin}} H(\hat{\mathbf{X}}_{tk})$.
 - 10: **if** $H(\hat{\mathbf{X}}_t) < H(\mathbf{X}_{\text{best}})$ **then**
 - 11: $\mathbf{X}_{\text{best}} \leftarrow \hat{\mathbf{X}}_t$.
 - 12: Generate a random number $0 \leq r \leq 1$.
 - 13: $P \leftarrow \exp((H(\hat{\mathbf{X}}_t) - H(\mathbf{X}_t))/T)$.
 - 14: **if** $H(\hat{\mathbf{X}}_t) < H(\mathbf{X}_t)$ or $r \leq P$ **then**
 - 15: $\mathbf{X}_{t+1} \leftarrow \hat{\mathbf{X}}_t$.
 - 16: $T \leftarrow \alpha T$.
 - 17: $t \leftarrow t + 1$.
 - 18: **return** $\mathbf{X}_{\text{best}}$
-

This optimization procedure starts when the target shipping orders from customers are confirmed. Let $\mathbf{X}_t$ be the searching point in iteration t . First, we generate M_0 searching points randomly and select the initial searching point $\mathbf{X}_1$ that has the minimum surrogate penalized objectives $h(\mathbf{X}) = f(\mathbf{X}) + \omega g(\mathbf{X})$, where $f(\mathbf{X})$ and $g(\mathbf{X})$ are surrogate models and ω is the penalty parameter. Then the M candidate points are sampled uniformly at random from the *neighborhood* $\mathbf{S}(\mathbf{X}_t)$, which contains all points with a Hamming distance of at most one from $\mathbf{X}_t$. In the *surrogate pruning*, the top- K candidate points are chosen from M candidate points according to the surrogate penalized objectives $h(\mathbf{X})$. This *surrogate pruning* can reduce the computational time by choosing

good candidate solutions quickly without time-consuming simulations.

For the selected K candidate points, the penalized objectives $H(\mathbf{X}) = F(\mathbf{X}) + \omega G(\mathbf{X})$ are evaluated by using their simulation results ($F(\mathbf{X})$ and $G(\mathbf{X})$). Then, the best candidate point $\hat{\mathbf{X}}_t$ is chosen from the K candidate points. If $\hat{\mathbf{X}}_t$ has a lower penalized objective than the best solution $\mathbf{X}_{\text{best}}$, $\mathbf{X}_{\text{best}}$ is replaced with $\hat{\mathbf{X}}_t$. The searching point is updated as same as SA. If $H(\hat{\mathbf{X}}_t)$ is smaller than $H(\mathbf{X}_t)$, let $\mathbf{X}_{t+1} \leftarrow \hat{\mathbf{X}}_t$. Otherwise, $\hat{\mathbf{X}}_t$ is adopted as $\mathbf{X}_{t+1}$ with the following probability P :

$$P = \exp\left((H(\hat{\mathbf{X}}_t) - H(\mathbf{X}_t))/T\right), \quad (5.6)$$

where T is the temperature parameter. T starts with a high T_0 to encourage exploration and then cools down by multiplying the cooling rate $\alpha \in [0, 1]$ to zoom in on a good solution.

These processes continue until the computational time reaches the time limit. Finally, the best searching point is returned as the solution order sequence. In this paper, we set $M_0 = M = 10000$, by which the surrogate runtime in a loop is almost equal to one simulation runtime, and $\omega = 10^6$ so that the deadlocked candidates always become worse than any candidates without deadlock. We also selected $K = 10$. In the preliminary experiments, we found that the optimization performance was excellent when $5 \leq K \leq 30$, and it became unstable when $K = 1$. It implies that selecting multiple candidates can improve the robustness of finding *good* solutions under prediction errors. Additionally, we selected $T_0 = 30$ and $\alpha = 0.95$ that showed the best optimization performance according to the preliminary experiments with the following parameter ranges; $T_0 \in [10, 30, 100]$ and $\alpha \in [0.9, 0.95, 0.99]$.

5.4 Evaluation

We conducted numerical experiments to investigate RQ1 and RQ2 and compared the proposed optimization method with conventional metaheuristic methods.

5.4.1 Conditions

In our experiments, the sequence of the given picking orders is optimized for minimizing the makespan without the deadlock of the PPCPP system (Refer back to Figure 5.2 for the system configuration.) All picking orders used in the experiments are generated randomly. One picking order set consists of 1 to 30 picking orders. Each picking order contains 1 to 5 items and the shipping quantity of each item is 1 to 700. The pre-training dataset is the PPCPP simulation results of 50,000 randomly generated picking order sets whose order sequence s are randomly sorted.

We formulate the deadlock occurrence prediction as a binary classification problem and the makespan prediction as a regression problem. In the following experiments, BRNN shown in

Figure 5.5 is compared with linear regression (LR), Gaussian process regression (GP), gradient boosting trees (GBT), and the Bayesian neural network without recurrent layer (BNN). Each picking order set is converted to the input features shown in Figure 5.6. BRNN uses the sequential feature whose shape is (N_s, N_t, N_f) , where N_s is the number of samples of the picking order sets, N_t is the maximum sequence number, and N_f is the number of the features. Other models, which cannot handle the sequential feature, use the 2D feature consisting of horizontally connected sequential features. This data shape is $(N_s, N_t \times N_f)$. In this experiment, we used $N_f = 34$ features and set $N_t = 150$ to cover all training dataset. In the pre-training of GP, only 200 randomly selected samples were used to shorten the training time. The prediction accuracies of makespans and deadlock occurrence were evaluated by 4-fold cross-validation. The computing environment is a Ryzen 3950X CPU (3.70GHz, 32threads), 128GB RAM, and NVIDIA RTX3080 GPU.

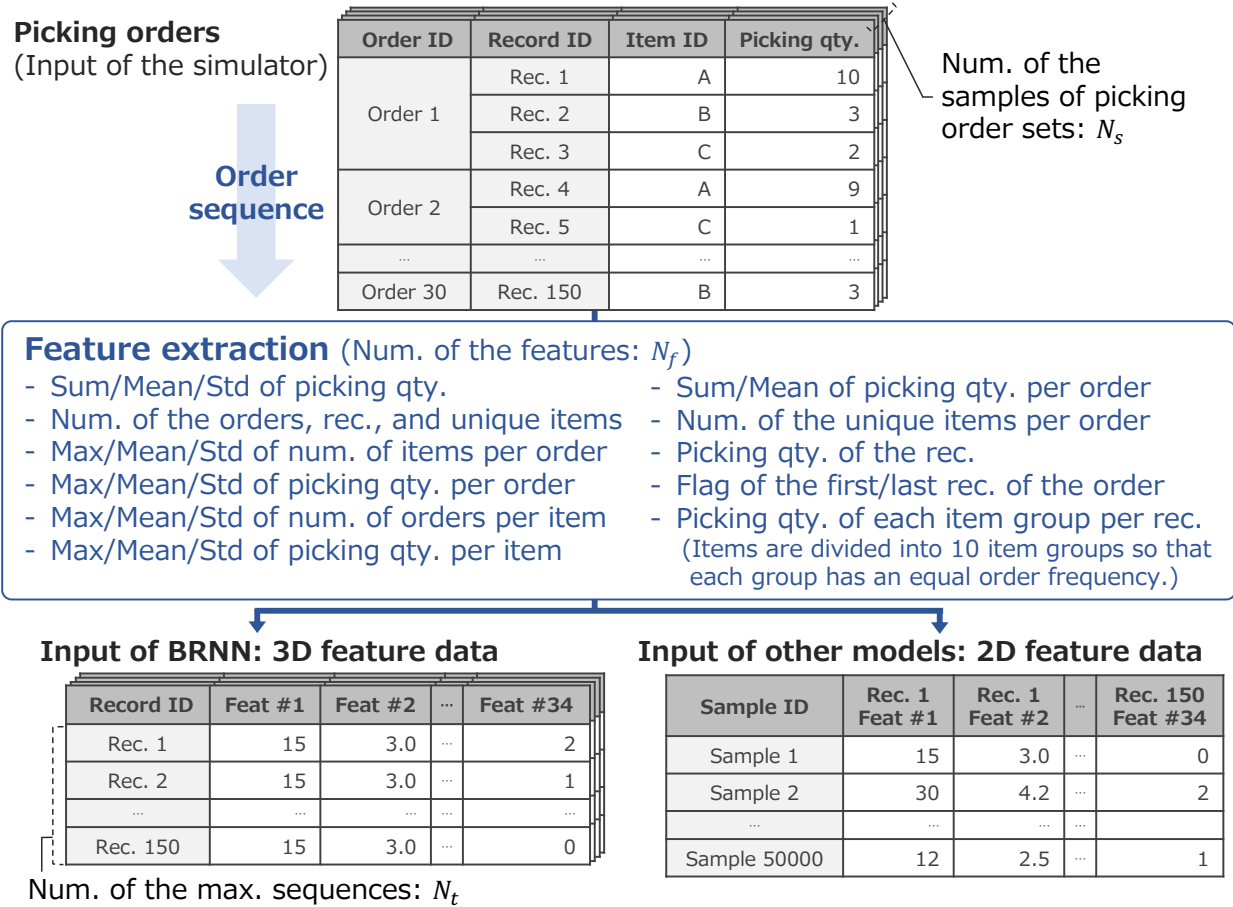


Figure 5.6. Feature formats of surrogate models. Picking orders are converted to 3D feature data for BRNN and 2D feature data for others.

5.4.2 Prediction Accuracy Comparison

In this experiment, we evaluate the prediction accuracies of the pre-trained surrogate models. These models are compared based on accuracy, defined as the number of correctly predicted data points out of all the data points and the area under the ROC curve (ROC-AUC) for the deadlock occurrence prediction. Mean absolute percentage error (MAPE) and mean absolute error (MAE) are used as metrics for the makespan regression.

The evaluation results of each metric are shown in Table 5.2. For the deadlock occurrence prediction, the classification accuracy is highest for GBT (93.7%), BRNN (93.2%), and BNN (93.2%) in that order. However, the differences in classification accuracies between all models are not so significant (92.7–93.7%), which implies that the order sequence has little effect on the deadlock occurrence. For the makespan regression, BRNN shows a lower MAPE (2.68%) than GBT (3.45%), BNN (5.15%), and other methods. Additionally, we confirmed the BRNN inference time (0.89 msec/sample) was over 10000 times faster than the simulation runtime (11.7 sec/sample).

Table 5.2. Prediction result comparison of makespan and deadlock occurrence.

Surrogate model	Deadlock classification		Makespan regression		Inference time [msec./sample]
	Accuracy[%]	ROC-AUC	MAPE[%]	MAE	
BRNN	93.2 ± 0.3	0.987 ± 0.001	2.68 ± 0.21	53.4 ± 2.2	0.89
BNN	93.2 ± 0.6	0.985 ± 0.001	5.15 ± 0.43	98.9 ± 1.4	0.96
GBT	93.7 ± 0.2	0.987 ± 0.000	3.45 ± 0.04	63.7 ± 1.2	0.02
GP	92.8 ± 0.3	0.982 ± 0.001	7.73 ± 0.61	123.8 ± 13.5	0.84
LR	92.7 ± 0.3	0.983 ± 0.001	7.87 ± 1.68	156.9 ± 22.7	0.002
Simulation	(100.0)	(1.0)	(0.0)	(0.0)	11752

We also evaluated how accurately each model can predict the makespan changes according to the order sequences of one order set. Figure 5.7 shows the simulated and predicted makespans of all combinations of the seven randomly generated orders (5040 combinations). The horizontal axis represents each order sequence sorted by its simulated makespan. The vertical axis represents the makespan of each order sequence obtained by simulation and prediction with the pre-trained surrogate models. We can see that the BRNN prediction (blue line) is closer to the simulation result (orange line) than any other surrogate model. These results suggest that BRNN is suitable as a surrogate model to predict performances strongly affected by the sequential feature such as the order sequence, which is the decision variable of OSPs.

Based on these findings, we answer the RQ1 as follows.

RQ1: Which surrogate model is most suitable for the OSPs of PPCPP systems?

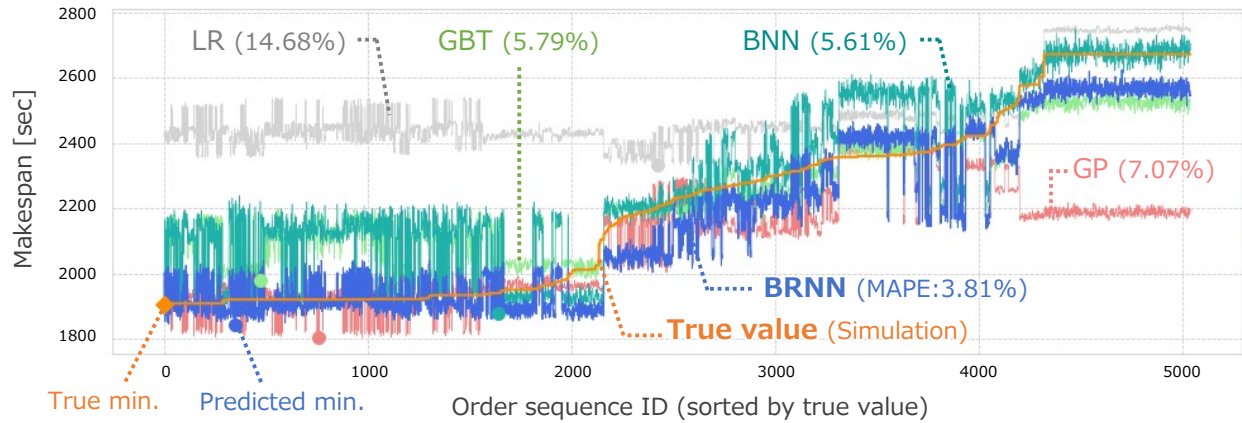


Figure 5.7. Simulated and predicted makespans (y-axis) of all sequence combinations of seven orders (x-axis) sorted by simulated makespans. The orange line indicates the simulated true makespans, and the others are the predicted makespans of each surrogate model annotated with MAPE. If the prediction results are always close to the simulation results, the surrogate model can predict the makespan changes according to the order sequence (decision variables).

A1: In the performance measures of OSPs, makespan is strongly affected by the order sequences. To infer such performance measures precisely, surrogate models that can consider the sequential input, such as BRNN, are suitable.

5.4.3 Optimization Runtime Comparison

Next, we evaluated the total runtimes to find the optimal order sequence by the proposed surrogate pruning optimization method. In this experiment, we used three problem instances consisting of 20 picking orders that are randomly generated from the same distribution with training data. These problems were solved by the proposed optimization methods with each surrogate model (BRNN, BNN, GBT, GP, LR) and the original SA. All the experiments were executed ten times each, and their time limits were 900 sec. We then compared the averages of the objective values of each final solution (Final obj.) and the runtimes to find the solution whose objective value is equal to or lower than the original SA's final solution (LE runtime). Table 5.3 shows the optimization results. As we can see, BRNN-SA can find the best solution to two problems and show the shortest LE time of all methods (8.9 times faster than the original SA on average). BRNN-SA shows the best optimization performances since BRNN can predict the makespan changes according to the order sequence more accurately than other surrogate models as shown in Figure 5.7.

Figure 5.8 shows the optimization trajectories of each optimization method in Problem 1. The horizontal axis represents the optimization runtime, and the vertical axis represents the makespan of the best solution $\mathbf{X}_{\text{best}}$ at that moment. We can see here that BRNN-SA quickly finds a solution whose objective is lower than SA's final solution. This demonstrates that surrogate pruning by

pre-trained BRNN can choose good candidate points more precisely than other surrogate models even at the beginning of the optimization.

On the basis of these findings, we answer RQ2 as follows;

RQ2: How quickly can the proposed optimization method (BRNN-SA) solve the OSPs of PPCPP systems compared to conventional metaheuristics?

A2: BRNN-SA can solve the 20 order OSPs by 8.9 times faster than the original SA. This is presumably because the surrogate pruning with the pre-trained BRNNs can find good order sequences successfully by inferring their performance measures without simulation.

Table 5.3. Objectives and runtimes of the optimization methods assisted by each surrogate model.

Problem	BRNN-SA	BNN-SA	GBT-SA	GP-SA	LR-SA	Orig. SA
#1: Final obj.	3169 $\pm$ 236	4090 $\pm$ 239	3305 $\pm$ 228	3537 $\pm$ 247	3442 $\pm$ 265	3706 $\pm$ 232
#1: LE runtime [s]	106.4	–	160.7	268.0	402.9	883.1
#1: Speed-up	$\times 8.3$	$\times 0.0$	$\times 5.5$	$\times 3.3$	$\times 2.2$	$\times 1.0$
#2: Final obj.	2174 $\pm$ 127	2171 $\pm$ 119	2204 $\pm$ 119	2194 $\pm$ 80	2311 $\pm$ 106	2699 $\pm$ 468
#2: LE runtime [s]	67.1	84.4	85.6	92.9	280.9	763.9
#2: Speed-up	$\times 11.4$	$\times 9.1$	$\times 8.9$	$\times 8.2$	$\times 2.7$	$\times 1.0$
#3: Final obj.	1805 $\pm$ 74	1853 $\pm$ 88	1823 $\pm$ 89	1929 $\pm$ 72	1818 $\pm$ 72	2064 $\pm$ 308
#3: LE runtime [s]	101.4	397.5	188.9	896.9	197.2	798.2
#3: Speed-up	$\times 7.9$	$\times 2.0$	$\times 4.2$	$\times 0.9$	$\times 4.1$	$\times 1.0$
Avg. Speed-up	$\times 8.9$	$\times 1.8$	$\times 5.6$	$\times 1.9$	$\times 2.8$	$\times 1.0$

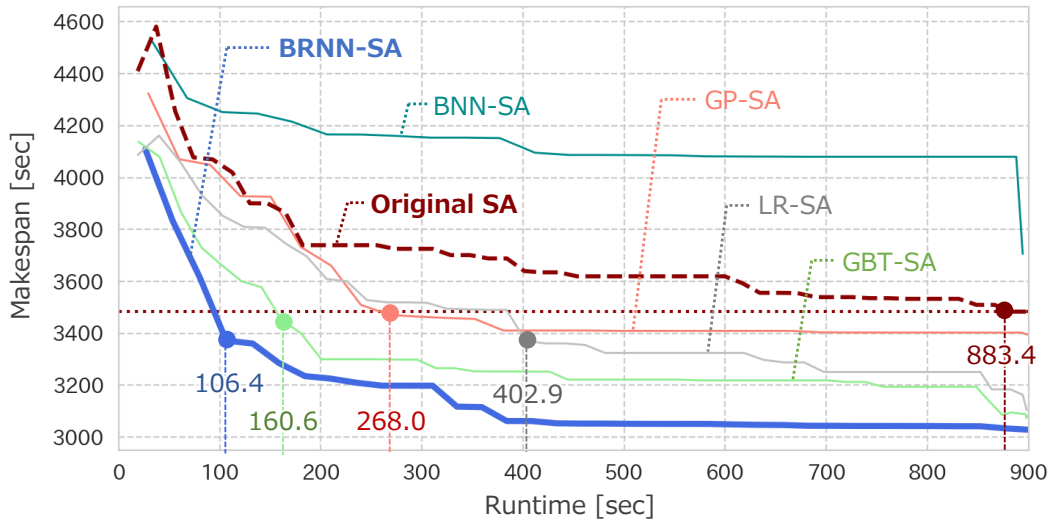


Figure 5.8. Makespan trajectories of Problem 1. The dashed line shows the original SA’s trajectory, and the other lines are those of surrogate models. The horizontal dotted line shows the makespan of the original SA’s final solution, and the vertical dotted lines indicate the LE runtimes.

5.5 Conclusion

This paper proposed the surrogate pruning optimization framework and its implementation (BRNN-SA) to quickly solve the order sequencing problem (OSP) of PPCPP systems, the automated order picking system in logistics warehouses. The proposed method utilizes the Bayesian recurrent neural network (BRNN) as surrogate models to infer sequence-dependent makespans and deadlock occurrences. Then we proposed BRNN-SA, which combines simulated annealing (SA) with pre-trained BRNNs to choose *good* candidate solutions without simulation (*surrogate pruning*). Our experimental results demonstrated the proposed method can streamline solving OSPs thanks to surrogate pruning with the pre-trained BRNNs. For future work, we plan to implement the incremental learning step of surrogate models in this framework for solving larger-scale problems and apply the proposed method to other sequence optimization problems such as job-shop scheduling problems.

Acknowledgment

AI Bridging Cloud Infrastructure (ABCI) of National Institute of Advanced Industrial Science and Technology (AIST) was used for parameter optimization.

Hyper-heuristics Framework for Online Order Batching and Sequencing Problem in Automated Logistics Warehouses with Order Consolidation Process

6.1 Introduction

Driven by the rapid expansion of the e-commerce market, distribution warehouses face increasing product diversity and shipment volumes while being required to meet high service level requirements such as same-day delivery. A declining labor force and stricter labor regulations make it difficult to secure warehouse workers, increasing the need for automation to improve efficiency and reduce staffing. In distribution warehouses, the order-picking process—collecting items from storage locations in response to customer orders—accounts for over 50% of total warehouse costs [137], [158]. Consequently, a variety of automation solutions for picking have been proposed, including automated storage and retrieval systems (AS/RS), automated guided vehicles (AGVs), and piece/case sorters. In recent years, advanced automation solutions have been deployed in practice for many processes, such as shelf-carrying AGV picking systems and palletizing robots [88], [159]. However, improving the productivity of the entire fulfillment line requires coordination among equipment and the removal of bottlenecks, which remains a significant challenge [160].

To enhance productivity across the warehouse through coordination among processes, this study investigates the optimization of order batching and sequencing for dynamically arriving shipping orders in distribution warehouses equipped with multiple automated systems. This problem is referred to as the Online Order Batching and Sequencing Problem (OOBSP), a variant of the Order Batching Problem (OBP)[161]. OBP is a combinatorial optimization problem that partitions orders into batches to minimize performance measures such as makespan and travel distance. Large-scale OBPs are generally solved using heuristics, but their performance depends heavily on instance characteristics, like the properties of shipping orders and fulfillment line configurations [153], [162].

In recent years, data-driven approaches such as deep reinforcement learning (DRL) and hyper-heuristics have been proposed for such combinatorial optimization problems [163], [164]. Cals et al. [165] proposed a DRL-based approach to the online OBP for a shipping line that includes two picking processes: AS/RS and cart picking. Their method classifies orders into 15 categories based on the number of items, processing paths, and slack time to the due date, and defines 31 strategies—consisting of single- or multi-order batches for each category or postponing batching—which are dynamically selected by a DRL policy to minimize shipment tardiness. Cheng et al. [166] proposed a DRL-driven hyper-heuristic method for the offline OBP in AGV picking systems. Their approach defines eight batching strategies that combine heuristics based on the number of repeated uses of storage racks and due dates, and a DRL policy selects among them to minimize rack transports and shipment delays. Furthermore, Wang et al. [167] addressed the combined problem of OBP and picker routing by proposing a hyper-heuristic approach based on genetic programming (GPHH) to automatically synthesize new order-selection indices by combining multiple indicators. In GPHH, formulas for the optimal order-selection index are generated using 12 order features, such as total order weight and the number of additional racks, and six operators (+, −, ×, ÷, max, min). However, these methods evaluate orders solely by static attributes—such as arrival order, due date, item count, and rack count—without considering the operational state of the shipping line. To improve productivity across the entire line, the order evaluation scheme should adapt to task progress and equipment utilization. Consequently, the effectiveness of conventional approaches may be limited.

Therefore, we explore an OOBSP solution that reflects the state of the shipping line in batching decisions to enhance system-wide productivity. Specifically, we consider a distribution warehouse equipped with modern automated systems, including multiple picking processes and an order consolidation process as post-picking work. In order consolidation, items picked from different processes are temporarily stored until all items for a shipping box are gathered in the AS/RS. This process tends to increase makespan and tardiness. Thus, it is important to control order batching and sequencing to reduce the time required for the completion of consolidation. However, to our knowledge, no method has explicitly targeted this aspect.

In this study, we first propose heuristics aimed at improving the processing efficiency of the consolidation process. Furthermore, to reduce makespan and tardiness throughout the warehouse, we propose an OOBSP solution approach based on a hyper-heuristic framework that incorporates both order characteristics and the current state of the shipping line. The remainder of this paper is organized as follows. First, we define and formulate the target OOBSP. Next, we present an overview of the proposed method, describe the set of low-level heuristics, and explain the hyper-heuristic scheme that integrates them. Finally, the effectiveness of the proposed method is evaluated through numerical experiments.

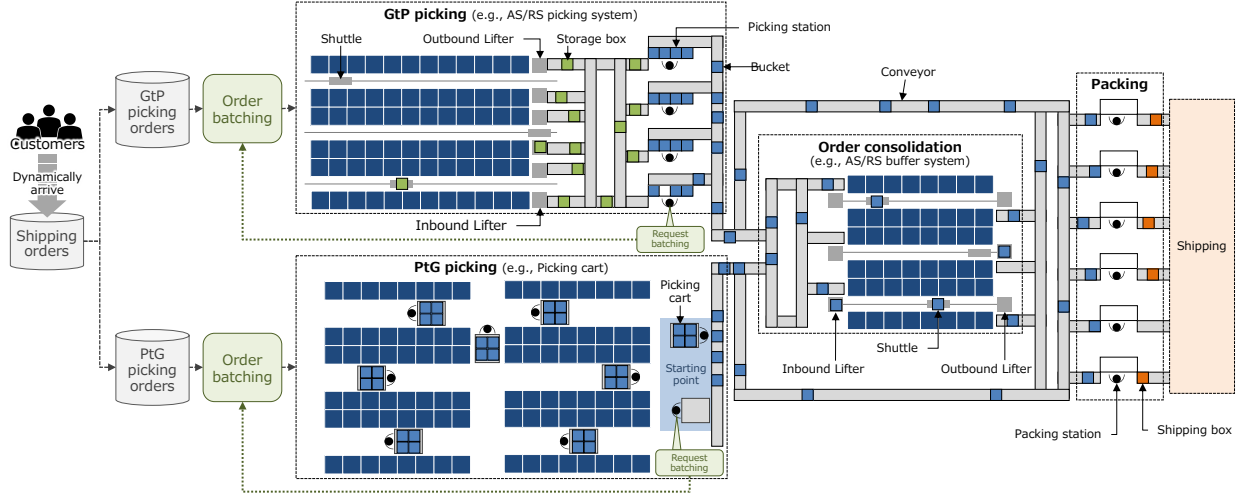


Figure 6.1. Target fulfillment line consisting of GtP picking, PtG picking, order consolidation, and packing processes.

6.2 Online Order Batching and Sequencing Problem

6.2.1 Problem Definition and Representative Solution Methods

The Order Batching Problem (OBP) is a combinatorial optimization problem that determines combinations (batches) of shipping orders that minimize picking costs such as total travel distance and makespan [162]. It is known that OBP is NP-hard when two or more orders can be assigned to a single batch [168]. Furthermore, OBP with batch sequencing, in which the execution order of batches must also be determined, is called the Order Batching and Sequencing Problem (OBSP). In conventional OBP and OBSP, all orders to be batched are assumed to be known when solving the problem. In contrast, the Online Order Batching and Sequencing Problem (OOBSP) refers to the setting where shipping orders are received dynamically, and batching and sequencing decisions must be made in real time. Since the total number of possible batches increases exponentially with the number of orders, realistic OBPs with more than 100 orders become extremely large combinatorial optimization problems [169]. Thus, exact optimization approaches are generally infeasible, and heuristics or metaheuristics are commonly employed [153]. In particular, for online OBPs such as OOBSP, fast heuristics are primarily used, because batches must be sequentially constructed from the set of currently available orders whenever batching is required.

Representative heuristic approaches for OBP include priority-based heuristics, such as First-Come, First-Served (FCFS) and Earliest Due Date (EDD), and are widely used. When processing times are known, the Longest/Shortest Processing Time and Least Slack Time heuristics are also applicable. Another well-known approach is the Savings algorithm, which is based on the

Clarke–Wright method for the Vehicle Routing Problem (VRP) [162]. In online OBP, time-window batching methods are often employed. In fixed time window batching, a predefined time interval is set, and all orders arriving within that interval are batched together. In variable time window batching, a batch is formed once the maximum allowable number of orders has been reached [170].

In addition, *seed heuristics* are widely applied to both online and offline OBPs. In this approach, an initial *seed* order is selected, and then additional orders are incrementally assigned until reaching the maximum batch capacity. Indicators for selecting the seed order include the number of items, number of aisles, travel distance, and processing time. Indicators for selecting additional orders include the number of added items or aisles, and measures of similarity to already assigned orders in the batch (e.g., item overlap or space overlap) [171]. Although these heuristic approaches have been extensively studied, their performance strongly depends on the characteristics of shipping orders, as well as the configuration and layout of the warehouse.

6.2.2 Mathematical Optimization Model of OBSP

In this study, we consider the OOBSP with the objective of minimizing both the total completion time (makespan) for processing all given shipping orders and the tardiness of each order relative to its shipping deadline. Following Cals et al. [165] and Henn [172], we analyze the mathematical properties of the OOBSP via the optimization model of its offline counterpart, OBSP. In the OBSP formulation, the set of all order combinations that satisfy batch-capacity constraints is enumerated in advance, explicitly representing all feasible batches. The processing time of each batch is assumed to be precomputed. Each order is assigned exactly once to one batch from this set, and the combination of batches that minimizes the objective function is selected. In warehouse operations, a single solution must be determined automatically online, even though some tardiness may be unavoidable. Therefore, we formulate the OBSP as a single-objective optimization problem whose objective is a weighted sum of makespan and tardiness, rather than a multi-objective problem or a formulation that treats tardiness as a hard constraint.

The following constants are used in the formulation:

- δ_{ij} : An indicator parameter, defined for all feasible batches that satisfy the batch-capacity constraint, equal to 1 if order i is included in batch j , and 0 otherwise.
- c_i : Capacity requirement of order i (e.g., size or volume).
- d_i : Due date of order i .
- t_j : Processing time of batch j .
- C : Maximum capacity per batch.
- M : A sufficiently large positive constant.

We define the following index sets:

- $\mathcal{I}$: Index set of orders, $\mathcal{I} = \{1, \dots, I_{\max}\}$, where $I_{\max}$ denotes the number of orders to be assigned.
- $\mathcal{J}$: Index set of all feasible batches that satisfy the batch-capacity constraint ($\mathcal{J} = \{j \mid \sum_{i \in \mathcal{I}} c_i \delta_{ij} \leq C\}$).
- $\mathcal{P}$: Index set of picking workers, $\mathcal{P} = \{1, \dots, P_{\max}\}$, where $P_{\max}$ denotes the number of workers available for assignment.
- $\mathcal{Q}$: Index set of sequence positions for batches assigned to each worker, $\mathcal{Q} = \{1, \dots, Q_{\max}\}$, where $Q_{\max}$ denotes the maximum number of batches a worker can process.

We also define the following decision variables:

- $x_{j pq}$: Binary decision variable equal to 1 if batch j is assigned to worker p in position q , and 0 otherwise.
- e_{pq} : Completion time of the q -th batch processed by worker p .
- r_i : Tardiness of order i .

Let $\mathcal{X} = \{x_{j pq}\}$ denote the set of decision variables representing batch assignments. Then, the OBSP optimization model can be formulated as follows [172]:

$$\min_{\mathcal{X}} f_{\text{makespan}} + \alpha f_{\text{tardiness}} \quad (6.1)$$

$$\text{s.t. } f_{\text{makespan}} \geq e_{pq}, \quad \forall p \in \mathcal{P}, q \in \mathcal{Q}, \quad (6.2)$$

$$f_{\text{tardiness}} = \sum_{i \in \mathcal{I}} r_i, \quad (6.3)$$

$$\sum_{j \in \mathcal{J}} x_{j pq} \leq 1, \quad \forall p \in \mathcal{P}, q \in \mathcal{Q}, \quad (6.4)$$

$$\sum_{j \in \mathcal{J}} \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{Q}} \delta_{ij} x_{j pq} = 1, \quad \forall i \in \mathcal{I}, \quad (6.5)$$

$$e_{p, q-1} + \sum_{j \in \mathcal{J}} t_j x_{j pq} \leq e_{pq}, \quad \forall p \in \mathcal{P}, q \in \mathcal{Q} \setminus 1, \quad (6.6)$$

$$\sum_{j \in \mathcal{J}} t_j x_{j p1} \leq e_{p1}, \quad \forall p \in \mathcal{P}, \quad (6.7)$$

$$e_{pq} - r_i \leq d_i + M(1 - \delta_{ij} x_{j pq}), \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, p \in \mathcal{P}, q \in \mathcal{Q}, \quad (6.8)$$

$$e_{pq} \geq 0, \quad \forall p \in \mathcal{P}, q \in \mathcal{Q}, \quad (6.9)$$

$$r_i \geq 0, \quad \forall i \in \mathcal{I}, \quad (6.10)$$

$$x_{j pq} \in 0, 1, \quad \forall j \in \mathcal{J}, p \in \mathcal{P}, q \in \mathcal{Q}. \quad (6.11)$$

Equation 6.1 defines the objective as minimizing a weighted sum of makespan and total tardiness. Equation 6.2 defines the makespan as the maximum completion time e_{pq} over all batches

and workers. Equation 6.3 defines total tardiness as the sum of r_i over all orders. Equation 6.4 enforces that, for each worker p and sequence position q , at most one batch j is assigned. Equation 6.5 enforces that each order i is assigned to exactly one batch. Equation 6.6 ensures that, for each worker p , the completion time of the q -th batch equals the completion time of the $(q - 1)$ -th batch plus its processing time. Equation 6.7 specifies the completion time of the first batch for each worker. Equation 6.8 bounds the tardiness r_i of each order i : if batch j is assigned as the q -th batch of worker p , then $e_{pq} - d_i$ is an upper bound for r_i . Equations 6.9 and 6.10 enforce non-negativity of completion times and tardiness. Equation 6.11 declares x_{jpq} as a binary variable. Based on this OBSP formulation, we now proceed to the formulation of the OOBSP.

6.2.3 Formulation of Simulation-Based OOBSP

Simulation Model and Flowchart

In this study, we address the above OBSP as an online optimization problem using discrete-event simulation, targeting a shipping line that comprises two picking modes: Goods-to-Person (GtP) via AS/RS and Person-to-Goods (PtG) via picking carts. The configuration of the shipping line is shown in Figure 6.1. The line consists of four processes — GtP picking ($k = 1$), PtG picking ($k = 2$), order consolidation ($k = 3$), and packing ($k = 4$) — connected by conveyors. We assume that each item has sufficient stock and that storage locations do not overlap between GtP and PtG picking.

The simulation flowchart is shown in Figure 6.2. Given a set of shipping orders $\mathcal{O} = \{O_i\}$ arriving at times $\mathcal{A} = \{a_i\}$, the batching decisions $\mathcal{X} = \{x_{jpq}\}$ and the completion times $\mathcal{E} = \{e_i\}$ are generated sequentially when a dispatching policy $\mathcal{H}$ is applied. Here, e_i corresponds to e_{pq} in the OBSP formulation as

$$e_i = \sum_{j \in \mathcal{J}} \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{Q}} \delta_{ij} x_{jpq} e_{pq}, \quad \forall i \in \mathcal{I}. \quad (6.12)$$

A shipping order O_i is defined as $O_i = \{P_{in} \mid n = 1, 2, \dots, N_i\}$, where P_{in} denotes the n -th item included in the order and N_i is the number of items in O_i . Upon the arrival of O_i from a customer at time a_i (specified per shipping box), the order is automatically split into GtP and PtG picking orders according to the storage locations of items P_{in} and added to the pool of open orders for each process. Below, the picking order produced for process k by splitting O_i is denoted by O_{ik} . Each worker in the GtP/PtG processes requests the next task upon completing the current picking operation. In response, online batching based on method $\mathcal{H}$ is performed over the current pool of open orders in that process, and a new picking task is assigned to the requesting worker.

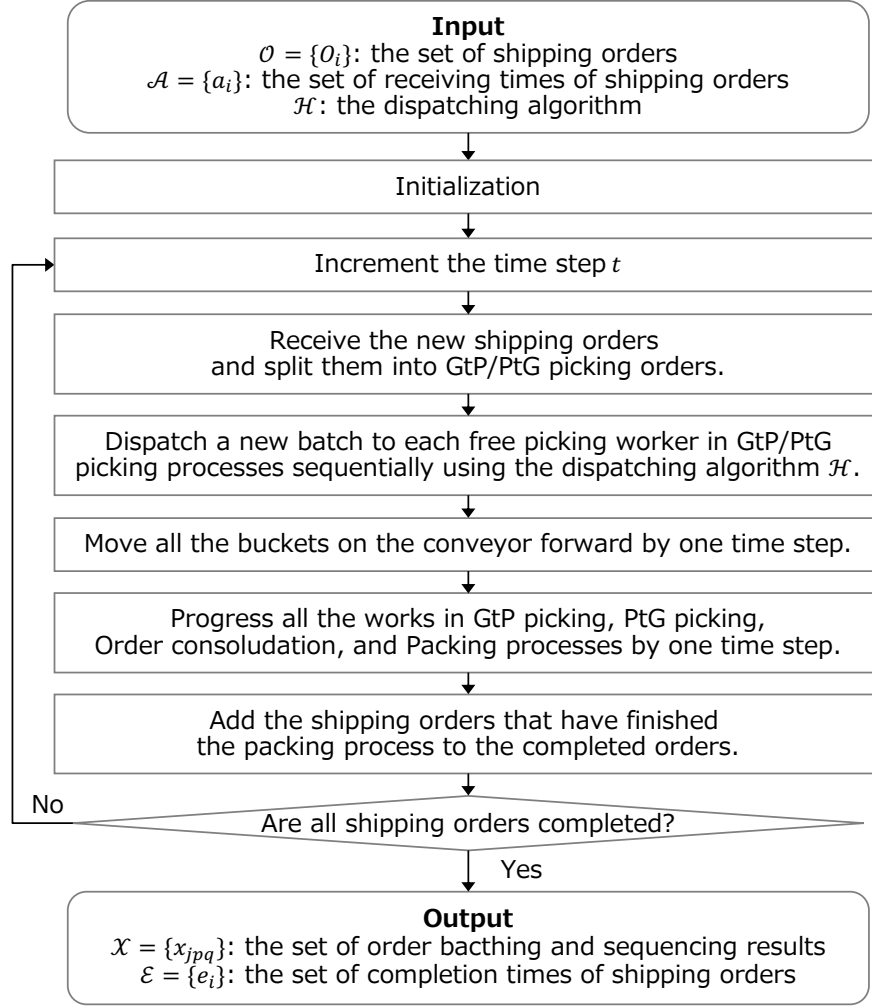


Figure 6.2. Simulation flowchart for OOBSP.

Picking orders completed in the GtP/PtG processes are placed into buckets and transferred to the next process. If a shipping order contains items from both processes, the corresponding buckets are sent to the consolidation process; otherwise, they are sent directly to the packing process. In the consolidation process, buckets arriving from the GtP and PtG processes are temporarily stored in an AS/RS until all items required for a shipping box are collected. Once all items for the corresponding shipping order are stored in the AS/RS, the buckets are released in sequence to the packing process. At the packing process, the buckets undergo inspection and packing per shipping order, after which they are combined into a single shipping box. The completion time e_i of shipping order i is recorded as the time its shipping box is completed at the packing process. These operations are repeated until all shipping orders complete the packing process. Note that no randomness is included in this simulation.

Detailed Simulation Process

We detail the operations of the GtP and PtG picking, consolidation, and packing processes in the simulation below.

1. GtP Picking Process

- A batch of shipping orders is assigned to a worker at a picking station (PST), and items in the batch are retrieved from storage boxes in the AS/RS. Boxes with lower remaining inventory are prioritized for allocation.
- Upon allocation, transport instructions are generated for the shuttle (horizontal movement) and the lifter (vertical movement) and executed in first-come, first-served (FCFS) order. The shuttle carries the storage box from its slot to the lifter, the lifter brings it to ground level, and it is then conveyed to the PST.
- At the PST, the worker picks the required quantity from the storage box and sorts the items into buckets according to each picking order. Buckets whose picking is complete are sent to the next process via conveyor.
- Once picking at the PST is completed, the storage box is returned to an empty slot in the AS/RS in the reverse order of allocation.
- Upon completion of a picking order, the PST worker requests the next batch of picking orders.

2. PtG Picking Process

- A batch is assigned to a worker with a picking cart, who traverses the warehouse, collects items, and sorts them into buckets on the cart.
- Aisles in the picking area permit passing; however, only one worker may pick from the same storage location at a time (FCFS).
- The tour starts at the picking start point and is constructed sequentially by a nearest-neighbor rule, directing the worker to the closest required item location from the current position. After collecting all required items, the worker returns to the start point.
- After returning to the start point, the worker places the buckets onto the conveyor and requests the next batch of picking orders.

3. Consolidation Process

- When a bucket arrives at the consolidation process, storage instructions are generated for the shuttle and inbound lifter of an AS/RS tier with empty slots, and executed in FCFS order.
- When all buckets corresponding to a shipping order have been stored in the AS/RS, retrieval instructions are generated for the shuttle and outbound lifter of the relevant

tiers and executed in FCFS order.

- Upon retrieval, all buckets corresponding to the same shipping order are released to the conveyor in sequence and sent to the packing process.

4. Packing Process

- Buckets arriving from upstream processes are grouped per shipping order; after checking items and quantities, the contents are packed into a single shipping box per order, thereby finalizing the shipping order.

Formulation of Simulation-based OOBSP

The above simulation can be expressed as $\mathcal{X}, \mathcal{E} = \text{Simulation}(\mathcal{H}, \mathcal{O}, \mathcal{A})$. Based on this definition and Equations 6.1–6.11, OOBSP can be formulated as follows:

$$\min_{\mathcal{H}} f(\mathcal{H}, \mathcal{O}, \mathcal{A}, \mathcal{D}) = f_{\text{makespan}} + \alpha f_{\text{tardiness}}, \quad (6.13)$$

$$\text{s.t. } f_{\text{makespan}} = \max_{i \in \mathcal{I}} e_i, \quad (6.14)$$

$$f_{\text{tardiness}} = \sum_{i \in \mathcal{I}} \max \{e_i - d_i, 0\}, \quad (6.15)$$

$$\mathcal{X}, \mathcal{E} = \text{Simulation}(\mathcal{H}, \mathcal{O}, \mathcal{A}), \quad (6.16)$$

$$\mathcal{X} = \{x_{jpq} \mid x_{jpq} \in \{0, 1\}, j \in \mathcal{J}, p \in \mathcal{P}, q \in \mathcal{Q}\}, \quad (6.17)$$

$$\mathcal{E} = \{e_i \mid i \in \mathcal{I}\}, \quad (6.18)$$

$$\mathcal{D} = \{d_i \mid i \in \mathcal{I}\}. \quad (6.19)$$

Equations 6.13, 6.14, and 6.15 correspond to Equations 6.1, 6.2, and 6.3, respectively. They define the objective as minimizing a weighted sum of makespan and total tardiness across orders. Equation 6.16 specifies the simulation dynamics and the mapping from inputs to outputs. Equations 6.17, 6.18, and 6.19 define, respectively, the batching decisions $\mathcal{X}$, the order completion times $\mathcal{E}$, and the order due dates $\mathcal{D}$ within the simulation.

In this formulation, the decision variables $\mathcal{X}$ for the OOBSP are induced by the simulation given a dispatching policy $\mathcal{H}$ as input. Accordingly, we focus on determining the dispatching policies that minimize the objective in Equation 6.13.

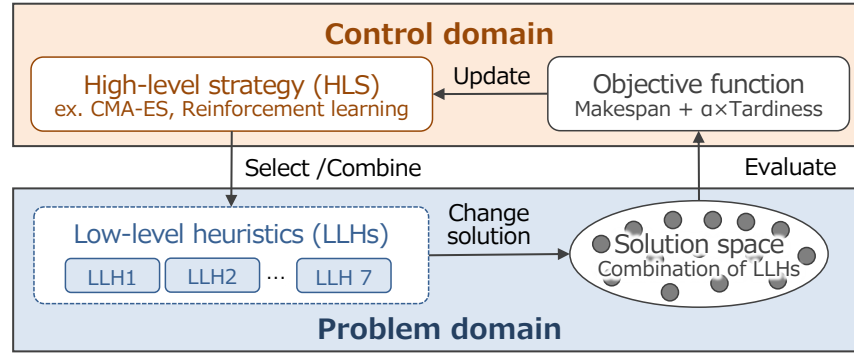


Figure 6.3. Hyper-heuristics framework consisting of the set of problem-specific low-level heuristics and the high-level strategy to select/combine them to minimize the objective function.

6.3 Methodology

6.3.1 Hyper-heuristics

In this section, we present the proposed dispatching policy for solving the OOBSP. Heuristics are preferable for solving large-scale OOBSP instances; however, their performance is problem-dependent. Therefore, we address this issue by adaptively combining existing order batching problem (OBP) heuristics proposed in the literature. This approach—selecting and combining multiple heuristics to generate new ones—is known as *hyper-heuristics* [164]. Hyper-heuristics provide a flexible approach and have been applied in scheduling, routing, and combinatorial optimization [166], [167]. As shown in Figure 6.3, a hyper-heuristic framework consists of problem-specific low-level heuristics (LLHs) and a high-level strategy (HLS) that selects and combines them. The HLS often employs metaheuristics (e.g., genetic programming) or learning-based methods (e.g., deep reinforcement learning, DRL). Guided by the HLS, LLHs are selected and combined to synthesize candidate heuristics, which are then evaluated to identify the best-performing combination of LLHs.

In this study, we focus on seed heuristics widely used for the OOBSP and propose generating both seed-order and additional-order selection indicators via a hyper-heuristic, conditioned on the problem context. Specifically, in addition to five commonly used selection indicators, we propose two new indicators that account for the characteristics of the consolidation process. Furthermore, we propose a hyper-heuristic approach that combines these indicators to generate seed- and additional-order selection indicators tailored to the characteristics of shipping orders.

6.3.2 Low-level Heuristics: LLHs

In this study, we propose two new selection indicators (z_{ik6}, z_{ik7}) that take into account the characteristics of the consolidation process, in addition to the five selection indicators $(z_{ik1}, \dots, z_{ik5})$ used in existing heuristics. Each indicator yields a score in $[0,1]$; the order with the highest score is selected. Below, the number of elements in a set S is denoted by $|S|$. At time t , the index set of received but unassigned orders (open orders) in process k by $\mathcal{I}_{tk}^{\text{open}}$, the index set of assigned but unfinished orders (executing orders) is $\mathcal{I}_{tk}^{\text{exec}}$, and the index set of orders assigned to batch j is $\mathcal{I}_{jk}^{\text{batch}}$. The set of items contained in batch j is defined as $B_{jk} = \bigcup_{i \in \mathcal{I}_{jk}^{\text{batch}}} O_{ik}$. The set of dispatch-target processes is denoted by $\mathcal{K}^* = \{1, 2\}$.

Item Count Score

This indicator is based on the seed-order selection using item counts [162]. It assigns higher scores to orders with fewer items; for $|O_{ik}| \geq 1$, it is defined as in Equation 6.20.

$$z_{ik1} = \frac{1}{|O_{ik}|}, \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.20)$$

Maximum Item Similarity Score

This indicator follows seed-order selection by item similarity [162]. For order i , it is the maximum item-overlap ratio with any other order in $\mathcal{I}_{tk}^{\text{open}}$, as in Equation 6.21.

$$z_{ik2} = \max_{i' \in \mathcal{I}_{tk}^{\text{open}} \setminus i} \left(\frac{|O_{ik} \cap O_{i'k}|}{|O_{ik} \cup O_{i'k}|} \right), \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.21)$$

Additional Item Count Score

This indicator follows additional-order selection by the number of newly added items to a batch [162]. It assigns higher scores when fewer new items are introduced upon adding order i to batch j , as in Equation 6.22.

$$z_{ik3} = \frac{1}{1 + |O_{ik}| - |B_{jk} \cup O_{ik}|}, \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.22)$$

Item Similarity Score

This indicator follows additional-order selection by item similarity [162]. It is the item-overlap ratio between order i and batch j , as in Equation 6.23.

$$z_{ik4} = \frac{|B_{jk} \cap O_{ik}|}{|B_{jk} \cup O_{ik}|}, \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.23)$$

Due-date Priority Score

This indicator is based on the EDD heuristic [162]. It assigns higher scores to orders with shorter slack from the current time t to the due date d_i . Let τ denote the maximum slack considered; the score is given in Equation 6.24.

$$z_{ik5} = 1 - \frac{\min(t - d_i, \tau)}{\tau}, \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.24)$$

Consolidation Priority Score

This is a newly proposed indicator. Orders requiring consolidation cannot proceed until all related orders arrive at the consolidation process, often leading to long completion times. Therefore, when a consolidation-required order is being processed, we prioritize assigning the other orders to be consolidated with it. Let the consolidation indicator $\delta(i, \mathcal{I})$ return 1 if order i requires consolidation with any order in $\mathcal{I}$, and 0 otherwise. The consolidation-priority score is defined in Equation 6.25.

$$z_{ik6} = \delta(i, \mathcal{I}_{tk}^{\text{exec}}), \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.25)$$

Task-Simplicity Score

This is another newly proposed indicator. The packing process — the final stage of shipping — tends to be underutilized early but often becomes a bottleneck later, delaying shipments. We therefore focus on “simple” orders — those with fewer items and no consolidation requirement — that can quickly proceed to packing. By prioritizing such orders when packing utilization is low, we aim to level the workload and reduce delays. Let u_t denote packing utilization at time t . The task-simplicity score is defined in Equation 6.26.

$$z_{ik7} = \frac{1 - u_t^2}{|O_{ik}|^{1 + \delta(i, \mathcal{I}_{tk}^{\text{open}} \cup \mathcal{I}_{tk}^{\text{exec}})}}, \forall i \in \mathcal{I}_{tk}^{\text{open}}, k \in \mathcal{K}^*. \quad (6.26)$$

In this study, we regard the packing process as underutilized when u_t is roughly 0–70%. Based on preliminary experiments, we use the numerator $1 - u_t^2$ so that the score increases under such

conditions.

6.3.3 High-level Strategy: HLS

Seed Heuristics with Weighted Selection Scores

We describe the HLS, which combines the seven LLHs in Equations 6.20–6.26 for a given problem setting. For each picking process $k \in \mathcal{K}^*$, we introduce a weight vector $\mathbf{w}_k = [w_{k1}, \dots, w_{k7}]$ corresponding to the LLHs. Let $\mathbf{W} = \{\mathbf{w}_k\}$ denote the set of weight vectors. Using these weights, we compute weighted sums of the LLH scores and obtain the seed- and additional-order selection scores as in Equations 6.27 and 6.28, respectively.

$$z_{ik}^{\text{seed}} = w_{k1}z_{ik1} + w_{k2}z_{ik2} + w_{k5}z_{ik5} + w_{k6}z_{ik6} + w_{k7}z_{ik7}, \quad (6.27)$$

$$z_{ik}^{\text{add}} = w_{k3}z_{ik3} + w_{k4}z_{ik4} + w_{k5}z_{ik5} + w_{k6}z_{ik6} + w_{k7}z_{ik7}. \quad (6.28)$$

Using these scores, a single seed order and additional orders are selected according to the seed-heuristic flow shown in Figure 6.4, yielding batches that satisfy the per-batch capacity limit C . Hereafter, we denote the proposed method parameterized by the weight coefficients $\mathbf{W}$ as $\mathcal{H}_{\mathbf{W}}$.

In this framework, the appropriate values of $\mathbf{W}$ depend on the characteristics of the target order stream and the configuration of the shipping line. Accordingly, we determine $\mathbf{W}$ to minimize the objective in Equation 6.13, using a training dataset of shipping orders $\mathcal{S} = \{\mathcal{S}_h\} = \{(O_h, \mathcal{A}_h, \mathcal{D}_h)\} (h = 1, 2, \dots, H)$. We consider two variants: a static hyper-heuristic (SHH), in which $\mathbf{W}$ is fixed, and an adaptive hyper-heuristic (AHH), in which $\mathbf{W}$ is determined online during simulation.

Static Hyper-heuristics (SHH)

In this approach, the fixed weight vectors $\mathbf{W}^*$ used throughout the simulation are obtained by solving the optimization problem in Equation 6.29.

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}} \frac{1}{H} \sum_{h=1}^H \{f(\mathcal{H}_{\mathbf{W}}, O_h, \mathcal{A}_h, \mathcal{D}_h)\}. \quad (6.29)$$

Equation 6.29 defines the minimization of the average OOBSP objective (Equation 6.13) over the H training instances. As the optimizer for Equation 6.29, we employ CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [57], a black-box, derivative-free optimization method.

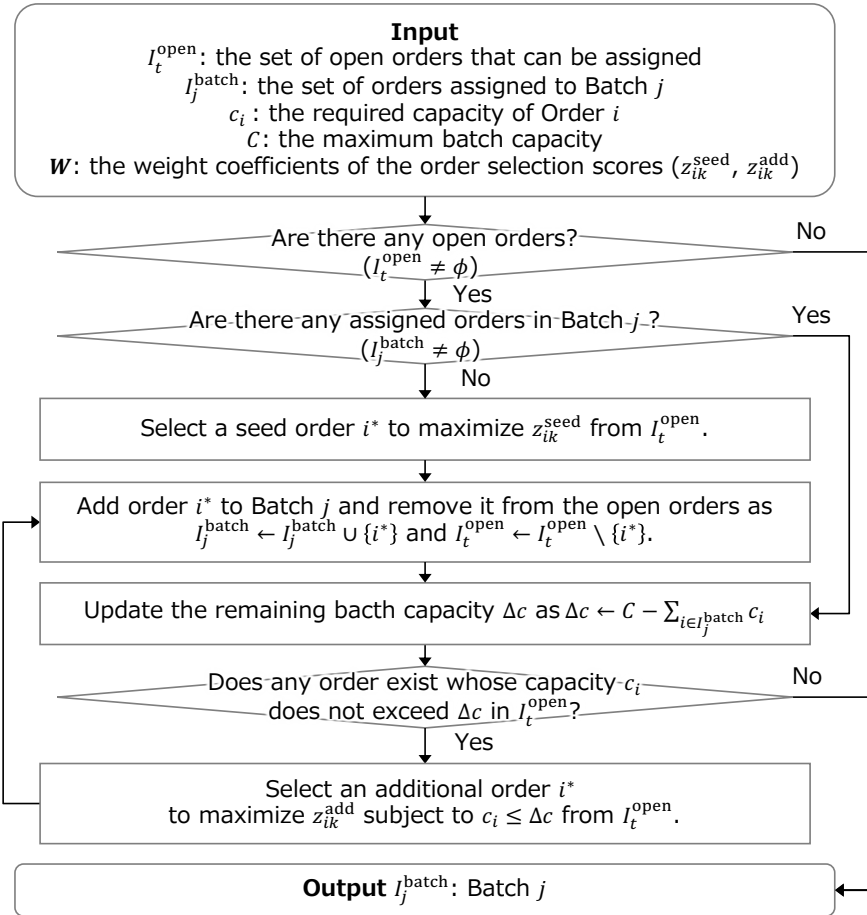


Figure 6.4. Flowchart of seed heuristics using the proposed order selection scores.

Adaptive Hyper-heuristics (AHH)

In this approach, we seek a policy π that, at each batching time t , maps the current system state s_t of the shipping line to a weight vector $\mathbf{W}_t$. Formally, $\mathbf{W}_t = \pi(s_t)$. We denote the proposed batching method based on weights determined dynamically by policy π as $\mathcal{H}_{\mathbf{W}|\pi}$. To obtain the policy via deep reinforcement learning (DRL), we represent π as a neural network π_θ parameterized by θ . The state s_t comprises 28 observations at time t listed in Table 6.1. To reflect progress and utilization in the action, s_t includes the ratios of open (or waiting), executing, and completed orders for each process (#1–12); the utilization of each resource within the processes (#13–21); the simulation time t (#22); the total shipping delay (#23) and the number of tardy orders (#24); and the counts of unfinished orders whose remaining time to due date falls within specified thresholds (#25–28).

We define the reward r_{ht} to be terminal-only: it returns a value based on the objective (Equa-

Table 6.1. DRL state space

#	Definition of state
1–3	Open / Executing / Completed order num. rate in GtP picking
4–6	Open / Executing / Completed order num. rate in PtG picking
7–9	Waiting / Executing / Completed order num. rate in Order consolidation
10–12	Waiting / Executing / Completed order num. rate in Packing
13–16	Inbound-lifter / Outbound-lifter / Shuttle / Picking station utilization rate in GtP picking
17	Picking cart utilization rate in PtG picking
18–20	Inbound-lifter / Outbound-lifter / Shuttle utilization rate in Order consolidation
21	Packing station utilization rate in Packing
22	Simulation time
23	Total shipping-delay time (Tardiness)
24	Num. of tardy orders
25–28	Num. of unfinished orders with due-date $\leq 10 / 30 / 60 / 120$ min.

tion 6.13) if and only if t equals the simulation end time t_{end} , and 0 otherwise, as in Equation 6.30.

$$r_{ht} = \begin{cases} 1 - f(\mathcal{H}_{\mathbf{W}|\pi_\theta}, \mathcal{O}_h, \mathcal{A}_h, \mathcal{D}_h) / \beta, & \text{if } t = t_{\text{end}}, \\ 0, & \text{otherwise.} \end{cases} \quad (6.30)$$

Here, β is a normalization constant satisfying $\beta > f(\mathcal{H}_{\mathbf{W}|\pi_\theta}, \mathcal{O}_h, \mathcal{A}_h, \mathcal{D}_h)$ ($h = 1, 2, \dots, H$).

The target OOBSP involves sparse rewards as in Equation 6.30 and a long horizon with t_{end} exceeding 10,000 steps, which makes learning difficult for standard DRL methods [160]. Therefore, we adopt reinforcement learning based on evolution strategies (neuro-evolution), which is effective in such cases [173], [174]. Here, we use CMA-ES as the evolutionary algorithm and determine θ^* to maximize the average cumulative reward over all training instances — equivalently, to minimize the average objective value f — as in Equation 6.31.

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \frac{1}{H} \sum_{h=1}^H \sum_{t=0}^{t_{\text{end}}} r_{ht} = \underset{\theta}{\operatorname{argmin}} \frac{1}{H} \sum_{h=1}^H f(\mathcal{H}_{\mathbf{W}|\pi_\theta}, \mathcal{O}_h, \mathcal{A}_h, \mathcal{D}_h). \quad (6.31)$$

6.4 Numerical Experiments

6.4.1 Experimental Conditions

In this section, to verify the effectiveness of the proposed method, we conducted numerical experiments using a simulation on the fulfillment-line configuration described above. As shipping

instruction data, we generated randomized shipping-order datasets derived from a real distribution warehouse (408 orders, 308 items) using the following procedure. Of these, five datasets were used for training and 100 for evaluation.

1. From the original shipment data, we created a dataset $\mathcal{S}_h$ by randomly deleting up to 20% of the item types contained in each order.
2. For each item in $\mathcal{S}_h$, we randomly determined the shipping quantity within the range of 0.5 – 2.0 times the original quantity.
3. We randomly determined the arrival time of each shipping box in $\mathcal{S}_h$ and the slack time until the shipping deadline from the uniform distributions $\text{Uniform}(0, 9000)$ [seconds] and $\text{Uniform}(7200, 10800)$ [seconds], respectively.

Batch capacity was measured in terms of the number of orders ($c_i = 1$), with per-batch limits of $C = 16$ orders for GtP and $C = 4$ orders for PtG. Given the practical importance of reducing shipment delays, we conducted preliminary experiments to ensure that total tardiness remained sufficiently small. Based on these results, we set the objective-function parameter to $\alpha = 1$ and the maximum slack time to $\tau = 7200$ [seconds].

We evaluate the proposed approach by comparing makespan and total tardiness across the following eight methods.

- (1) **FCFS**: Orders are batched in the order they are received.
- (2) **Seed heuristics**: Weights are set to $w_{k5} = w_{k6} = w_{k7} = 0$, yielding conventional seed heuristics.
- (3) **Seed + Due-date heuristics**: Weights are set to $w_{k5} = 1, w_{k6} = w_{k7} = 0$, i.e., (2) augmented with due-date priority.
- (4) **Seed + Consolidation heuristics**: Weights are set to $w_{k6} = 1, w_{k5} = w_{k7} = 0$, i.e., (2) augmented with consolidation priority.
- (5) **Seed + Task-simplicity heuristics**: Weights are set to $w_{k7} = 1, w_{k5} = w_{k6} = 0$, i.e., (2) augmented with task-simplicity.
- (6) **Static Hyper-heuristics (SHH)**: The proposed method $\mathcal{HW}^*$ based on the weight coefficients $\mathbf{W}^*$ obtained by solving Equation 6.29.
- (7) **Adaptive Hyper-heuristics (AHH)**: The proposed method $\mathcal{HW}|\pi_{\theta^*}$ based on the policy π_{θ^*} learned by solving Equation 6.31 using DRL.
- (8) **AHH with conventional LHHs (AHHConv)**: A comparison method for verifying the effectiveness of the two proposed LHHs, where the weights of the proposed LHHs are fixed at $w_{k6} = w_{k7} = 0$, and only the remaining five LHH weights ($w_{k1}, \dots, w_{k5}$) are adjusted using a policy π'_{θ} learned in the same manner as in AHH.

Here, methods (1)–(2) are conventional heuristics; methods (3)–(5) are simple extensions of method (2) with additional heuristics; methods (6)–(7) are the proposed methods; and method (8) is a conventional DRL baseline, all compared in the experiments. For (6) SHH, Equation 6.29 was solved using CMA-ES with 32 individuals and 200 generations. For (7) AHH, Equation 6.31 was solved using CMA-ES with 100 individuals and 200 generations. Following Cals et al. [165], for (7) AHH the policy π_θ was implemented as a fully connected neural network with a 28-dimensional input layer, two hidden layers with 64 units each, and a 14-dimensional output layer. We used ReLU for the hidden layers and tanh for the output layer. For (8) AHHConv, which serves as a baseline for validating the effectiveness of the proposed LLHs, the policy π_θ used a fully connected neural network similar to AHH but with a 10-dimensional output layer.

For methods (2)–(5), we identified the combination of seed- and additional-order selection methods that minimized the objective (Equation 6.13) on the training data via brute-force search over 16 patterns. The candidate seed-selection methods (w_{k1}, w_{k2}) were: minimum item count (1, 0), maximum item count (−1, 0), maximum item similarity upper bound (0, 1), and minimum item similarity upper bound (0, −1). The candidate additional-order selection methods (w_{k3}, w_{k4}) were: minimum additional item count (1, 0), maximum additional item count (−1, 0), maximum item similarity (0, 1), and minimum item similarity (0, −1), making 16 patterns in total. All numerical computations were performed in Python 3.11 on a system with a Ryzen 9950X (4.30 GHz, 32 threads) CPU and 128 GB RAM.

6.4.2 Experimental Results

The evaluation results for makespan, total tardiness, and objective values on the validation dataset are shown in Table 6.2. The solutions obtained by each method are listed in Table 6.3. Additionally, a comparison of the objective values is presented in Figure 6.5. From these results, the ranking by objective value (best to worse) is: (7) AHH, (6) SHH, and (4) Seed + Consolidation Heuristics. The best-performing method, AHH, achieved a 51.34% improvement in the objective value relative to FCFS. Breaking down the improvements, both makespan and total tardiness improved compared with FCFS, with a particularly large reduction observed in total tardiness. Moreover, compared with (8) AHHConv, AHH further improved the objective value by 20.0%, confirming the effectiveness of the two newly proposed LLHs.

A comparison of the average cumulative count of completed orders over time for methods (7) AHH, (8) AHHConv, and (1) FCFS on the validation dataset is shown in Figure 6.6. The four subplots report results for the GtP picking (top left), PtG picking (top right), consolidation (bottom left), and packing (bottom right) processes, respectively. The consolidation process completes

fastest under AHH, followed by AHHConv and then FCFS. Notably, although FCFS attains the earliest completions in GtP picking, its completions in consolidation are delayed, which in turn slows completions in packing. By contrast, AHH yields the fastest completions in consolidation,

Table 6.2. Optimization results of the multiple OOBSP solution methods

Dispatching method	Makespan [hour]	Tardiness [hour]	Objective [hour]
(1) FCFS	3.137 ± 0.157 (0.00%)	4.851 ± 4.429 (0.00%)	7.988 ± 4.519 (0.00%)
(2) Seed heuristics	3.082 ± 0.138 (-1.74%)	1.728 ± 2.135 (-64.39%)	4.810 ± 2.200 (-39.79%)
(3) Seed + Due-date heuristics	3.092 ± 0.111 (-1.46%)	1.874 ± 2.877 (-61.38%)	4.953 ± 2.965 (-37.99%)
(4) Seed + Consolidation heuristics	3.055 ± 0.145 (-2.61%)	1.185 ± 1.763 (-75.57%)	4.240 ± 1.808 (-46.92%)
(5) Seed + Task-simplicity heuristics	3.139 ± 0.155 (+0.08%)	4.882 ± 4.301 (+0.62%)	8.021 ± 4.405 (+0.41%)
(6) SHH: Static Hyper-heuristics	3.049 ± 0.141 (-2.81%)	0.933 ± 1.318 (-80.77%)	3.982 ± 1.357 (-50.15%)
(7) AHH: Adaptive Hyper-heuristics	3.059 ± 0.139 (-2.49%)	0.829 ± 1.400 (-82.92%)	3.887 ± 1.451 (-51.34%)
(8) AHHConv: AHH with conventional LHHs	3.082 ± 0.140 (-1.74%)	1.774 ± 2.141 (-63.44%)	4.856 ± 2.183 (-39.21%)

Table 6.3. Obtained solutions with the multiple OOBSP solution methods

Dispatching method	Weights (For AHH, averaged weights in simulation)
(1) FCFS	—
(2) Seed heuristics	$w_1 = [0, -1, 0, 1, 0, 0, 0], w_2 = [1, 0, 1, 0, 0, 0, 0]$
(3) Seed + Due-date heuristics	$w_1 = [0, 1, 0, 1, 1, 0, 0], w_2 = [-1, 0, 0, 1, 1, 0, 0]$
(4) Seed + Consolidation heuristics	$w_1 = [1, 0, 0, 1, 0, 1, 0], w_2 = [0, 1, 0, 1, 0, 1, 0]$
(5) Seed + Task-simplicity heuristics	$w_1 = [0, 1, 1, 0, 0, 0, 1], w_2 = [1, 0, 1, 0, 0, 0, 1]$
(6) SHH: Static Hyper-heuristics	$w_1 = [-0.34, 0.89, -0.29, 0.79, -0.51, 0.99, -0.91]$ $w_2 = [0.13, 0.48, -0.74, -0.09, 0.99, 0.90, 0.16]$
(7) AHH: Adaptive Hyper-heuristics	$w_1 = [-0.29, -0.49, 0.99, 0.82, -0.31, 0.09, -1.00]$ $w_2 = [-0.35, 0.77, -0.98, 0.99, 0.66, -0.11, 0.45]$
(8) AHHConv: AHH with conventional LHHs	$w_1 = [0.59, -0.70, 0.53, -0.15, -0.31, 0, 0]$ $w_2 = [0.67, -0.91, -0.92, 0.36, -1.00, 0, 0]$

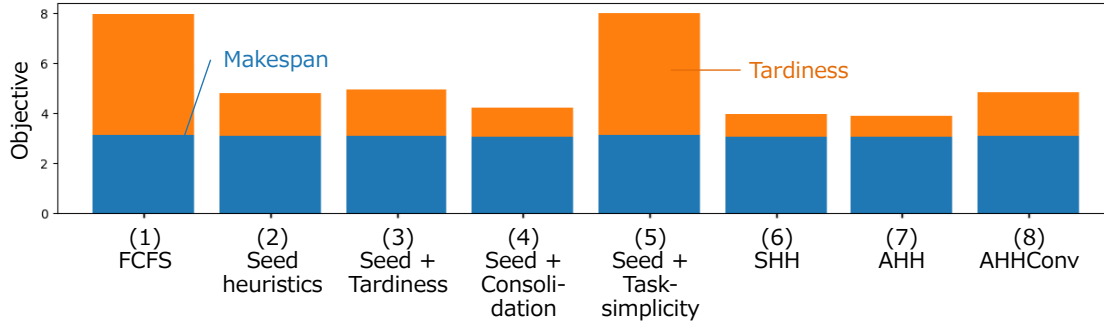


Figure 6.5. Objective value comparison for each batching method.

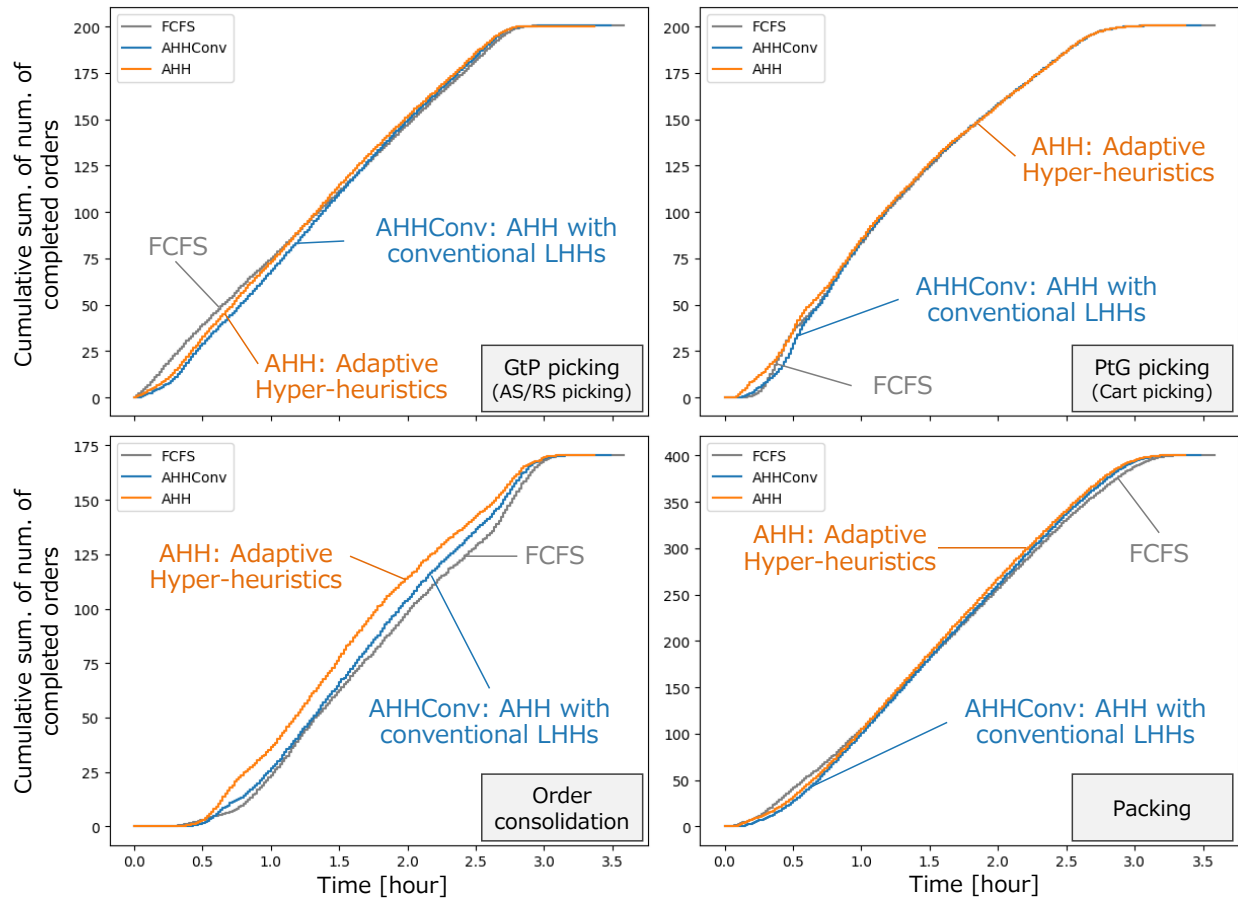


Figure 6.6. Averaged cumulative summation of the number of the completed orders in GtP picking, PtG picking, order consolidation, and packing processes.

thereby reducing both makespan and total tardiness.

Next, Figure 6.7 shows histograms of turnover time (the elapsed time from order receipt to completion of work) for AHH, AHHConv, and FCFS on the validation dataset, stratified by

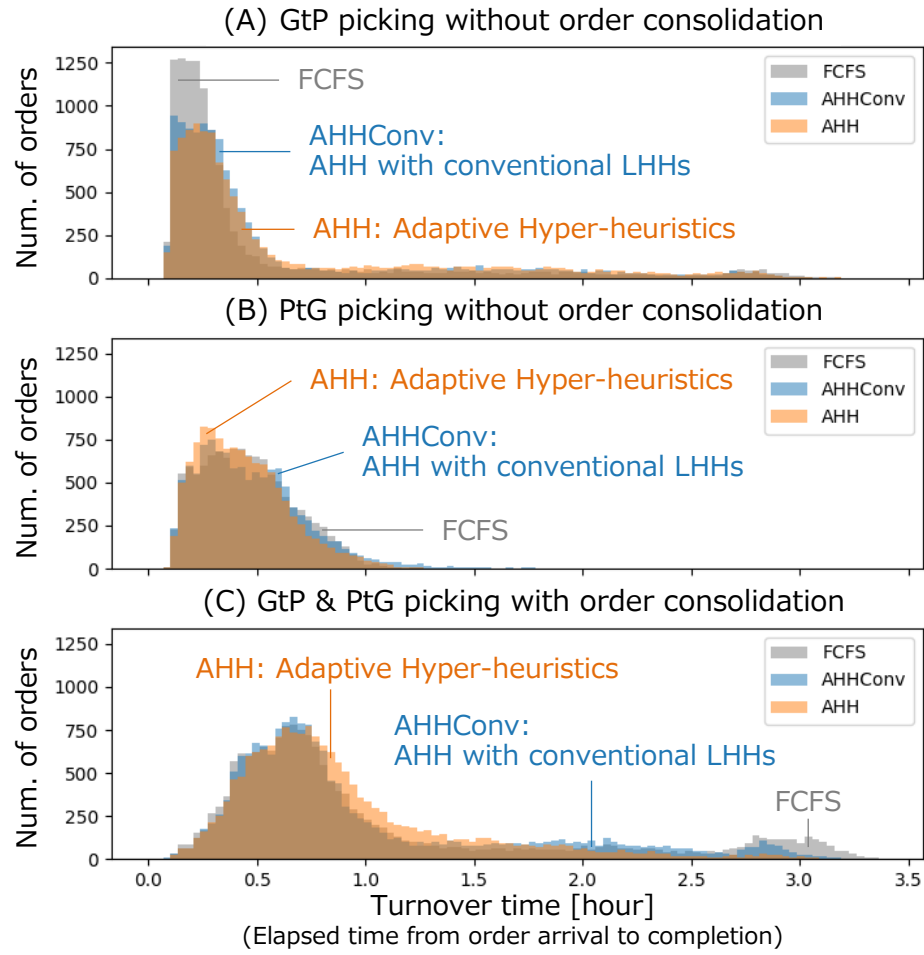


Figure 6.7. Histograms of each order's elapsed times from arrival to completion by processing paths.

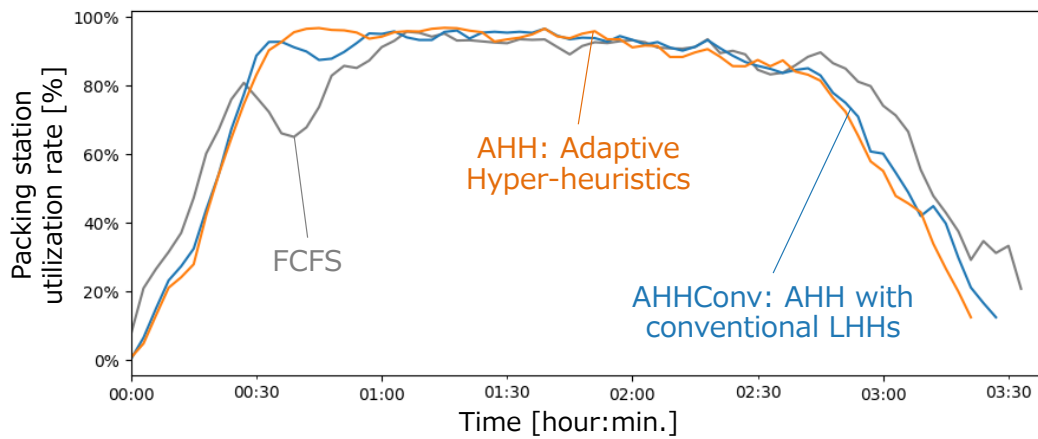


Figure 6.8. Averaged picking station utilization rate.

shipping-process paths. The shipping-process paths are:

- (A) Direct path from GtP picking to packing;
- (B) Direct path from PtG picking to packing; and
- (C) Path from both GtP and PtG picking via consolidation to packing.

Under path (C), both AHHConv and FCFS exhibit many orders with turnover times exceeding 2.5 hours (9,000 seconds). In contrast, AHH markedly reduces the occurrence of such long turnover times, which likely contributes to the improved objective value.

Figure 6.8 also presents the time series of packing-station utilization for AHH, AHHConv, and FCFS on the validation dataset. The figure shows that packing-station utilization under AHH is consistently higher than under AHHConv and FCFS. This is attributable to the combined effects of the proposed Consolidation Priority Score, which accelerates completions in the consolidation process, and the Task-Simplicity Score, which enhances utilization of the packing process, thereby enabling more efficient packing operations.

Finally, Figure 6.9 shows the time series of the average weight vector $\mathbf{W}_t$ produced by the AHH policy π_θ during simulations on the validation dataset. Each subfigure presents the time series of weights for the seven score types, where blue ($k = 1$) denotes GtP picking and orange ($k = 2$) denotes PtG picking. The figure shows that, in GtP picking, the weight on the Additional Item Count Score (#3) remains high; emphasis shifts from the Additional Item Similarity Score (#4) in the early stage, to the Due-Date Priority Score (#5) in the middle stage, and to the Consolidation Priority Score (#6) in the late stage. In PtG picking, in addition to a high weight on the Additional Item Similarity Score (#4), the Consolidation Priority Score (#6) and the Task-Simplicity Score (#7) are emphasized in the early stage, while the Item-Similarity Upper-Bound Score (#2) and the Due-Date Priority Score (#5) gain importance in the middle-to-late stages. Thus, AHH dynamically adjusts batching strategies as operations progress, explaining its performance advantage relative to SHH, which relies on fixed weights. These results confirm both (i) the benefit of explicitly modeling consolidation effects and (ii) that effective OOBSP policies can be synthesized via hyper-heuristics that appropriately combine multiple evaluation indicators.

6.5 Conclusion

In this study, we proposed a hyper-heuristics framework for solving the OOBSP that minimizes both makespan and tardiness by dynamically controlling batching and sequencing in a distribution warehouse with two picking modes: GtP and PtG. The proposed method focuses on post-picking order consolidation, which has a substantial impact on performance metrics. To address this, we introduced two new heuristics designed to shorten the time to consolidation completion. In addition,

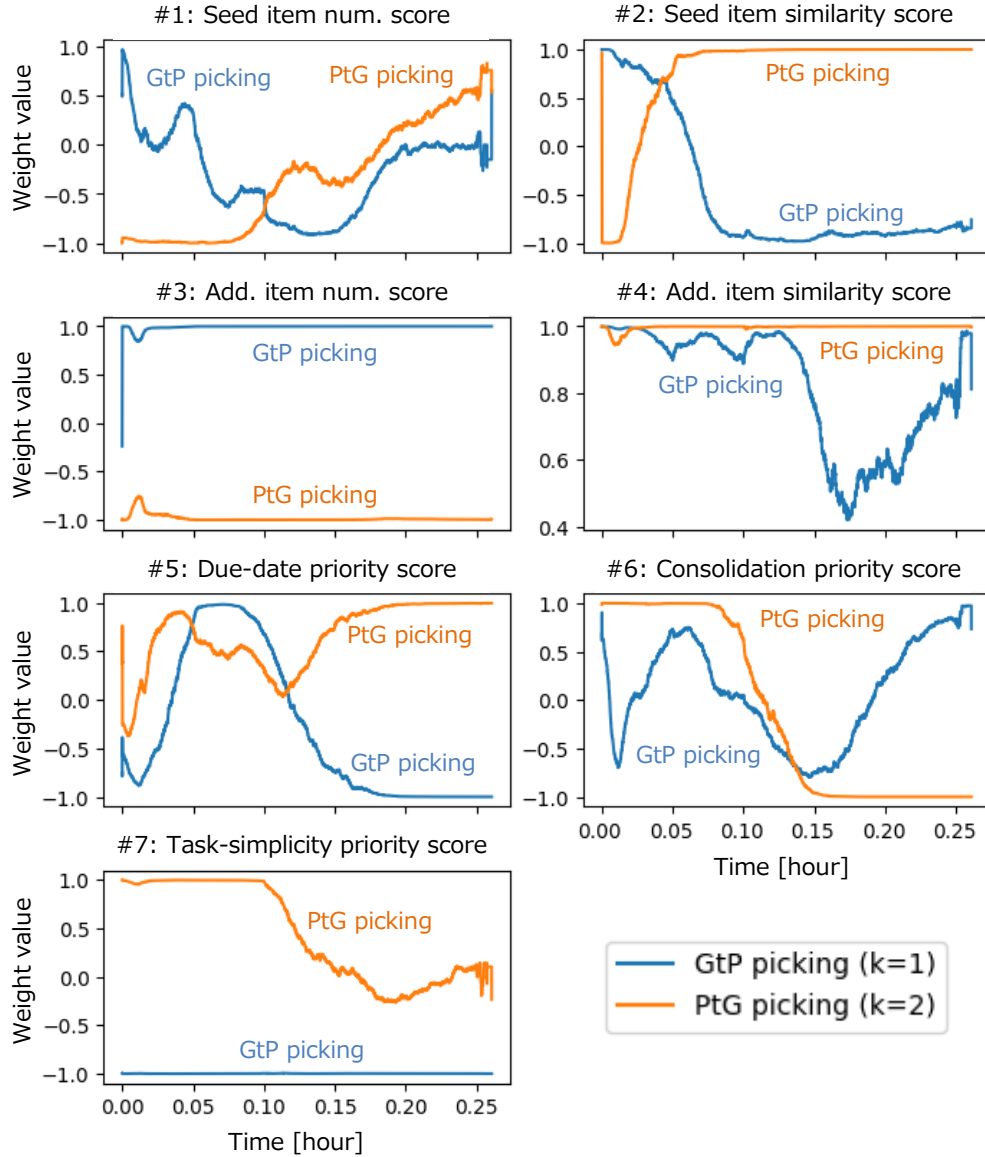


Figure 6.9. Time-series score weights obtained from the policy π_θ .

by combining these with five existing heuristics (seven in total) and tuning their weights to problem characteristics, we developed a hyper-heuristics approach that improves performance. As hyper-heuristic strategies, we consider two variants: SHH, which uses fixed weights, and AHH, which adapts the weights online via DRL. Simulation experiments demonstrate that AHH significantly reduces both makespan and tardiness compared with FCFS and seed heuristics. Taken together, these results indicate that the proposed method effectively shortens shipping operation times and reduces delays in distribution warehouses.

On the other hand, the performance evaluation in this study was limited to a single fulfillment

line configuration and a limited set of real-world shipment data. Future work includes assessing the sensitivity of performance to line configurations and order data characteristics. Moreover, as noted by Cals et al. [165], postponing batch formation can, in some cases, improve overall efficiency in online batching; incorporating such mechanisms could further enhance the proposed method. Beyond OOBSP, many combinatorial optimization problems in production and logistics—such as production scheduling, workforce allocation, and delivery planning—are typically addressed with heuristics due to the large scale of real-world instances. We plan to extend the evaluation of our approach to these problems in future work.

Acknowledgment

This research was conducted with the support of Hitachi Industrial Products, Ltd. We would like to express our gratitude to Mr. Shun Motoyama and Mr. Shinji Takeda for providing practical insights and data from real operations, as well as for their valuable advice on the progress of this research.

Chapter 7

Discussion and General Conclusion

7.1 Summary of Case Studies and Key Insights

This dissertation addresses advanced decision-making challenges in contemporary manufacturing and logistics systems, specifically (A) escalating structural and operational complexity driven by automation, and (B) the requirement for robustness under nonstationary and uncertain environments. In Chapter 1, decision problems are organized into three hierarchical layers: (1) strategic (2-5 years), (2) tactical (several months to 2 years), and (3) operational (real-time to weekly). For each layer, Chapters 2-6 develop methods that jointly pursue model *fidelity* — faithful representation of system dynamics — and decision *optimality* — high-quality solutions within a reasonable computational time — through five complementary case studies.

For the strategic layer, Chapter 2 develops a design-support methodology that embeds dynamic multi-robot coordination into the conceptual layout design of robot-cell manufacturing systems (RCMS). Departing from conventional static formulations, the proposed design approach jointly optimizes layout configuration and robot motion scheduling. To improve fidelity, analytical performance models of cycle time, layout area, and manipulability are coupled with a lightweight robot motion simulation. The proposed chromosome encoding plus problem-aware crossover and mutation operators are implemented within a multiobjective genetic algorithm for higher optimality by efficient exploration of the vast design space. Computational experiments demonstrate that the method produces well-distributed Pareto fronts and enables rigorous quantification of trade-offs among the three performance metrics.

At the tactical layer, Chapter 3 tackles scenario generation for simulation-based stochastic programs, where naïve enumeration is computationally untenable. Traditional distribution-driven methods rely on analytical models and cannot be applied when simulation is required. To bridge this gap, the chapter introduces inferred cost-space scenario clustering (ICSSC), a surrogate-assisted approach that works across modeling paradigms and fidelity levels, including simulation. A learned surrogate model approximates expensive simulation evaluations, enabling estimation of decision-relevant cost geometry and clustering scenarios into a compact, decision-faithful set. Computational studies on portfolio optimization, server location, and inventory placement show that ICSSC achieves higher-optimality solutions with dramatically fewer scenarios than conventional

techniques.

Chapter 4 formulates a capacitated dynamic joint replenishment problem with time-varying transport costs (CDJRP-TC) to support tactical S&OP under nonstationary prices. The problem integrates sales planning, multi-item inventory control, and shared transport capacity with period-dependent costs, however direct solution of the resulting large-scale mixed-integer linear program (MILP) is computationally prohibitive. To overcome this, an analytical model enhancement (AME) procedure is used: (i) solve a simplified MILP to obtain globally cost-minimal inventory trajectories under time-varying unit costs; (ii) infer parameters of discrete inventory policies; and (iii) iteratively refine target inventories via high-fidelity simulation. Experiments show the method yields high-optimality solutions within short runtimes, including for large instances that defeat conventional approaches.

In the operational layer, Chapter 5 tackles short-horizon scheduling by accelerating order sequencing in automated picking systems using surrogate-assisted search. Conventionally, surrogate-assisted optimization has focused on continuous problems and is less suited to sequence optimization. To close this gap, the chapter proposes a surrogate-assisted search method tailored to sequences. A bidirectional Bayesian recurrent neural network is pretrained as a surrogate for high-fidelity simulation, and combined with simulated annealing to prune weak candidates early and concentrate computation on promising regions. Experiments on representative instances show order-of-magnitude speedups over the baseline search without loss of solution optimality. These results demonstrate that learned surrogates can reliably replace repeated simulations when the sequence-to-performance mapping is sufficiently accurate.

Chapter 6 addresses real-time operational control in an automated fulfillment line that integrates goods-to-person (GtP) and person-to-goods (PtG) processes. Treating consolidation as the bottleneck, it introduces two new heuristics and embeds them, together with five established ones, in a hyper-heuristic framework. Both a fixed-weight switching scheme and a reinforcement-learning-based adaptive weighting scheme are evaluated, with parameters tuned via high-fidelity simulation to minimize the sum of makespan and tardiness. Results show simultaneous reductions in both metrics and indicate that learning-based adaptation is beneficial when arrivals and process conditions vary online. Although assessed on a limited line configuration and dataset, the study demonstrates that adaptive hyper-heuristics and reinforcement learning can deliver tangible operational gains.

7.2 Proposed Decision-making Framework

Integrating the five studies from Chapters 2-6, this dissertation presents a unified decision-making framework for modern manufacturing and logistics. Figure 7.1 summarizes its structure and key

components.

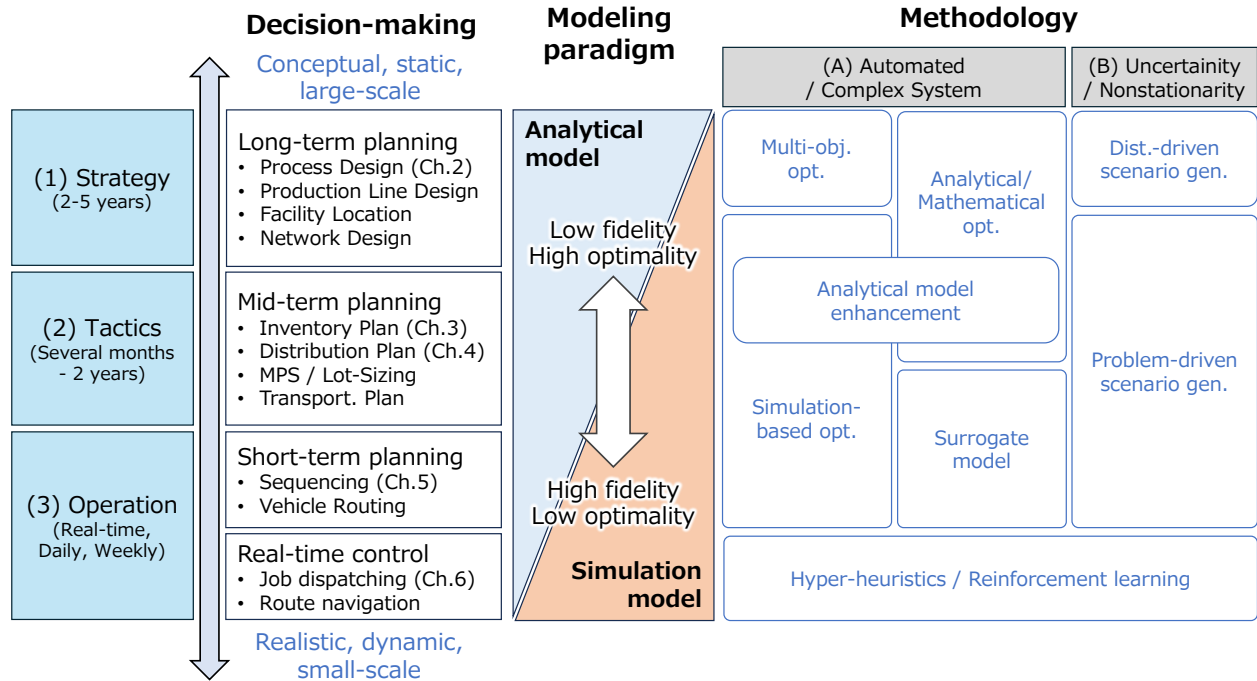


Figure 7.1. Proposed decision-making framework for modern manufacturing and logistics systems

Strategic decisions have broad temporal and spatial impact, involve major capital, and are made without fine-grained operational data. Their models are therefore conceptual, mostly static, and large-scale, requiring comprehensive evaluation of alternative designs and policies. Multiobjective optimization built on analytical models that capture system-level behavior is appropriate, with lightweight simulation added when dynamic coordination materially affects outcomes (e.g., the RCMS layout in Chapter 2). Similar problems include line design for factories and warehouses, facility location, and network design, where many alternatives must be assessed. Accounting for dynamic interactions, such as buffer capacities between processes, can substantially improve design accuracy. For uncertainty, distribution-driven scenario generation is suitable when analytical models suffice; when simulation is required, problem-driven, surrogate-assisted methods like ICSSC (Chapter 3) yield more robust solutions.

At the tactical layer, medium- to large-scale, multi-facility, multi-period, multi-item decisions require higher-fidelity models that reflect nonstationarity and uncertainty. This includes inventory placement (Chapter 3), joint replenishment (Chapter 4), master production scheduling, and transportation planning, where multi-month plans must satisfy equipment, transport capacity, and other operational constraints. Many of these problems admit tractable analytical formulations when selected constraints or dynamics are relaxed. In these cases, an AME approach — global search on the relaxed analytical model followed by simulation-based evaluation and refinement — is effective.

Under uncertainty, surrogate-assisted, problem-driven scenario generation (e.g., ICSSC) provides compact, decision-faithful scenario sets and improves robustness.

At the operational layer, detailed information are available and decisions are narrow in scope yet must be executed immediately with little chance of revision. High-fidelity models, often fine-grained discrete-event simulation, are therefore essential. For short daily or weekly horizons (e.g., job sequencing or vehicle routing), surrogate-assisted search that replaces expensive simulation calls yields fast, high-quality schedules (Chapter 5). For high-frequency real-time control (e.g., dispatching or route navigation), adaptive hyper-heuristics that select among multiple heuristics (Chapter 6) are effective. Because execution follows decisions almost immediately, deferring commitment until the last feasible moment and acting on current information is often better than spending longer time to compute a nominally robust plan.

In summary, choosing modeling paradigms and solution methods that fit the strategic, tactical, and operational layers strikes a practical balance between fidelity and optimality, enabling efficient management of modern manufacturing and logistics systems.

7.3 Conclusion and Future Directions

Finally, I summarize the dissertation's contributions and outline future directions. This work addresses two core challenges in modern manufacturing and logistics: (A) rising structural and operational complexity due to automation and (B) the need for robustness under nonstationary, uncertain conditions. Decision making is organized into three layers — strategic, tactical, and operational — and five studies are tailored to the characteristics of each layer. Integrating these studies, the dissertation proposes a unified decision-making framework and discusses its applicability to problems with similar properties.

Future work should validate the framework more broadly and extend it toward a general, reusable decision-making architecture. Although five case studies illustrate the approach, manufacturing decisions are highly diverse: differences in products and sites introduce distinct objectives, constraints, and operating rules, so the current framework is not universally applicable. Nevertheless, its hierarchical organization and layer-specific criteria for balancing fidelity and optimality are widely transferable. I expect it to serve as a practical starting point for formulating new problems and selecting or developing tailored solution methods.

References

- [1] W. J. Hopp and M. L. Spearman, *Factory Physics third edition*. 2008, ISBN: 9781577667391.
- [2] A. A. Tubis and J. Rohman, “Intelligent Warehouse in Industry 4.0—Systematic Literature Review,” *Sensors*, vol. 23, no. 8, 2023, ISSN: 14248220. DOI: 10.3390/s23084105.
- [3] H. He, T. Brush, H. Cai, D. Thakkar, S. Dunlop, and S. Biller, “Developing a strategic supply chain resiliency index: an assessment of the global lithium supply for U.S. automotive EV battery production,” *International Journal of Production Research*, vol. 7543, 2025, ISSN: 1366588X. DOI: 10.1080/00207543.2025.2534842. [Online]. Available: <https://doi.org/10.1080/00207543.2025.2534842>.
- [4] D. Ivanov, “Viable supply chain model: integrating agility, resilience and sustainability perspectives—lessons from and thinking beyond the COVID-19 pandemic,” *Annals of Operations Research*, vol. 319, no. 1, pp. 1411–1431, 2022, ISSN: 15729338. DOI: 10.1007/s10479-020-03640-6. [Online]. Available: <https://doi.org/10.1007/s10479-020-03640-6>.
- [5] European Environmental Agency-EEA, *Reducing greenhouse gas emissions from heavy-duty vehicles in Europe*, 2022. [Online]. Available: https://www.eea.europa.eu/en/analysis/publications/reducing-greenhouse-gas-emissions-from-heavy-duty-vehicles-in-europe?utm%7B%5C_%7Dsource=chatgpt.com (visited on 09/15/2025).
- [6] D. Stüve, R. van der Meer, M. S. Ali Agha, and M. Lütke Entrup, “A systematic literature review of modelling approaches and implementation of enabling software for supply chain planning in the food industry,” *Production and Manufacturing Research*, vol. 10, no. 1, pp. 470–493, 2022, ISSN: 21693277. DOI: 10.1080/21693277.2022.2091057. [Online]. Available: <https://doi.org/10.1080/21693277.2022.2091057>.
- [7] D. Ivanov, A. Tsipoulanidis, and J. Schönberger, *Global Supply Chain and Operations Management*. 2019, p. 593. [Online]. Available: <http://www.springer.com/series/10099>.
- [8] H. Stadtler, C. Kilger, and H. Meyr, Eds., *Supply Chain Management and Advanced Planning* (Springer Texts in Business and Economics). Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, ISBN: 978-3-642-55308-0. DOI: 10.1007/978-3-642-55309-7. [Online]. Available: <https://link.springer.com/10.1007/978-3-642-55309-7>.

- [9] S. Chopra and P. Meindl, "Supply Chain Management. Strategy, Planning & Operation," in *Das Summa Summarum des Management*, Wiesbaden: Gabler, 2014. DOI: 10.1007/978-3-8349-9320-5_22. [Online]. Available: http://link.springer.com/10.1007/978-3-8349-9320-5%7B%5C_%7D22.
- [10] S. H. Owen and M. S. Daskin, "Strategic facility location: A review," *European Journal of Operational Research*, vol. 111, no. 3, pp. 423–447, 1998, ISSN: 03772217. DOI: 10.1016/S0377-2217(98)00186-6.
- [11] R. Z. Farahani, M. SteadieSeifi, and N. Asgari, "Multiple criteria facility location problems: A survey," *Applied Mathematical Modelling*, vol. 34, no. 7, pp. 1689–1709, 2010, ISSN: 0307904X. DOI: 10.1016/j.apm.2009.10.005. [Online]. Available: <http://dx.doi.org/10.1016/j.apm.2009.10.005>.
- [12] F. Misni and L. S. Lee, "A Review on Strategic, Tactical and Operational Decision Planning in Reverse Logistics of Green Supply Chain Network Design," *Journal of Computer and Communications*, vol. 05, no. 08, pp. 83–104, 2017, ISSN: 2327-5219. DOI: 10.4236/jcc.2017.58007.
- [13] G. B. Dantzig, "Mathematical Programming: An Overview: 6 - Integration of Strategic and Tactical Planning in the Aluminum Industry," pp. 269–290,
- [14] M. T. Melo, S. Nickel, and F. Saldanha-da-Gama, "Facility location and supply chain management - A review," *European Journal of Operational Research*, vol. 196, no. 2, pp. 401–412, 2009, ISSN: 03772217. DOI: 10.1016/j.ejor.2008.05.007. [Online]. Available: <http://dx.doi.org/10.1016/j.ejor.2008.05.007>.
- [15] J. M. V. Cedillo and J. C. P. García, "Sales and Operations Planning: A Business Practice to Align Supply Chains," *International Journal of Advanced Engineering, Management and Science*, vol. 5, no. 4, pp. 250–261, 2019. DOI: 10.22161/ijaems.5.4.3.
- [16] J. A. Grimson and D. F. Pyke, "Sales and operations planning: An exploratory study and framework," *The International Journal of Logistics Management*, vol. 18, no. 3, pp. 322–346, 2007, ISSN: 17586550. DOI: 10.1108/09574090710835093.
- [17] A. M. T. Thomé, L. F. Scavarda, N. S. Fernandez, and A. J. Scavarda, "Sales and operations planning: A research synthesis," *International Journal of Production Economics*, vol. 138, no. 1, pp. 1–13, 2012, ISSN: 09255273. DOI: 10.1016/j.ijpe.2011.11.027.
- [18] Y. Nemati, M. Madhoshi, and A. S. Ghadikolaei, "The effect of Sales and Operations Planning (S&OP) on supply chain's total performance: A case study in an Iranian dairy company," *Computers and Chemical Engineering*, vol. 104, pp. 323–338, 2017, ISSN:

00981354. DOI: 10.1016/j.compchemeng.2017.05.002. [Online]. Available: <http://dx.doi.org/10.1016/j.compchemeng.2017.05.002>.
- [19] Z. Liu and A. Nagurney, "Multiperiod competitive supply chain networks with inventorying and a transportation network equilibrium reformulation," *Optimization and Engineering*, vol. 13, no. 3, pp. 471–503, 2012, ISSN: 13894420. DOI: 10.1007/s11081-011-9170-2.
- [20] M. L. Pinedo, *Scheduling*. Cham: Springer International Publishing, 2022, p. 691, ISBN: 978-3-031-05920-9. DOI: 10.1007/978-3-031-05921-6. [Online]. Available: <https://link.springer.com/10.1007/978-3-031-05921-6>.
- [21] M. M. Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems With Time Window Constraints.," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987, ISSN: 0030364X. DOI: 10.1287/opre.35.2.254.
- [22] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury, "Simulation optimization: a review of algorithms and applications," *Annals of Operations Research*, vol. 240, no. 1, pp. 351–380, 2016, ISSN: 15729338. DOI: 10.1007/s10479-015-2019-x. arXiv: 1706.08591.
- [23] R. D. Tordecilla, A. A. Juan, J. R. Montoya-Torres, C. L. Quintero-Araujo, and J. Panadero, "Simulation-optimization methods for designing and assessing resilient supply chain networks under uncertainty scenarios: A review," *Simulation Modelling Practice and Theory*, vol. 106, no. March 2020, 2021, ISSN: 1569190X. DOI: 10.1016/j.simpat.2020.102166.
- [24] E. Acar, G. Bayrak, Y. Jung, I. Lee, P. Ramu, and S. S. Ravichandran, *Modeling, analysis, and optimization under uncertainties: a review*. Springer Berlin Heidelberg, 2021, vol. 64, pp. 2909–2945, ISBN: 0015802103026. DOI: 10.1007/s00158-021-03026-7. [Online]. Available: <https://doi.org/10.1007/s00158-021-03026-7>.
- [25] T. Rajapakshe, M. Dawande, and C. Sriskandarajah, "Quantifying the impact of layout on productivity: An analysis from robotic-cell manufacturing," *Operations Research*, vol. 59, no. 2, pp. 440–454, 2011, ISSN: 0030364X. DOI: 10.1287/opre.1100.0874.
- [26] D. Barral, J. P. Perrin, E. Dombre, and A. Liégeois, "Simulated annealing combined with a constructive algorithm for optimising assembly workcell layout," *International Journal of Advanced Manufacturing Technology*, vol. 17, no. 8, pp. 593–602, 2001, ISSN: 02683768. DOI: 10.1007/s001700170143.
- [27] M. L. Tay and B. K. Ngoi, "Optimising robot workcell layout," *International Journal of Advanced Manufacturing Technology*, vol. 12, no. 5, pp. 377–385, 1996, ISSN: 02683768. DOI: 10.1007/BF01179814.

- [28] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996, ISSN: 02780070. DOI: 10.1109/43.552084.
- [29] A. Drakidis, R. J. Mack, and R. E. Massara, "Packing-based VLSI module placement using genetic algorithm with sequence-pair representation," *IEE Proceedings - Circuits, Devices and Systems*, vol. 153, no. 6, pp. 545–551, 2006. DOI: 10.1049/ip-cds:20050134. [Online]. Available: <https://digital-library.theiet.org/doi/abs/10.1049/ip-cds%7B%5C%7D3A20050134>.
- [30] Q. Liu and R. D. Meller, "A sequence-pair representation and MIP-model-based heuristic for the facility layout problem with rectangular departments," *IIE Transactions (Institute of Industrial Engineers)*, vol. 39, no. 4, pp. 377–394, 2007, ISSN: 0740817X. DOI: 10.1080/07408170600844108.
- [31] R. D. Meller, W. Chen, and H. D. Sherali, "Applying the sequence-pair representation to optimal facility layout designs," *Operations Research Letters*, vol. 35, no. 5, pp. 651–659, 2007, ISSN: 01676377. DOI: 10.1016/j.orl.2006.10.007.
- [32] K. Izui, Y. Murakumo, I. Suemitsu, S. Nishiwaki, A. Noda, and T. Nagatani, "Multiobjective layout optimization of robotic cellular manufacturing systems," *Computers and Industrial Engineering*, vol. 64, no. 2, pp. 537–544, 2013, ISSN: 03608352. DOI: 10.1016/j.cie.2012.12.003. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2012.12.003>.
- [33] S. Emami and A. S. Nookabadi, "Managing a new multi-objective model for the dynamic facility layout problem," *International Journal of Advanced Manufacturing Technology*, vol. 68, no. 9-12, pp. 2215–2228, 2013, ISSN: 14333015. DOI: 10.1007/s00170-013-4820-5.
- [34] A. Saraswat, U. Venkatadri, and I. Castillo, "A framework for multi-objective facility layout design," *Computers and Industrial Engineering*, vol. 90, pp. 167–176, 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2015.09.006.
- [35] Z. Taha, F. Tahriri, and A. Zuhdi, "Job sequencing and layout optimization in virtual production line," *Journal of Quality*, vol. 18, no. 4, pp. 351–374, 2011, ISSN: 10220690.
- [36] H. Gultekin, M. S. Akturk, and O. Ekin Karasan, "Scheduling in a three-machine robotic flexible manufacturing cell," *Computers and Operations Research*, vol. 34, no. 8, pp. 2463–2477, 2007, ISSN: 03050548. DOI: 10.1016/j.cor.2005.09.015.

- [37] D. Barral, J. P. Perrin, E. Dombre, and A. Liegeois, "Development of optimization tools in the context of an industrial robotic CAD software product," *International Journal of Advanced Manufacturing Technology*, vol. 15, no. 11, pp. 822–831, 1999, ISSN: 02683768. DOI: 10.1007/s001700050138.
- [38] W. Bu, Z. Liu, and J. Tan, "Industrial robot layout based on operation sequence optimisation," *International Journal of Production Research*, vol. 47, no. 15, pp. 4125–4145, 2009, ISSN: 00207543. DOI: 10.1080/00207540802001525.
- [39] K. S. N. Ripon, K. Glette, M. Hovin, and J. Torresen, "Job shop scheduling with transportation delays and layout planning in manufacturing systems: A multi-objective evolutionary approach," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7326 LNAI, pp. 209–219, 2012, ISSN: 03029743. DOI: 10.1007/978-3-642-31368-4_25.
- [40] N. Izadinia, K. Eshghi, and M. H. Salmani, "A robust model for multi-floor layout problem," *Computers and Industrial Engineering*, vol. 78, pp. 127–134, 2014, ISSN: 03608352. DOI: 10.1016/j.cie.2014.09.023. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2014.09.023>.
- [41] M. H. Salmani, K. Eshghi, and H. Neghabi, "A bi-objective MIP model for facility layout problem in uncertain environment," *International Journal of Advanced Manufacturing Technology*, vol. 81, no. 9-12, pp. 1563–1575, 2015, ISSN: 14333015. DOI: 10.1007/s00170-015-7290-0.
- [42] D. Zhang and L. Qi, "Virtual engineering: optimal cell layout method for improving productivity for industrial robot," *2008 IEEE International Conference on Robotics, Automation and Mechatronics, RAM 2008*, no. 31, pp. 6–11, 2008. DOI: 10.1109/RAMECH.2008.4690870.
- [43] T. Yoshikawa, "Manipulability of Robotic Mechanisms.," pp. 439–446, 1985.
- [44] M. Solimanpur and M. A. Kamran, "Solving facilities location problem in the presence of alternative processing routes using a genetic algorithm," *Computers and Industrial Engineering*, vol. 59, no. 4, pp. 830–839, 2010, ISSN: 03608352. DOI: 10.1016/j.cie.2010.08.010. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2010.08.010>.
- [45] R. Şahin, K. Ertoral, and O. Türkbey, "A simulated annealing heuristic for the dynamic layout problem with budget constraint," *Computers and Industrial Engineering*, vol. 59, no. 2, pp. 308–313, 2010, ISSN: 03608352. DOI: 10.1016/j.cie.2010.04.013.

- [46] Komarudin and K. Y. Wong, "Applying Ant System for solving Unequal Area Facility Layout Problems," *European Journal of Operational Research*, vol. 202, no. 3, pp. 730–746, 2010, ISSN: 03772217. DOI: 10.1016/j.ejor.2009.06.016. [Online]. Available: <http://dx.doi.org/10.1016/j.ejor.2009.06.016>.
- [47] C. Ou-Yang and A. Utamima, "Hybrid Estimation of Distribution Algorithm for solving Single Row Facility Layout Problem," *Computers and Industrial Engineering*, vol. 66, no. 1, pp. 95–103, 2013, ISSN: 03608352. DOI: 10.1016/j.cie.2013.05.018. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2013.05.018>.
- [48] H. Samarghandi, P. Taabayan, and F. F. Jahantigh, "A particle swarm optimization for the single row facility layout problem," *Computers and Industrial Engineering*, vol. 58, no. 4, pp. 529–534, 2010, ISSN: 03608352. DOI: 10.1016/j.cie.2009.11.015. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2009.11.015>.
- [49] T. Dokeroglu, "Hybrid teaching-learning-based optimization algorithms for the Quadratic Assignment Problem," *Computers and Industrial Engineering*, vol. 85, pp. 86–101, 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2015.03.001. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2015.03.001>.
- [50] S. Nakaya, T. Koide, and S. Wakabayashi, "Adaptive genetic algorithm for VLSI floorplaning based on sequence-pair," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 65–68, 2000, ISSN: 02714310. DOI: 10.1109/ISCAS.2000.855997.
- [51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002, ISSN: 1089778X. DOI: 10.1109/4235.996017.
- [52] K. Izui, T. Yamada, S. Nishiwaki, and K. Tanaka, "Multiobjective optimization using an aggregative gradient-based method," *Structural and Multidisciplinary Optimization*, vol. 51, no. 1, pp. 173–182, 2015, ISSN: 16151488. DOI: 10.1007/s00158-014-1125-8.
- [53] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999, ISSN: 1089778X. DOI: 10.1109/4235.797969.
- [54] M. Chen, S. Mehrotra, and D. Papp, "Scenario generation for stochastic optimization problems via the sparse grid method," *Computational Optimization and Applications*, vol. 62, no. 3, pp. 669–692, 2015, ISSN: 15732894. DOI: 10.1007/s10589-015-9751-7.
- [55] R. Henrion and W. Römis, "Problem-based optimal scenario generation and reduction in stochastic programming," *Mathematical Programming*, vol. 191, no. 1, pp. 183–205, 2022, ISSN: 14364646. DOI: 10.1007/s10107-018-1337-6.

-
- [56] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on Stochastic Programming: Modeling and Theory, Third Edition*. 2021, ISBN: 9780898716870. DOI: 10.1137/1.9781611976595.
 - [57] H. G. Beyer and H. P. Schwefel, “Evolution strategie: A comprehensive introduction,” *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002, ISSN: 1567-7818. DOI: 10.1023/A:1015059928466.
 - [58] H. Jalali and I. V. Nieuwenhuyse, “Simulation optimization in inventory replenishment: A classification,” *IIE Transactions (Institute of Industrial Engineers)*, vol. 47, no. 11, pp. 1217–1235, 2015, ISSN: 15458830. DOI: 10.1080/0740817X.2015.1019162.
 - [59] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, “Data-Driven Evolutionary Optimization: An Overview and Case Studies,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2019, ISSN: 1089778X. DOI: 10.1109/TEVC.2018.2869001.
 - [60] W. Chen, W. Hong, H. Zhang, P. Yang, and K. Tang, “Multi-Fidelity Simulation Modeling for Discrete Event Simulation: An Optimization Perspective,” *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1156–1169, 2022, ISSN: 15583783. DOI: 10.1109/TASE.2022.3173296.
 - [61] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, “A study on polynomial regression and Gaussian Process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm,” *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings*, vol. 3, pp. 2832–2839, 2005. DOI: 10.1109/cec.2005.1555050.
 - [62] W. Chen, H. Guo, and K. L. Tsui, “An Adaptive Gaussian Process-Based Search for Stochastically Constrained Optimization via Simulation,” *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1718–1729, 2021, ISSN: 15583783. DOI: 10.1109/TASE.2020.3016489.
 - [63] N. V. Queipo and E. Nava, “A gradient boosting approach with diversity promoting measures for the ensemble of surrogates in engineering,” *Structural and Multidisciplinary Optimization*, vol. 60, no. 4, pp. 1289–1311, 2019, ISSN: 16151488. DOI: 10.1007/s00158-019-02325-4.
 - [64] S. Guo, W. Guo, L. Bian, and Y. B. Guo, “A Deep-Learning-Based Surrogate Model for Thermal Signature Prediction in Laser Metal Deposition,” *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 482–494, 2023, ISSN: 15583783. DOI: 10.1109/TASE.2022.3158204.

- [65] N. Baharom and P. Hamzah, “Inventory Optimization using Simulation Approach,” *Journal of Computing Research and Innovation*, vol. 3, no. 2, pp. 38–47, 2018. doi: 10.24191/jcrinn.v3i2.93.
- [66] E. d. O. Pacheco, S. Cannella, R. Lüders, and A. P. Barbosa-Povoa, “Order-up-to-level policy update procedure for a supply chain subject to market demand uncertainty,” *Computers and Industrial Engineering*, vol. 113, no. March 2018, pp. 347–355, 2017, issn: 03608352. doi: 10.1016/j.cie.2017.09.015.
- [67] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming* (Springer Series in Operations Research and Financial Engineering 8-9). New York, NY: Springer New York, 2011, vol. 831, pp. 701–705, isbn: 978-1-4614-0236-7. doi: 10.1007/978-1-4614-0237-4.
- [68] J. Keutchan, J. Ortmann, and W. Rei, *Problem-driven scenario clustering in stochastic optimization*. Springer Berlin Heidelberg, 2023, vol. 20, pp. 1–33, isbn: 0123456789. doi: 10.1007/s10287-023-00446-2. arXiv: 2106.11717. [Online]. Available: <https://doi.org/10.1007/s10287-023-00446-2>.
- [69] G. Pflug, “Scenario tree generation for multiperiod financial optimization by optimal discretization,” *Mathematical Programming*, vol. 89, no. 2, pp. 251–271, Jan. 2001, issn: 0025-5610. doi: 10.1007/PL00011398. [Online]. Available: <http://link.springer.com/10.1007/PL00011398>.
- [70] X. Chou and E. Messina, “Problem-Driven Scenario Generation for Stochastic Programming Problems: A Survey,” *Algorithms*, vol. 16, no. 10, 2023, issn: 19994893. doi: 10.3390/a16100479.
- [71] A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello, “The sample average approximation method for stochastic discrete optimization,” *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002, issn: 10526234. doi: 10.1137/S1052623499363220.
- [72] H. Heitsch, H. Leövey, and W. Römisch, “Are Quasi-Monte Carlo algorithms efficient for two-stage stochastic programs?” *Computational Optimization and Applications*, vol. 65, no. 3, pp. 567–603, 2016, issn: 15732894. doi: 10.1007/s10589-016-9843-z. arXiv: 1306.3815.
- [73] G. C. Pflug and A. Pichler, “Dynamic generation of scenario trees,” *Computational Optimization and Applications*, vol. 62, no. 3, pp. 641–668, 2015, issn: 15732894. doi: 10.1007/s10589-015-9758-0.

-
- [74] K. Ponomareva, D. Roman, and P. Date, “An algorithm for moment-matching scenario generation with application to financial portfolio optimisation,” *European Journal of Operational Research*, vol. 240, no. 3, pp. 678–687, 2015, ISSN: 03772217. DOI: 10.1016/j.ejor.2014.07.049.
 - [75] N. C. Koutsoukis, D. O. Siagkas, P. S. Georgilakis, and N. D. Hatziaargyriou, “Online Reconfiguration of Active Distribution Networks for Maximum Integration of Distributed Generation,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 437–448, Apr. 2017, ISSN: 1545-5955. DOI: 10.1109/TASE.2016.2628091. [Online]. Available: <http://ieeexplore.ieee.org/document/7776824/>.
 - [76] E. Schubert and P. J. Rousseeuw, “Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms,” *Information Systems*, vol. 101, p. 101 804, 2021, ISSN: 03064379. DOI: 10.1016/j.is.2021.101804. arXiv: 2008.05171.
 - [77] N. Löhndorf, “An empirical analysis of scenario generation methods for stochastic optimization,” *European Journal of Operational Research*, vol. 255, no. 1, pp. 121–132, 2016, ISSN: 03772217. DOI: 10.1016/j.ejor.2016.05.021.
 - [78] X. Jiang, R. Bai, S. W. Wallace, G. Kendall, and D. Landa-Silva, “Soft clustering-based scenario bundling for a progressive hedging heuristic in stochastic service network design,” *Computers and Operations Research*, vol. 128, p. 105 182, 2021, ISSN: 03050548. DOI: 10.1016/j.cor.2020.105182.
 - [79] L. F. Escudero, M. A. Garín, and A. Unzueta, “Cluster Lagrangean decomposition in multistage stochastic optimization,” *Computers and Operations Research*, vol. 67, pp. 48–62, 2016, ISSN: 03050548. DOI: 10.1016/j.cor.2015.09.005.
 - [80] G. C. Calafiore, “Repetitive Scenario Design,” *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1125–1137, 2017, ISSN: 00189286. DOI: 10.1109/TAC.2016.2575859.
 - [81] D. Bertsimas and N. Mundru, “Optimization-Based Scenario Reduction for Data-Driven Two-Stage Stochastic Optimization,” *Operations Research*, 2022, ISSN: 0030-364X. DOI: 10.1287/opre.2022.2265.
 - [82] J. Fairbrother, A. Turner, and S. W. Wallace, “Problem-driven scenario generation: an analytical approach for stochastic programs with tail risk measure,” *Mathematical Programming*, vol. 191, no. 1, pp. 141–182, 2022, ISSN: 14364646. DOI: 10.1007/s10107-019-01451-7. arXiv: 1511.03074.

- [83] K. Ryan, S. Ahmed, S. S. Dey, D. Rajan, A. Musselman, and J.-P. Watson, "Optimization-Driven Scenario Grouping," *INFORMS Journal on Computing*, vol. 32, no. 3, pp. 805–821, 2020, ISSN: 1091-9856. DOI: 10.1287/ijoc.2019.0924.
- [84] J. Dumouchelle, R. Patel, E. B. Khalil, and M. Bodur, "Neur2SP: Neural Two-Stage Stochastic Programming," *arXiv.org*, no. NeurIPS, 2022. arXiv: 2205.12006.
- [85] Y. Wu, W. Song, Z. Cao, and J. Zhang, "Learning Scenario Representation for Solving Two-Stage Stochastic Integer Programs," *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.
- [86] M. Hewitt, J. Ortmann, and W. Rei, "Decision-based scenario clustering for decision-making under uncertainty," *Annals of Operations Research*, pp. 1–25, Jan. 2021, ISSN: 15729338. DOI: 10.1007/s10479-020-03843-x. [Online]. Available: <https://link.springer.com/article/10.1007/s10479-020-03843-x>.
- [87] C. R. Lin, Z. C. Cao, and M. C. Zhou, "Autoencoder-Embedded Iterated Local Search for Energy-Minimized Task Schedules of Human-Cyber-Physical Systems," *IEEE Transactions on Automation Science and Engineering*, vol. PP, pp. 1–11, 2023, ISSN: 15583783. DOI: 10.1109/TASE.2023.3267714.
- [88] I. Suemitsu, H. K. Bhamgara, K. Utsugi, J. Hashizume, and K. Ito, "Fast Simulation-based Order Sequence Optimization Assisted by Pre-trained Bayesian Recurrent Neural Network," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 1–8, 2022. DOI: 10.1109/lra.2022.3185778.
- [89] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, p. 77, Mar. 1952, ISSN: 00221082. DOI: 10.2307/2975974. [Online]. Available: <https://www.jstor.org/stable/2975974?origin=crossref>.
- [90] M. Engles, "Portfolio Optimization: Beyond Markowitz," Master's thesis, Leiden University, 2004, pp. 1–134.
- [91] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015. arXiv: 1412.6980.
- [92] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," *US Naval Research Laboratory Technical Report*, vol. 5510, no. 026, pp. 1–21, 2018. arXiv: 1803.09820. [Online]. Available: <http://arxiv.org/abs/1803.09820>.

-
- [93] L. Ntaimo and S. Sen, "The million-variable "March" for stochastic combinatorial optimization," *Journal of Global Optimization*, vol. 32, no. 3, pp. 385–400, 2005, ISSN: 09255001. DOI: 10.1007/s10898-004-5910-6.
 - [94] S. Ahmed, R. Garcia, N. Kong, *et al.*, *SIPLIB: A Stochastic Integer Programming Test Problem Library*, 2015. [Online]. Available: <https://www2.isye.gatech.edu/%7B~%7Dsahmed/siplib>.
 - [95] A. Kiuchi, H. Wang, Q. Wang, *et al.*, "Bayesian Optimization Algorithm with Agent-based Supply Chain Simulator for Multi-echelon Inventory Management," *IEEE International Conference on Automation Science and Engineering*, vol. 2020-Augus, pp. 418–425, 2020, ISSN: 21618089. DOI: 10.1109/CASE48305.2020.9216792.
 - [96] F. R. Jacobs and R. Chase, *Operations and Supply Chain Management*. McGraw-Hill Irwin, 2014.
 - [97] R. F. Jacobs, W. L. Berry, C. Whybark, and T. E. Vollmann, *Manufacturing Planning and Control for Supply Chain Management*. McGraw-Hill Education, 2011, ISBN: 978-1260108385.
 - [98] S. Noroozi and J. Wikner, "Sales and operations planning in the process industry: A literature review," *International Journal of Production Economics*, vol. 188, no. April 2016, pp. 139–155, 2017, ISSN: 09255273. DOI: 10.1016/j.ijpe.2017.03.006. [Online]. Available: <http://dx.doi.org/10.1016/j.ijpe.2017.03.006>.
 - [99] D. F. Pereira, J. F. Oliveira, and M. A. Carravilla, "Tactical sales and operations planning: A holistic framework and a literature review of decision-making models," *International Journal of Production Economics*, vol. 228, no. February, p. 107 695, 2020, ISSN: 09255273. DOI: 10.1016/j.ijpe.2020.107695. [Online]. Available: <https://doi.org/10.1016/j.ijpe.2020.107695>.
 - [100] N. Brahimi, N. Absi, S. Dauzère-Pérès, and S. Kedad-Sidhoum, "Models and Lagrangian heuristics for a two-level lot-sizing problem with bounded inventory," *OR Spectrum*, vol. 37, no. 4, pp. 983–1006, 2015, ISSN: 14366304. DOI: 10.1007/s00291-015-0404-0.
 - [101] M. Darvish, H. Larrain, and L. C. Coelho, "A dynamic multi-plant lot-sizing and distribution problem," *International Journal of Production Research*, vol. 54, no. 22, pp. 6707–6717, 2016, ISSN: 1366588X. DOI: 10.1080/00207543.2016.1154623.
 - [102] M. Khouja and S. Goyal, "A review of the joint replenishment problem literature: 1989-2005," *European Journal of Operational Research*, vol. 186, no. 1, pp. 1–16, 2008, ISSN: 03772217. DOI: 10.1016/j.ejor.2007.03.007.

- [103] M. K. Salameh, A. A. Yassine, B. Maddah, and L. Ghaddar, "Joint replenishment model with substitution," *Applied Mathematical Modelling*, vol. 38, no. 14, pp. 3662–3671, 2014, ISSN: 0307904X. DOI: 10.1016/j.apm.2013.12.008.
- [104] C.-Y. Lee, S. Çetinkaya, and A. P. M. Wagelmans, "A Dynamic Lot-Sizing Model with Demand Time Windows," *Management Science*, vol. 47, no. 10, pp. 1384–1395, Oct. 2001, ISSN: 0025-1909. DOI: 10.1287/mnsc.47.10.1384. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.47.10.1384>.
- [105] F. F. Boctor, G. Laporte, and J. Renaud, "Models and algorithms for the dynamic-demand joint replenishment problem," *International Journal of Production Research*, vol. 42, no. 13, pp. 2667–2678, Jul. 2004, ISSN: 0020-7543. DOI: 10.1080/00207540410001671660. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/00207540410001671660>.
- [106] P. Robinson, A. Narayanan, and F. Sahin, "Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms," *Omega*, vol. 37, no. 1, pp. 3–15, 2009, ISSN: 03050483. DOI: 10.1016/j.omega.2006.11.004.
- [107] L. d. S. L. Bastos, M. L. Mendes, D. R. d. L. Nunes, A. C. S. Melo, and M. P. Carneiro, "A systematic literature review on the joint replenishment problem solutions: 2006-2015," *Production*, vol. 27, 2017, ISSN: 19805411. DOI: 10.1590/0103-6513.222916.
- [108] A. Federgruen and M. Tzur, "The Joint Replenishment Problem with Time-Varying Costs and Demands: Efficient, Asymptotic and ϵ -Optimal Solutions," *Operations Research*, vol. 42, no. 6, pp. 1067–1086, Dec. 1994, ISSN: 0030-364X. DOI: 10.1287/opre.42.6.1067. [Online]. Available: <http://pubsonline.informs.org/doi/10.1287/opre.42.6.1067>.
- [109] E. P. Robinson, A. Narayanan, and L. L. Gao, "Effective heuristics for the dynamic demand joint replenishment problem," *Journal of the Operational Research Society*, vol. 58, no. 6, pp. 808–815, 2007, ISSN: 14769360. DOI: 10.1057/palgrave.jors.2602197.
- [110] L. L. Gao, N. Altay, and E. P. Robinson, "A comparative study of modeling and solution approaches for the coordinated lot-size problem with dynamic demand," *Mathematical and Computer Modelling*, vol. 47, no. 11-12, pp. 1254–1263, 2008, ISSN: 08957177. DOI: 10.1016/j.mcm.2007.02.035.
- [111] M. A. Bushuev, A. Guiffrida, M. Y. Jaber, and M. Khan, "A review of inventory lot sizing review papers," *Management Research Review*, vol. 38, no. 3, pp. 283–298, Mar. 2015, ISSN: 2040-8269. DOI: 10.1108/MRR-09-2013-0204. [Online]. Available: <https://www.emerald.com/insight/content/doi/10.1108/MRR-09-2013-0204/full/html>.

- [112] G. P. Kiesmüller, “Multi-item inventory control with full truckloads: A comparison of aggregate and individual order triggering,” *European Journal of Operational Research*, vol. 200, no. 1, pp. 54–62, 2010, ISSN: 03772217. DOI: 10.1016/j.ejor.2008.12.008.
- [113] K. Singha, J. Buddhakulsomsiri, and P. Parthanadee, “Mathematical Model of (R,Q) Inventory Policy under Limited Storage Space for Continuous and Periodic Review Policies with Backlog and Lost Sales,” *Mathematical Problems in Engineering*, vol. 2017, no. Dc, 2017, ISSN: 15635147. DOI: 10.1155/2017/4391970.
- [114] J. Noh, J. S. Kim, and S. J. Hwang, “A multi-item replenishment problem with carbon cap-and-trade under uncertainty,” *Sustainability (Switzerland)*, vol. 12, no. 12, pp. 1–15, 2020, ISSN: 20711050. DOI: 10.3390/SU12124877.
- [115] Y. Chen, L. Yang, Y. Jiang, M. I. Wahab, and J. Yang, “Joint replenishment decision considering shortages, partial demand substitution, and defective items,” *Computers and Industrial Engineering*, vol. 127, pp. 420–435, 2019, ISSN: 03608352. DOI: 10.1016/j.cie.2018.10.031. [Online]. Available: <https://doi.org/10.1016/j.cie.2018.10.031>.
- [116] A. Lorenzo-Espejo, J. Muñuzuri, J. Guadix, and A. Escudero-Santana, “A Hybrid Meta-heuristic for the Omnichannel Multiproduct Inventory Replenishment Problem,” *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 17, no. 2, pp. 476–492, 2022. DOI: 10.3390/jtaer17020025.
- [117] R. Liu, S. Wang, Y. Pi, and Q. Qin, “An effective heuristic with evolutionary algorithm for the coordinated capacitated dynamic lot-size and delivery problem,” *Computers and Industrial Engineering*, vol. 153, p. 107051, 2021, ISSN: 03608352. DOI: 10.1016/j.cie.2020.107051. [Online]. Available: <https://doi.org/10.1016/j.cie.2020.107051>.
- [118] K. Nagasawa, T. Irohara, Y. Matoba, and S. Liu, “Applying genetic algorithm for can-order policies in the joint replenishment problem,” *Industrial Engineering and Management Systems*, vol. 14, no. 1, pp. 1–10, 2015, ISSN: 22346473. DOI: 10.7232/iems.2015.14.1.001.
- [119] Y. Chu and F. You, *Simulation-based optimization for multi-echelon inventory systems under uncertainty*. 2015, vol. 2015-Janua, pp. 385–394, ISBN: 9781479974863. DOI: 10.1109/WSC.2014.7019905.
- [120] L. L. Lim, G. Alpan, and B. Penz, “A simulation-optimization approach for sales and operations planning in build-to-order industries with distant sourcing: Focus on the automotive industry,” *Computers and Industrial Engineering*, vol. 112, pp. 469–482, 2017, ISSN: 03608352. DOI: 10.1016/j.cie.2016.12.002.

- [121] C. S. Grewal, S. T. Enns, and P. Rogers, "Dynamic reorder point replenishment strategies for a capacitated supply chain with seasonal demand," *Computers and Industrial Engineering*, vol. 80, pp. 97–110, 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2014.11.009. [Online]. Available: <http://dx.doi.org/10.1016/j.cie.2014.11.009>.
- [122] A. Alrabghi and A. Tiwari, "State of the art in simulation-based optimisation for maintenance systems," *Computers and Industrial Engineering*, vol. 82, pp. 167–182, 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2014.12.022.
- [123] W. Trigueiro de Sousa Junior, J. A. Barra Montevechi, R. de Carvalho Miranda, and A. Teberga Campos, "Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review," *Computers and Industrial Engineering*, vol. 128, no. July 2017, pp. 526–540, 2019, ISSN: 03608352. DOI: 10.1016/j.cie.2018.12.073.
- [124] C. A. Hochmuth, J. Lässig, and S. Thiem, "Simulation-based evolutionary optimization of complex multi-location inventory models," *2010 3rd International Conference on Computer Science and Information Technology*, pp. 703–708, 2010. DOI: 10.1109/ICCSIT.2010.5564022..
- [125] M. F. Yegul, F. S. Erenay, S. Striepe, and M. Yavuz, "Improving configuration of complex production lines via simulation-based optimization," *Computers & Industrial Engineering*, vol. 109, pp. 295–312, Jul. 2017, ISSN: 03608352. DOI: 10.1016/j.cie.2017.04.019. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0360835217301675>.
- [126] S. Olafsson and Jumi Kim, "Simulation optimization," in *Proceedings of the Winter Simulation Conference*, vol. 1, IEEE, 2002, pp. 79–84, ISBN: 0-7803-7614-5. DOI: 10.1109/WSC.2002.1172871.
- [127] G. Figueira and B. Almada-Lobo, "Hybrid simulation-optimization methods: A taxonomy and discussion," *Simulation Modelling Practice and Theory*, vol. 46, pp. 118–134, Aug. 2014, ISSN: 1569190X. DOI: 10.1016/j.simpat.2014.03.007.
- [128] A. Nikolopoulou and M. G. Ierapetritou, "Hybrid simulation based optimization approach for supply chain management," *Computers & Chemical Engineering*, vol. 47, pp. 183–193, Dec. 2012, ISSN: 00981354. DOI: 10.1016/j.compchemeng.2012.06.045.
- [129] M. Hrušovský, E. Demir, W. Jammerneegg, and T. Van Woensel, "Hybrid simulation and optimization approach for green intermodal transportation problem with travel time uncertainty," *Flexible Services and Manufacturing Journal*, vol. 30, no. 3, pp. 486–516, 2018, ISSN: 19366590. DOI: 10.1007/s10696-016-9267-1.

-
- [130] M. de Keizer, R. Haijema, J. M. Bloemhof, and J. G. van der Vorst, “Hybrid optimization and simulation to design a logistics network for distributing perishable products,” *Computers & Industrial Engineering*, vol. 88, pp. 26–38, Oct. 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2015.06.017.
 - [131] M. Zhang, A. Matta, A. Alfieri, and G. Pedrielli, “A simulation-based benders’ cuts generation for the joint workstation, workload and buffer allocation problem,” *IEEE International Conference on Automation Science and Engineering*, vol. 2017-Augus, pp. 1067–1072, 2017, ISSN: 21618089. DOI: 10.1109/COASE.2017.8256247.
 - [132] A. M. Geoffrion, “Generalized Benders decomposition,” *Journal of Optimization Theory and Applications*, vol. 10, no. 4, pp. 237–260, Oct. 1972, ISSN: 0022-3239. DOI: 10.1007/BF00934810.
 - [133] M. Forbes, M. Harris, M. Jansen, F. van der Schoot, and T. Taimre, “Combining Optimization and Simulation Using Logic-Based Benders Decomposition,” *arXiv.org*, Jul. 2021. arXiv: 2107.08390.
 - [134] S. Anily, M. Tzur, and L. A. Wolsey, “Multi-item lot-sizing with joint set-up costs,” *Mathematical Programming*, vol. 119, no. 1, pp. 79–94, 2009, ISSN: 00255610. DOI: 10.1007/s10107-007-0202-9.
 - [135] M. Nomura, S. Watanabe, Y. Akimoto, Y. Ozaki, and M. Onishi, “Warm Starting CMA-ES for Hyperparameter Optimization,” 2020. arXiv: 2012.06932. [Online]. Available: <http://arxiv.org/abs/2012.06932>.
 - [136] R. de Koster, T. Le-Duc, and K. J. Roodbergen, “Design and control of warehouse order picking: A literature review,” *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, 2007, ISSN: 03772217. DOI: 10.1016/j.ejor.2006.07.009.
 - [137] J. Bartholdi and S. Hankman, “Warehouse and Distribution Science Release 0.98.1,” 2019.
 - [138] P. Stibinger, G. Broughton, F. Majer, *et al.*, “Mobile Manipulator for Autonomous Localization, Grasping and Precise Placement of Construction Material in a Semi-Structured Environment,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2595–2602, 2021, ISSN: 23773766. DOI: 10.1109/LRA.2021.3061377. arXiv: 2011.07972.
 - [139] V. Digani, L. Sabattini, and C. Secchi, “A probabilistic Eulerian traffic model for the coordination of multiple AGVs in automatic warehouses,” *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 26–32, 2016, ISSN: 23773766. DOI: 10.1109/LRA.2015.2505646.

- [140] H. Yoshitake, R. Kamoshida, and Y. Nagashima, “New Automated Guided Vehicle System Using Real-Time Holonic Scheduling for Warehouse Picking,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1045–1052, 2019, ISSN: 23773766. DOI: 10.1109/LRA.2019.2894001.
- [141] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, “A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016, ISSN: 23773766. DOI: 10.1109/LRA.2016.2532924. arXiv: 1509.01277.
- [142] S. D. Han, S. W. Feng, and J. Yu, “Toward Fast and Optimal Robotic Pick-And-Place on a Moving Conveyor,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 446–453, 2020, ISSN: 23773766. DOI: 10.1109/LRA.2019.2961605. arXiv: 1912.08009.
- [143] A. R. A. Keshavarz, D. Jaafari, M. Khalaj, and P. Dokouhaki, “A survey of the literature on order-picking systems by combining planning problems,” *Applied Sciences (Switzerland)*, vol. 11, no. 22, 2021, ISSN: 20763417. DOI: 10.3390/app112210641.
- [144] E. M. González-Neira and J. R. Montoya-Torres, “A simheuristic for bi-objective stochastic permutation flow shop scheduling problem,” *Journal of Project Management*, vol. 4, pp. 57–80, 2019, ISSN: 23718366. DOI: 10.5267/j.jpm.2019.1.003.
- [145] N. Rouky, M. N. Abourraja, J. Boukachour, D. Boudebous, A. E. H. Alaoui, and F. El Khoukhi, “Simulation optimization based ant colony algorithm for the uncertain quay crane scheduling problem,” *International Journal of Industrial Engineering Computations*, vol. 10, no. 1, pp. 111–132, 2019, ISSN: 19232934. DOI: 10.5267/j.ijiec.2018.2.002.
- [146] M. S. Umam, M. Mustafid, and S. Suryono, “A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem,” *Journal of King Saud University - Computer and Information Sciences*, 2021, ISSN: 22131248. DOI: 10.1016/j.jksuci.2021.08.025. [Online]. Available: <https://doi.org/10.1016/j.jksuci.2021.08.025>.
- [147] C. Y. Cheng, S. W. Lin, P. Pourhejazy, K. C. Ying, and J. W. Zheng, “Minimizing Total Completion Time in Mixed-Blocking Permutation Flowshops,” *IEEE Access*, vol. 8, no. August, pp. 142 065–142 075, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3014106.
- [148] B. Khurshid, S. Maqsood, M. Omair, B. Sarkar, I. Ahmad, and K. Muhammad, “An Improved Evolution Strategy Hybridization with Simulated Annealing for Permutation Flow Shop Scheduling Problems,” *IEEE Access*, vol. 9, pp. 94 505–94 522, 2021, ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3093336.

-
- [149] C. Oh, J. M. Tomczak, E. Gavves, and M. Welling, “Combinatorial Bayesian Optimization using the Graph Cartesian Product,” *Advances in Neural Information Processing Systems*, vol. 32, Feb. 2019, ISSN: 10495258. arXiv: 1902.00448. [Online]. Available: <http://arxiv.org/abs/1902.00448>.
 - [150] A. Deshwal, S. Belakaria, and J. R. Doppa, “Mercer Features for Efficient Combinatorial Bayesian Optimization,” *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pp. 7210–7218, 2021. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/16886>.
 - [151] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys, “Sequencing and scheduling: Algorithms and complexity,” *Handbooks in Operations Research and Management Science*, vol. 4, no. C, pp. 445–522, 1993, ISSN: 09270507. DOI: 10.1016/S0927-0507(05)80189-6.
 - [152] N. Boysen, K. Stephan, and F. Weidinger, “Manual order consolidation with put walls: the batched order bin sequencing problem,” *EURO Journal on Transportation and Logistics*, vol. 8, no. 2, pp. 169–193, 2019, ISSN: 21924384. DOI: 10.1007/s13676-018-0116-0. [Online]. Available: <http://dx.doi.org/10.1007/s13676-018-0116-0>.
 - [153] A. R. F. Pinto and M. S. Nagano, “An approach for the solution to order batching and sequencing in picking systems,” *Production Engineering*, vol. 13, no. 3-4, pp. 325–341, 2019, ISSN: 18637353. DOI: 10.1007/s11740-019-00904-4.
 - [154] Z. Pitra, L. Bajer, and M. Holena, “Comparing SVM, Gaussian process and random forest surrogate models for the CMA-ES,” *CEUR Workshop Proceedings*, vol. 1422, pp. 186–193, 2015, ISSN: 16130073.
 - [155] G. Briffoteaux, M. Gobert, R. Ragonnet, *et al.*, “Parallel surrogate-assisted optimization: Batched Bayesian Neural Network-assisted GA versus q-EGO,” *Swarm and Evolutionary Computation*, vol. 57, 2020, ISSN: 22106502. DOI: 10.1016/j.swevo.2020.100717.
 - [156] J. Stork, M. Friese, M. Zaefferer, *et al.*, “Open Issues in Surrogate-Assisted Optimization,” *Studies in Computational Intelligence*, vol. 833, pp. 225–244, 2020, ISSN: 1860949X. DOI: 10.1007/978-3-030-18764-4_10.
 - [157] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” *33rd International Conference on Machine Learning, ICML 2016*, pp. 1050–1059, Jun. 2015. arXiv: 1506.02142. [Online]. Available: <http://arxiv.org/abs/1506.02142>.
 - [158] E. H. Frazelle, *World-Class Warehousing and Material Handling, Second Edition*, 2nd ed. New York: McGraw-Hill Education, 2016, ISBN: 0071842829. DOI: 10.1036/9780071842839.

- [159] K. Azadeh, R. De Koster, and D. Roy, “Robotized and automated warehouse systems: Review and recent developments,” *Transportation Science*, vol. 53, no. 4, pp. 917–945, 2019, ISSN: 15265447. DOI: 10.1287/trsc.2018.0873.
- [160] H. Yoshitake and P. Abbeel, “The Impact of Overall Optimization on Warehouse Automation,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 1621–1628, 2023, ISSN: 21530866. DOI: 10.1109/IROS55552.2023.10342333. arXiv: 2308.06036.
- [161] Ç. Cergibozan and A. S. Tasan, “Order batching operations: an overview of classification, solution techniques, and future research,” *Journal of Intelligent Manufacturing*, vol. 30, no. 1, pp. 335–349, 2019, ISSN: 15728145. DOI: 10.1007/s10845-016-1248-4.
- [162] S. Henn, S. Koch, and G. Wäscher, “Order batching in order picking warehouses: A survey of solution approaches,” 2012. DOI: 10.1007/978-1-4471-2274-6_6.
- [163] Y. Zhang, R. Bai, R. Qu, C. Tu, and J. Jin, “A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties,” *European Journal of Operational Research*, vol. 300, no. 2, pp. 418–427, 2022, ISSN: 03772217. DOI: 10.1016/j.ejor.2021.10.032.
- [164] T. Dokeroglu, T. Kucukyilmaz, and E.-G. Talbi, “Hyper-heuristics: A survey and taxonomy,” *Computers & Industrial Engineering*, vol. 187, p. 109 815, Jan. 2024, ISSN: 03608352. DOI: 10.1016/j.cie.2023.109815. [Online]. Available: <https://doi.org/10.1016/j.cie.2023.109815>
<https://linkinghub.elsevier.com/retrieve/pii/S0360835223008392>.
- [165] B. Cals, Y. Zhang, R. Dijkman, and C. van Dorst, “Solving the online batching problem using deep reinforcement learning,” *Computers and Industrial Engineering*, vol. 156, no. February, p. 107 221, 2021, ISSN: 03608352. DOI: 10.1016/j.cie.2021.107221.
- [166] B. Cheng, L. Wang, Q. Tan, and M. Zhou, “A deep reinforcement learning hyper-heuristic to solve order batching problem with mobile robots,” *Applied Intelligence*, vol. 54, no. 9-10, pp. 6865–6887, 2024, ISSN: 15737497. DOI: 10.1007/s10489-024-05532-9.
- [167] Y. Wang, N. Xie, N. Chen, H. Ma, and G. Chen, “Genetic Programming Hyper Heuristic with Elitist Mutation for Integrated Order Batching and Picker Routing Problem,” *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 2, pp. 346–359, 2025, ISSN: 19410026. DOI: 10.1109/TEVC.2025.3532022.
- [168] N. Gademann and S. van de Velde, “Order batching to minimize total travel time in a parallel-aisle warehouse,” *IIE Transactions (Institute of Industrial Engineers)*, vol. 37, no. 1, pp. 63–75, 2005, ISSN: 0740817X. DOI: 10.1080/07408170590516917.

-
- [169] O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A.-L. Ladier, and M. Ogier, “An efficient and general approach for the joint order batching and picker routing problem,” *European Journal of Operational Research*, vol. 285, no. 2, pp. 497–512, Sep. 2020, ISSN: 03772217. DOI: 10.1016/j.ejor.2020.01.059. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0377221720300977>.
- [170] S. Henn, “Algorithms for on-line order batching in an order picking warehouse,” *Computers & Operations Research*, vol. 39, no. 11, pp. 2549–2563, 2012. DOI: 10.1016/j.cor.2011.12.019.
- [171] E. G. Pardo, S. Gil-Borrás, A. Alonso-Ayuso, and A. Duarte, “Order batching problems: Taxonomy and literature review,” *European Journal of Operational Research*, vol. 313, no. 1, pp. 1–24, 2024, ISSN: 03772217. DOI: 10.1016/j.ejor.2023.02.019.
- [172] S. Henn, “Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses,” *Flexible Services and Manufacturing Journal*, vol. 27, no. 1, pp. 86–114, 2015, ISSN: 19366590. DOI: 10.1007/s10696-012-9164-1.
- [173] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, “Evolution Strategies as a Scalable Alternative to Reinforcement Learning,” pp. 1–13, 2017. arXiv: 1703.03864. [Online]. Available: <http://arxiv.org/abs/1703.03864>.
- [174] P. Pagliuca, N. Milano, and S. Nolfi, “Efficacy of Modern Neuro-Evolutionary Strategies for Continuous Control Optimization,” *Frontiers in Robotics and AI*, vol. 7, no. July, pp. 1–13, 2020, ISSN: 22969144. DOI: 10.3389/frobt.2020.00098. arXiv: 1912.05239.

研究業績一覧

原著論文

- [1] **Issei Suemitsu**, Kazuhiro Izui, Takayuki Yamada, Shinji Nishiwaki, Akio Noda, and Tatsuya Nagatani, “Simultaneous optimization of layout and task schedule for robotic cellular manufacturing systems,” *Computers and Industrial Engineering*, vol. 102, pp. 396–407, Dec. 2016, doi: 10.1016/j.cie.2016.05.036. (査読あり)
- [2] **Issei Suemitsu**, Hanoz Kaiwan Bhamgara, Kei Utsugi, Jiro Hashizume, and Kiyoto Ito, “Fast Simulation-based Order Sequence Optimization Assisted by Pre-trained Bayesian Recurrent Neural Network,” *IEEE Robotics and Automation Letters*, vol. 7 (3), pp. 7818–7825, June 2022, doi:10.1109/LRA.2022.3185778. (査読あり)
- [3] **Issei Suemitsu**, Naoko Miyashita, Junko Hosoda, Yoshihito Shimazu, Takahiro Nishikawa, and Kazuhiro Izui, “Integration of sales, inventory, and transportation resource planning by dynamic-demand joint replenishment problem with time-varying costs”, *Computers & Industrial Engineering*, vol. 188, 109922, pp. 1–15, Jan. 2024, <https://doi.org/10.1016/j.cie.2024.109922>. (査読あり)
- [4] **Issei Suemitsu** and Kazuhiro Izui, “Surrogate-Assisted Scenario-Generation Method for Simulation-Based Stochastic Programming Problems”, *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 13161–13174, Mar. 2025, doi: 10.1109/TASE.2025.3551716. (査読あり)
- [5] **末光 一成**, 吉武 宏, 藤 大樹, 泉井 一浩, 「ハイパーヒューリスティクスを用いた荷合せ工程を含む自動化物流倉庫における動的なオーダーバッチングと実行順序の最適化」, *精密工学会誌 92 巻1 号*, 2026 年 1 月. (査読あり)
- [6] **Issei Suemitsu**, Toru Kono, Ryo Wakabayashi, Jun Tsunoda, Yosuke Yamaguchi, and Wenpeng Wei, “From Empirical Analysis to Mixture-of-Experts Modeling: Zero-Shot Cross-Frequency Forecasting in Real-Time Photovoltaic Management”, *Advanced Energy and Sustainability Research*, 2025. (査読あり)

共著論文

- [7] Kazuhiro Izui, Y. Murakumo, **Issei Suemitsu**, Shinji Nishiwaki, Akio Noda, and Tatsuya Nagatani, “Multiobjective layout optimization of robotic cellular manufacturing systems”, *Computers & Industrial Engineering*, vol. 64 (2), pp. 537–544, 2013, <https://doi.org/10.1016/j.cie.2012.12.003>. (査読あり)

- [8] 松本 紀子, 小坂 忠義, 荒 宏視, **末光 一成**, 堀田 哲裕, 尾崎 友哉, 「作業ロス要因分析のための拡張現実技術を用いた物流倉庫内可視化ツールの開発」, *情報処理学会論文誌コンシューマ・デバイス&システム (CDS)*, vol. 8 (2), 74-83, 2017. (査読あり)
- [9] 古家 直樹, 永原 聡士, **末光 一成**, 植木 隆雄, 荒 宏視, 嶋津 泰毅, 「在庫格納期間を加味した逐次処理型の倉庫内格納先推奨技術の開発」, *日本経営工学会論文誌*, 70 (2), 81-93, 2019. (査読あり)
- [10] 古家 直樹, **末光 一成**, 薦田 憲久, 藤原 融, 「格納期間評価による倉庫内の在庫移動方式」, *日本経営工学会論文誌*, 73 (4), 210-221, 2023. (査読あり)
- [11] Yosuke Yamaguchi, **Issei Suemitsu**, Wenpeng Wei, “Citras: Covariate-informed transformer for time series forecasting”, arXiv:2503.24007, 2025. (査読なし)

国際学会発表 (査読あり)

- [12] **Issei Suemitsu**, Hanoz Kaiwan Bhamgara, Kei Utsugi, Jiro Hashizume, and Kiyoto Ito, “Fast Simulation-based Order Sequence Optimization Assisted by Pre-trained Bayesian Recurrent Neural Network,” *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, 2022.
- [13] Irene Erlyn Wina Rachmawan, **Issei Suemitsu**, Matsuba Hiroya, “Survival Analysis-Based Delay Prediction for Production Line with Irregular Events”, *5th European International Conference on Industrial Engineering and Operations Management (IEOM) Rome*, 2022.
- [14] **Issei Suemitsu**, Tohru Kohno, Ryo Wakabayashi, Jun Tsunoda, “Quantitative evaluation method for regional variations in electricity supply-demand balance fluctuation by weather forecast error”, *42nd European Photovoltaic Solar Energy Conference and Exhibition (EU-PVSEC)*, 2024.
- [15] **Issei Suemitsu** and Kazuhiro Izui, “Surrogate-Assisted Scenario-Generation Method for Simulation-Based Stochastic Programming Problems”, *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, 2025.
- [16] Quoc Hung Nguyen, **Issei Suemitsu**, Itoe Akutsu, Daisuke Aimi, Tsuyoshi Oka, “Safety Stock Model Selection Optimization for Budget-Constrained Multi-Item Inventory Management: A Scalable Framework”, *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, 2025.
- [17] **Issei Suemitsu**, Tohru Kohno, Ryo Wakabayashi, Jun Tsunoda, Yosuke Yamaguchi, and Wenpeng Wei, “Empirical Analysis of Zero-Shot Cross-Frequency

Forecasting for Missing Data Prediction in Real-time PV Management”, *43rd European Photovoltaic Solar Energy Conference and Exhibition (EU-PVSEC)*, 2025.

国内学会発表（査読なし）

[18] 末光 一成, 村雲 泰, 泉井 一浩, 西脇 眞二, 野田 哲男, 永谷 達也, 「自律型セル生産ロボットシステムのレイアウト多目的最適化」 *第11回計測自動制御学会システムインテグレーション部門講演会*, 2010.

[19] 末光 一成, 村雲 泰, 泉井 一浩, 西脇 眞二, 野田 哲男, 永谷 達也, 「ロボットによる組立工程のレイアウト最適設計支援」, *2011年度精密工学会春季大会*, 673-674, 2011, <https://doi.org/10.11522/pscjspe.2011S.0.673.0>.

[20] 末光 一成, 泉井 一浩, 西脇 眞二, 野田 哲男, 永谷 達也, 田中 健一, 「ロボットセル生産システムのレイアウト設計法」, *2011年度精密工学会秋季大会*, 5-6, 2011, <https://doi.org/10.11522/pscjspe.2011A.0.5.0>.

[21] Issei Suemitsu, Yutaka Murakumo, Kazuhiro Izui, Shinji Nishiwaki, Akio Noda, Tatsuya Nagatani, Kanichi Tanaka, “A Multiobjective Optimization Technique for Layout Design of Robotic Assembly Systems”, *9th World Congress on Structural and Multidisciplinary Optimization*, 2011.

[22] Issei Suemitsu, Takayuki Yamada, Kazuhiro Izui, Shinji Nishiwaki, Akio Noda, and Tatsuya Nagatani, “A Layout Design Optimization Method for Multi-robot Assembly Systems”, *The Proceedings of Mechanical Engineering Congress, Japan 2012*, 2012, https://doi.org/10.1299/jsmemecj.2012._CO-JP-7-1.

[23] 末光 一成, 山田 崇恭, 泉井 一浩, 西脇 眞二, 野田 哲男, 永谷 達也, 「多目的 GA を用いた自律型ロボットセル生産システムのレイアウト最適設計支援法」, *日本機械学会生産システム部門研究発表講演会 2013*, 2013.

[24] 白 暁博, 山田 崇恭, 泉井 一浩, 西脇 眞二, 末光 一成, 野田 哲男, 永谷 達也, 「生産システムの多目的レイアウト最適化」, *2013年度精密工学会秋季大会*, 749-750, 2013.

[25] 白 暁博, 山田 崇恭, 泉井 一浩, 西脇 眞二, 末光 一成, 野田 哲男, 永谷 達也, 「ロボット組立工程の二段階レイアウト最適設計法」, *日本機械学会第23回設計工学・システム部門講演会*, 2013.

[26] 末光 一成, 永原 聡士, 櫻田 崇治, 嶋津 泰毅, 「倉庫内ピッキング作業を効率化するオーダ割付最適化技術の開発」, *日本経営工学会 2018 年春季大会*, 2018.

[27] 末光 一成, 細田 順子, 嶋津 慶人, 西川 貴裕, 「工場からの補充リードタイムの変動を考慮した物流倉庫の在庫計画決定方式」, *日本経営工学会 2019 年秋季大会*, 2019.

[28] 末光 一成, 細田 順子, 嶋津 慶人, 西川 貴裕, 「時系列在庫シミュレーションに基づく変動リードタイムを考慮した在庫計画決定方式」, *日本バーチャルリアリティ学会 サイバースペースと仮想都市研究委員会 2019 年度 第22 回シンポジウム*, 2019.

[29] 末光 一成, 宮下 直子, 嶋津 慶人, 西川 貴裕, 「物流リソースの時系列変動を考慮した発注パラメータ最適化方式の開発」, *日本経営工学会 2020 年秋季大会*, 2020.

[30] 末光 一成, 宇都木 契, 永田 真斗, 「深層学習による不確実性を考慮した物流ライン性能評価モデルの構築」, *2021 年度精密工学会春季大会学術講演会*, 2021.

[31] 北山 大智, 末光 一成, 泉井 一浩, 西脇 眞二, 「代理モデルを用いた問題駆動型のシナリオ生成法によるトラス構造設計のロバスト最適化」, *日本機械学会 2025 年度年次大会*, 2025.

その他講演

[32] 末光 一成, 「デジタル技術を活かしたロジスティクス革新を実現する研究開発の取り組み」 *インダストリアルデジタルツインサミット 2023*, 2023.

受賞

[33] 「在庫格納期間を加味した逐次処理型の倉庫内格納先推奨技術の開発」 *日本経営工学会論文賞 2020 年度論文賞*

[34] 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Best conference and application paper award finalists (Aug. 2022.)