Int j simul model 23 (2024) 2, 203-214
Original scientific paper

OPTIMAL COMPUTING BUDGET ALLOCATION FOR OPERATIONS OF A ZONE-PICKING SYSTEM

Kim, S.[*]; Kim, H.[*]; Park, C.[*,#]; Jeong, J.[**]; Yang, H.[**] & Kong, S.[**]
[*] Department of Industrial Engineering, Hanyang University, Seoul, 04763, Republic of Korea
[**] CJ Logistics, Seoul, 04514, Republic of Korea
E-mail: parkcj@hanyang.ac.kr ([#] Corresponding author)

Abstract
This paper discusses a case study of a zone-picking system in a distribution centre. In particular, we designed a basic simulation model for analysing the system using Simio and connected the model to the MySQL database using the existing and new steps in Simio. A ranking and selection problem was subsequently formulated to determine the capacity of the system that maximises the expected throughput while satisfying the constraints on: (1) the expected maximum utilisation, (2) expected time-averaged number of boxes in the main conveyor, and (3) expected flow time. To solve the problem under a limited simulation budget, we implemented an optimal computing budget allocation procedure, which may work in the presence of stochastic constraints. The experimental results demonstrate that our approach can more efficiently and effectively determine the capacity of the system than the equal allocation scheme.
(Received in July 2023, accepted in February 2024. This paper was with the authors 4 months for 2 revisions.)

Key Words: Zone-Picking System, Simio, Database, Optimal Computing Budget Allocation

1. INTRODUCTION

Recently, e-commerce markets have grown significantly, and these trends have been expedited by COVID-19. As a result, it is imperative to ensure that the enormous volume of demand in e-commerce markets is handled in a timely fashion. Hence, several logistics companies set up and operate distribution centres. In this situation, the order-picking system may improve the performance of the distribution centre. Order-picking refers to a job in which a worker collects items called stock keeping units (SKUs) from different locations of the distribution centre and puts them in a box according to order information [1]. Generally, 55–65 % of the total operation cost of the distribution centre is the order-picking cost [2, 3]; therefore, efficiently handling order-picking jobs becomes more important [4, 5].

Researchers and practitioners have developed different types of order-picking systems to improve the efficiency and accuracy of order-picking jobs in distribution centres. Designing separate order-picking areas requires various factors that significantly affect costs and service levels [6]. Zone-picking systems partition an entire space into several zones and connect each zone via conveyors that transport the boxes. Among these systems, sequential zone-picking lines [7] have recently attracted considerable attention. Sequential zone-picking lines are beneficial because they are easy to implement, and the downstream sorter can be eliminated [8]. Nevertheless, the line fixes the path of the box; thus, the box must visit every zone in the system [9]. Ho and Lin [10] proposed a zone-picking network system that connected picking zones with conveyor networks. Unlike the sequential zone-picking line, the zone-picking network allows flexible box paths in a limited space and avoids visits to unnecessary zones. In this study, we focus on an order-picking system in which a box may visit zones only when needed, which is similar to the zone-picking network system.

Simulation models are widely used to evaluate the performance of order-picking systems. Petersen [9] evaluated the mail order-picking policy and Li et al. [11] analysed the picking performance of a mobile fulfilment system by developing simulation models in C and C++

programming languages, respectively. Hwang and Cho [12] designed a simulation model for the order-picking system of a supply centre, and Yu and De Koster [13] investigated the effects of controlling the batch size and number of picking zones using simulation models developed using AutoMod software. Yener and Yazgan [14] studied an order-picking problem for the robotic compact bin-storage system and Liu et al. [15] constructed an order allocation and route planning model of unmanned vehicles to realize efficient order-picking using MATLAB. Moreover, Ho and Lin [10] demonstrated the empirical performance of different order selection rules and dispatching rules in a zone-picking network system using Arena software. Although various simulation languages and software have been used to assess the performance of order-picking systems, to the best of our knowledge, Simio has not yet been actively applied. In this study, we assessed the performance of a specific zone-picking system as a case study, using a simulation model developed using Simio.

In the field of simulation, ranking and selection procedures have been developed to determine the best system design among a finite (usually less than 1000) number of alternatives when performance measures can only be observed via stochastic simulation. Scholars have recently focused on developing ranking and selection procedures in the presence of stochastic constraints. Such ranking and selection procedures are primarily categorised into two types: fully sequential indifference zone and optimal computing budget allocation (OCBA) procedures. Fully sequential indifference zone procedures are designed to terminate when they guarantee the pre-specified probability of correctly selecting the best (or nearly best) system design that satisfies the stochastic constraint(s) (refer to [16-19] for further details of fully sequential indifference zone procedures). However, OCBA procedures allocate a given simulation budget to maximise the probability of correctly selecting the best system design in the presence of stochastic constraints (refer to [20-22] for further details of OCBA procedures). As our objective is to determine the best system design (i.e., the most effective capacity level of the system) when the simulation budget is restricted, we focus on an OCBA procedure rather than a fully sequential indifference zone procedure.

The study that is most closely related to our study is that of Ho and Lin [10], who suggested a zone-picking network system using conveyors and tested different order selection rules and box dispatching rules through a simulation model with a maximum order size of 250 orders. Compared to the work in [10], our original contribution is twofold. We (1) implemented a simulation model using Simio connected with a database, and (2) formulated and solved a decision problem to determine the capacity of the system. Specifically, we designed a simulation model along with a real zone-picking system currently operating in a distribution centre located in South Korea. We then connected the model to the MySQL database through steps in Simio and stored procedures in the MySQL database. As a result, we store the real order data (which possibly includes several million orders) in the database, and subsequently, such data can be properly searched and calculated for operating logic within the model. In addition, we formulated a problem for determining the capacity of the system as a ranking and selection problem in the presence of stochastic constraints, and then applied a version of the OCBA procedure, namely OCBA-CO, which may work with stochastic constraints. We believe that our approach is applicable to improve computational efficiency for designing and analysing various types of zone-picking systems operated by logistics companies.

The remainder of this paper is organized as follows. Section 2 provides a brief description of the zone-picking system and the simulation model to assess the performance of the system. In Section 3, we formulate the problem of determining the capacity of the system as a ranking and selection problem with stochastic constraints and introduce the OCBA-CO procedure to solve our problem. Section 4 presents the experimental results and compares the OCBA-CO procedure with the equal allocation scheme, and concluding remarks are presented in Section 5.

2. SIMULATION MODELLING

In this section, we first describe the zone-picking system in a distribution centre. Subsequently, we provide the key properties of the basic simulation model, which was developed using Simio software. Furthermore, we explain the connection between the Simio model and the MySQL database.

2.1 System description

We consider our target system, called the zone-picking system. Fig. 1 presents an example of the system structure. The entire system was partitioned into several zones. The zones were connected by a main conveyor, which is a large unidirectional loop responsible for transporting the boxes. The main conveyor is shown as a thick solid black line in Fig. 1. The box may remain in the system by moving through the main conveyor if an order assigned to the box must be handled further. Otherwise (i.e., if the order assigned to the box is completely handled), the box exits the system through another conveyor connected to the main conveyor after the last zone (e.g., zone 6 in Fig. 1.).

Each zone mainly consists of racks, a sub-conveyor, and a worker. Each rack is filled with SKUs, and the types of SKUs at each location in the rack may change daily. The sub-conveyor consists of three parts: a queue conveyor, working conveyor, and work-end conveyor, which mainly control the locations of the boxes in each zone. When a box enters a zone from the main conveyor through a diverter, it moves or waits in the queue conveyor. The queue conveyor is then connected to the working conveyor, and the box may not move on the working conveyor until the worker completes the picking jobs for all SKUs in the zone in accordance with the order information. Subsequently, the box moves on a work-end conveyor. If necessary, the box can be moved directly to the next zone. Otherwise, the box re-enters the main conveyor through a diverter only when there is no interference. Each part of the sub-conveyor contains a limited number of boxes.

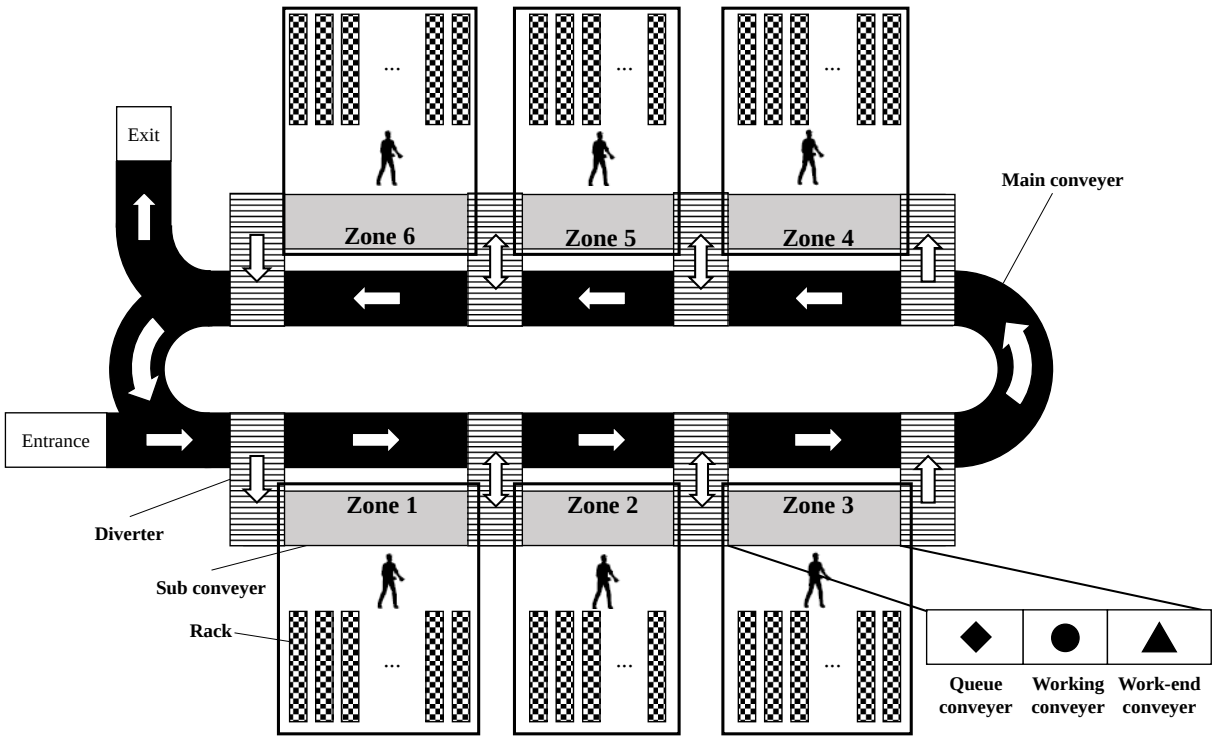

Figure 1: A zone-picking system.

Table I: Example of the order information.

Order ID	SKU ID	xSKU	Quantity	Station	xLOC	Row	Column
141020210207018	150567	9	1	3	117	1	6
141020210207018	190582	10	1	4	147	3	1
141020210207025	190582	10	1	4	147	3	1
141020210207026	150567	9	3	3	117	1	6
141020210207028	190582	10	1	4	147	3	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Whenever a customer requests an order with a single item or multiple items, an order identification number (order ID) is created corresponding to each order. An order consists of one or more order lines, each of which contains detailed information corresponding to each item (i.e., SKU) in that order. A worker subsequently checks the order ID assigned to the box to obtain detailed information regarding the order in the system. Table I lists some order data. Each row represents the information regarding each order line. The first column is the order ID, and the second column is the SKU ID that specifies the type of SKU. The fourth column represents the number of SKU required for the order. The remaining columns indicate the locations of the SKU. It should be noted that the number of rows (i.e., order lines) in the order data for each day may be approximately tens or hundreds of thousands.

Before a simulation runs, one may have a list of orders which must be handled on that day. Specifically, one may first need to check whether a new box can enter the system, because the system has a limit on capacity, and the minimum distance between two boxes consecutively entering the system to prevent interference. If a box can enter the system, an order ID is assigned to the box by attaching an RFID chip or barcode sticker. A box with its own order ID enters the system through the main conveyor and visits picking zones, where a worker needs to pick up SKUs from a rack according to the order information and put them into the box. If the order is completed, the box leaves the system. Otherwise, the box continues to move through the main conveyor until it contains all SKUs in the corresponding order. If all orders for each day are handled, a simulation replication ends.

2.2 Simulation model

The zone-picking system for our case study included six picking zones, and the system boundary of the simulation model was designated from the entrance to the exit of each box. Fig. 2 shows a 2-dimensional representation of the simulation model.

We designed five main logics of the simulation model along with detailed operation information in a real system and implemented them as processes in Simio. In Fig. 2, circled numbers from ① to ⑤ represent locations where main logics are executed. When a box arrives at location ①, an order ID is assigned as the state of the box entity. At location ②, the system checks whether the box can enter a zone using the entrance logic, as shown in Fig. 3 a. Note that the system determines whether a box is required to visit a zone by checking the order information and whether the zone is not full. After the box enters the zone through a diverter and arrives at location ③, the system executes the picking logic shown in Fig. 3 b. According to the picking logic, a worker needs to perform a picking job for each SKU in the order. After completing all picking jobs, the box moves to location ④, and a box may leave the zone by the discharge logic in Fig. 4 a. In the discharge logic, the system checks whether the box must move to an adjacent (i.e., the next) zone. If moving to an adjacent picking zone is not necessary, the box is joined to the main conveyor after considering interference with other boxes. Otherwise, the box is directly transferred to the adjacent picking zone.

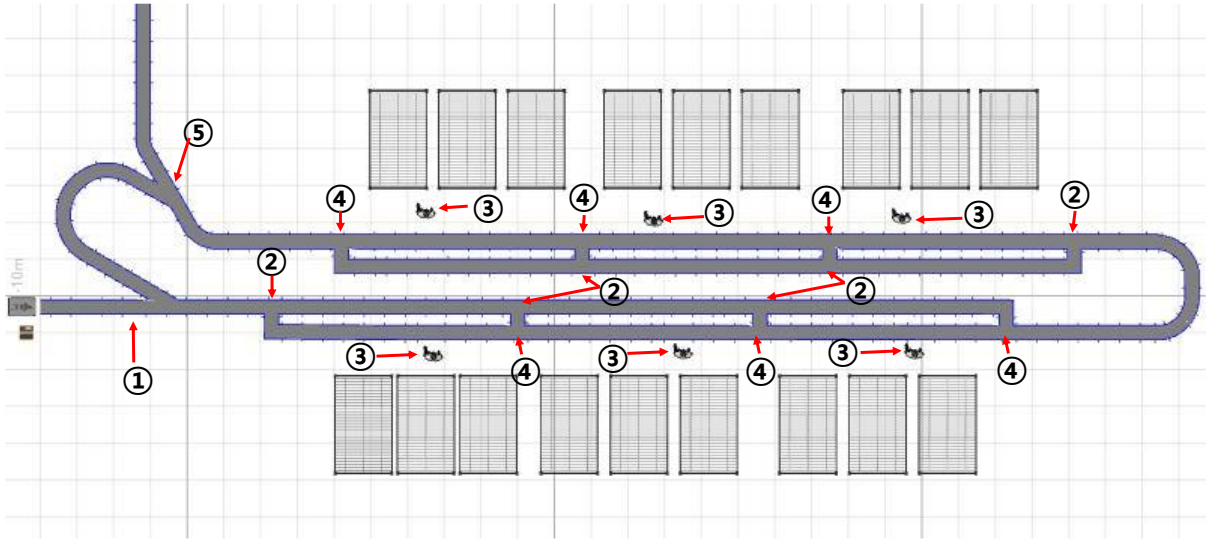

Figure 2: A basic simulation model employed for the case study.

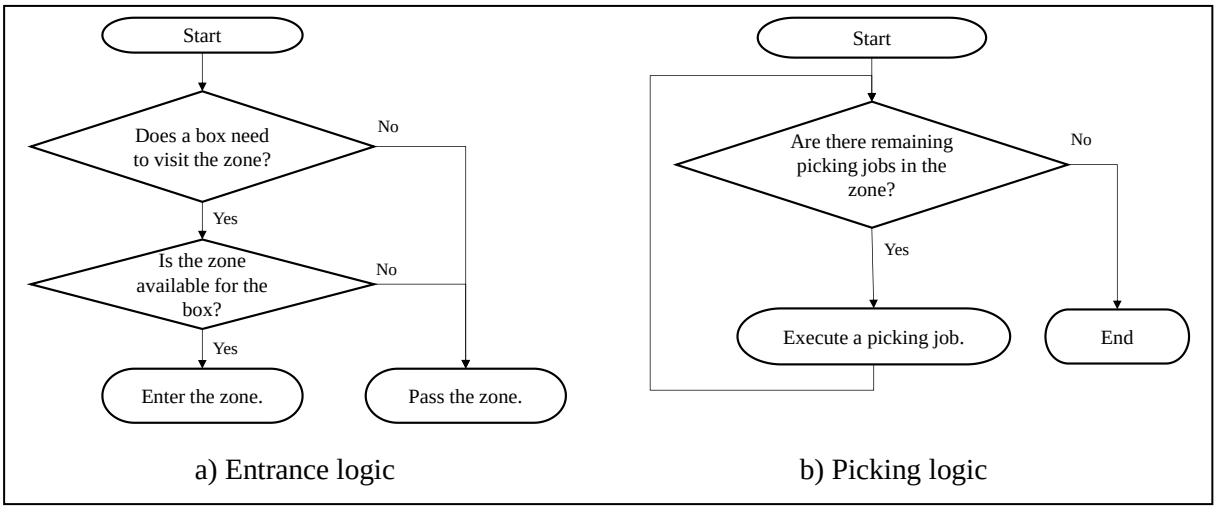

Figure 3: Entrance and picking logics.

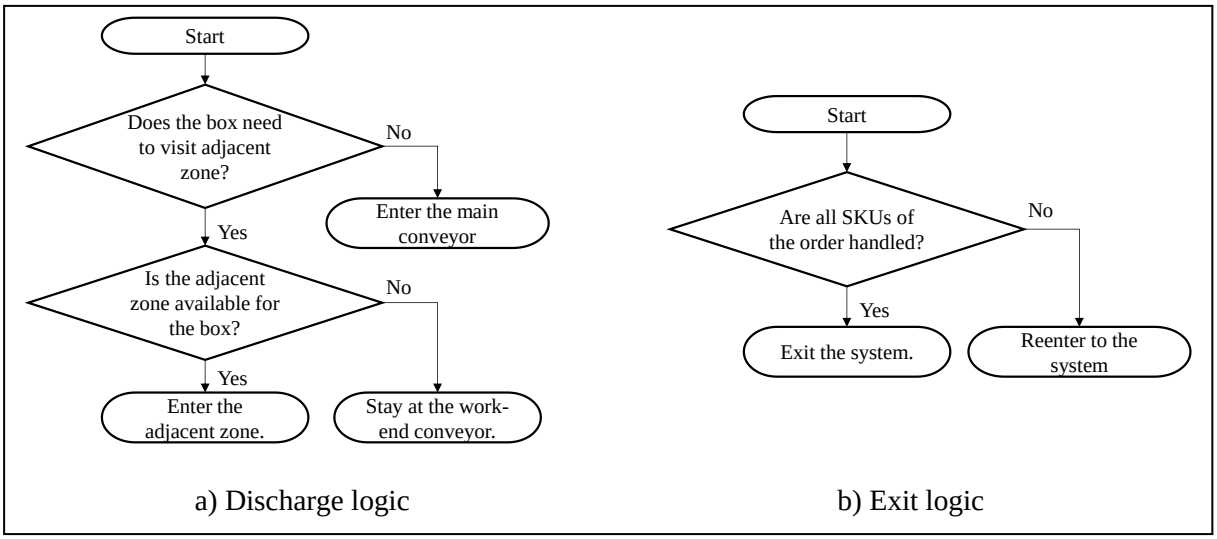

Figure 4: Discharge and exit logics.

When the box arrives at location ⑤, the exit logic shown in Fig. 4 b is executed. The exit logic first checks whether the order assigned to the box is handled completely. If the order must be handled further with additional SKUs, the logic enforces the box to re-enter the main conveyor. Otherwise, the box exits the system.

In addition, we connected the Simio model to a MySQL database to handle the order data of our case study properly. Note that the given order data are too large (millions of rows) to be directly stored in a Simio model; thus, the model needs to be connected to a database. Simio software provides basic steps, such as DBExecute, DBRead, and DBWrite, for dealing with values in the database during model runs. For our model, we first developed stored procedures to preprocess and calculate values in the database and then executed the stored procedures using the DBExecute step, if necessary. In addition, we designed processes using the DBRead and DBWrite steps to search, read, and write values in the database. Nevertheless, the DBRead step returns only the top of the multiple rows that match the condition that the user may want to check. For example, if one attempts to search a row with order ID 141020210207018 in Table I, the first row is returned, but the second row is not recognized using the DBRead step. To overcome this, we implemented new steps called DBRead_rows and DBRead_countrow, which were implemented in C# and added to the processes in Simio. The DBRead_rows and DBRead_countrow steps return all possible rows (not just a single row) and the number of rows that satisfy the search conditions along with the specific columns. We also implemented another step called DBRead_min, which enabled the model to search for the row with the minimum value of a certain column. Therefore, one may deal with order data in customized ways when applying these steps to the case study.

3. OPTIMISATION MODEL

In this section, we formulate our problem in the context of the ranking and selection of the model and then explain the OCBA procedure as a solution approach. Note that the detail descriptions of notations used in this section are presented in Table II.

3.1 Problem formulation

System design (i.e., the decision variable) is distinguished by the total capacity of the system, denoted by i which is the maximum number of boxes (i.e., orders) that the system can contain. In this study, the total capacity was increased incrementally by one within a certain range.

Let X_{0i} be the throughput of the system, X_{1i} be the maximum utilisation for all workers, X_{2i} be the time-averaged number of boxes on the main conveyor, and X_{3i} be the flow time of each box for a system with capacity i during a day. Note that X_{0i} is calculated by dividing the number of orders handled daily by the total processing time, and X_{1i} is the highest daily utilisation for all workers during a day. Because the system includes uncertainty in the order information and picking times of each worker, the optimisation model is formulated with the expected values of each performance measure, denoted by J_{0i}, J_{1i}, J_{2i}, and J_{3i}, respectively. In addition, c_1, c_2, and c_3 are the constraint thresholds that restrict the values of J_{1i}, J_{2i}, and J_{3i}.

The optimisation problem is provided as follows:

$$\max_i \; J_{0i} = E[X_{0i}] \tag{1}$$

$$\text{s.t.} \;\; J_{hi} = E[X_{hi}] \le c_h, \;\; h = 1, 2, 3. \tag{2}$$

Table II: Notions and definitions.

Notation	Definition
i	index of the system design distinguished by the total capacity of the system
h	performance measure index ($h = 0, 1, 2, 3$)
X_{0i}	throughput (the number of boxes per hour) of the system with system design i
X_{1i}	maximum utilisation for all workers with system design i
X_{2i}	time-averaged number of boxes in the main conveyor with system design i
X_{3i}	flow time of each box for system design i per box
c_h	constraint threshold that restricts the value of J_{hi} ($h = 1, 2, 3$)
b	index of the best system design
q_i	most critical performance measure determining feasibility
Θ_O	set of system designs where optimality is more dominant than feasibility,
Θ_F	set of system designs where feasibility is more critical than optimality
j	index of the simulation observation
X_{hij}	j^{th} observation of performance measure h for system design i ($h = 0, 1, 2, 3$)
N_i	simulation budget of system design i
n_0	initial simulation budget for each system design
T	total simulation budget for all system designs
J_{hi}	expected value of X_{hi} ($h = 0, 1, 2, 3$)
$\bar{J}_{hi}$	sample mean of the X_{hij} ($h = 0, 1, 2, 3$)
s_{hi}^2	sample variance of the $X_{hij}(h = 0, 1, 2, 3)$
α_i	simulation budget allocation weight for system design i

3.2 Solution approach

To solve Eqs. (1) and (2) efficiently, we introduced a version of the OCBA procedure [20]. The OCBA procedure is designed in the presence of stochastic constraints, namely OCBA-CO, and approximates the probability of correct selection (PCS). We assume that the index h = 0, 1, 2, and 3 for system measures. Then, we denote b as the index of the best system design. The PCS value is given by Eq. (3), where $\bar{J}_{hi}$ represents the sample mean of X_{hij}.

$$PCS = P\left\{\cap_{h=1}^{3}(\bar{J}_{hb} \le c_h) \ \cap_{\forall i \neq b}\left\{\left[\cap_{h=1}^{3}(\bar{J}_{hi} \le c_h) \cap (\bar{J}_{0b} < \bar{J}_{0i})\right]^c\right\}\right\} \tag{3}$$

The PCS value cannot be transformed into a closed form, but it can be approximated using the Bonferroni inequality. We denote α_i as the simulation budget allocation weight for system design i, and then the Approximated PCS (APCS) that satisfies $PCS \geq APCS$ is calculated using Eq. (4).

$$APCS = \sum_{h=1}^{3} P(\bar{J}_{hb} \le c_h) - \sum_{\forall i \neq b}[\min\,[\min_{h \neq 0} P(\bar{J}_{hi} \le c_h),\ P(\bar{J}_{0b} < \bar{J}_{0i})]] + (1 - 3) \tag{4}$$

Let q_i be the most critical performance measure for system i to determine the feasibility which is defined in Eq. (5). Subsequently, non-best system designs are categorised in one of the two mutually exclusive and collectively exhaustive sets, denoted by Θ_O and Θ_F shown in Eqs. (6) and (7), respectively. When maximising APCS, Θ_O represents the set of designs where optimality is more dominant than feasibility, while Θ_F denotes the set of designs where feasibility is more critical than optimality.

$$q_i \equiv \arg\min_{h \neq 0} P(\bar{J}_{hi} \le c_h) \tag{5}$$

$$\Theta_O \equiv \left\{i \middle| i \neq b,\ P\left(\bar{J}_{q_i i} \le c_{q_i}\right) \geq P(\bar{J}_{0b} < \bar{J}_{0i})\right\} \tag{6}$$

$$\Theta_F \equiv \{i | i \neq b,\ P(\bar{J}_{q_i i} \leq c_{q_i}) < P(\bar{J}_{0b} < \bar{J}_{0i})\} \tag{7}$$

OCBA-CO is designed to calculate the budget weights (i.e., the number of required simulation replications) of each system design that maximises the APCS. Its formulation is given by Eqs. (8) and (9), respectively. Note that the problem solution satisfying the Karush-Kuhn-Tucker condition is optimal because of the concavity of Eq. (8) [20].

$$\max_{\alpha_i} \left[\sum_{h=1}^{3} P(\bar{J}_{hb} \leq c_h) - \sum_{i \in \Theta_F} P(\bar{J}_{q_i i} \leq c_{q_i}) - \sum_{i \in \Theta_O} P(\bar{J}_{0b} < \bar{J}_{0i}) + (1-3)\right] \tag{8}$$

$$\text{s.t.} \quad \sum_{\forall i} \alpha_i = 1 \tag{9}$$

OCBA-CO Procedure

Step 0. Set initial simulation budget n_0 and total budget T.

Step 1. Obtain n_0 observations and calculate the sample mean, and sample variance:

$$\bar{J}_{hi} = \frac{\sum_{j=1}^{n_0} X_{hij}}{n_0}, s_{hi}^2 = \frac{\sum_{j=1}^{n_0} (X_{hij} - \bar{J}_{hi})^2}{n_0 - 1} \text{ for each } i \text{ and } h.$$

Step 2. Select the sample-best design such that

$$\hat{b} = \arg\max_i \bar{J}_{0i} \text{ s.t. } \bar{J}_{hi} \leq c_h,\ \forall h \neq 0.$$

Step 3. Separate non-best design into two sets where $\hat{q}_i = \arg\min_{h \neq 0} \frac{-(\bar{J}_{hi} - c_h)}{s_{hi}}$ for each i:

$$\hat{\Theta}_O = \left\{ i \middle| i \neq \hat{b},\ \frac{(\bar{J}_{\hat{q}_i i} - c_{\hat{q}_i})}{s_{\hat{q}_i i}} \leq \frac{(\bar{J}_{0\hat{b}} - \bar{J}_{0i})}{s_{0i}} \right\} \& \hat{\Theta}_F = \left\{ i \middle| i \neq \hat{b},\ \frac{(\bar{J}_{\hat{q}_i i} - c_{\hat{q}_i})}{s_{\hat{q}_i i}} > \frac{(\bar{J}_{0\hat{b}} - \bar{J}_{0i})}{s_{0i}} \right\}.$$

Step 4. Calculate noise-to-signal ratios:

$$\hat{\eta}_i = \frac{s_{\hat{q}_i i}}{(\bar{J}_{\hat{q}_i i} - c_{\hat{q}_i i})} \text{ for } i \in \hat{\Theta}_F,\ \hat{\eta}_i = \frac{s_{0i}}{(\bar{J}_{0\hat{b}} - \bar{J}_{0i})} \text{ for } i \in \hat{\Theta}_O, \text{ and } \hat{\eta}_{\hat{b}} = \frac{s_{\hat{q}_{\hat{b}} \hat{b}}}{\left(\bar{J}_{\hat{q}_{\hat{b}} \hat{b}} - \hat{c}_{\hat{q}_{\hat{b}}}\right)}.$$

Step 5. Compute budget allocation weights for each design: obtain weights with $\sum_{\forall i} \alpha_i = 1$,

$$\frac{\alpha_i}{\alpha_j} = \left(\frac{\hat{\eta}_i}{\hat{\eta}_j}\right)^2 \text{ for } i \neq j \neq \hat{b},\ \alpha_{\hat{b}} = \max(\alpha_{O\hat{b}},\ \alpha_{F\hat{b}}),$$

$$\text{where } \alpha_{O\hat{b}} = s_{0\hat{b}} \sqrt{\sum_{i \in \hat{\Theta}_O} (\alpha_i / s_{0i})^2}, \frac{\alpha_{F\hat{b}}}{\alpha_i} = \left(\frac{\hat{\eta}_{\hat{b}}}{\hat{\eta}_i}\right)^2,\ \forall i \neq \hat{b}.$$

Step 6. Derive allocation N_i and extra simulation budget Δ_i:

$$N_i = T\alpha_i \ \& \ \Delta_i = \begin{cases} n_0 & \text{if } N_i \leq n_0 \\ \lfloor N_i + 0.5 \rfloor - n_0 & \text{if } N_i > n_0 \end{cases} \text{ for each } i.$$

Figure 5: Overall description of OCBA-CO.

The steps of OCBA-CO are shown in Fig. 5. Let j be the replication number of the simulation. Then, X_{hij} is an observation of system performance measure h in design i from the the j^{th} simulation replication, and N_i is the simulation budget of each design i. Note that the noise-to-signal ratio in Step 4 is the key variable determining the allocation weights theoretically validated in [20]. The noise-to-signal ratio reflects the possibility of an incorrect decision being made. A high noise-to-signal ratio value indicates a high probability that a non-optimal design with respect to optimality or feasibility will be selected as the best. The estimated allocation weights denoted by α_i are calculated based on the noise-to-signal ratio to determine the additional number of replications for each design.

4. EXPERIMENTAL RESULTS

In this section, we describe the experimental settings of our case study, based on a zone-picking system in a real distribution centre in South Korea. In addition, we present the results of applying the OCBA-CO procedure and an equal allocation scheme to determine the capacity of the zone-picking system.

4.1 Experimental settings

Before any simulation runs, each SKU is stored in a single location designated via preprocessing. There were six zones in the system, and the number of SKU was assumed to be sufficient. The size of the boxes is unified with width of 425 mm, length of 370 mm, and height of 330 mm. The boxes may enter the system by considering the capacity of the system and interference among boxes (e.g., two boxes can consecutively enter the system with an interval of at least 4.5 sec to prevent interference, even when the system is not full). In a zone, each sub-conveyor can contain seven boxes (at most four boxes in the queue conveyor and two boxes in the work-end conveyor). A box can enter or re-enter the main conveyor only when the minimum distance between boxes is maintained as at least 1.4 m. The speed of the main conveyor is set to 30 m/min, and the width of the conveyor is set to 560 mm.

As one of the main input data, the list of real-order information for 90 days is stored in the MySQL database and used as input data (i.e., each row is searched and imported via Simio during the model is running). Note that the average number of orders for a day is 2478. For picking times of each worker, we determined the distributions based on real data and then used a random variate generation scheme according to the empirical distributions [23]. In addition, we empirically verified that the different order selection rules in [10] performed similarly in this system, and the random selection rule performed the best. Therefore, we used random selection as the order selection rule.

In accordance with the opinions of field engineers, we set $i = 20, 21, \ldots, 30$ (i.e., $k = 11$) and the constraint thresholds as $c_1 = 0.6$, $c_2 = 27$ and $c_3 = 3.5$. A single simulation replication was terminated when the order list for a single day was completely handled. Therefore, a simulation replication using a daily order list results in a single observation of X_{hij} for $h = 0, 1, 2$, and 3. Finally, we set the initial budget for each system design to 10 (i.e., $n_0 = 10$) and the total budget to 90 (i.e., $T = 90$) due to the available order list.

4.2 Result

Initially, we simulated 10 replications for each system design (i.e., each i), obtained 10 observations ($n_0 = 10$) for each performance measure and system design, and then reported the sample means and variances of each performance measure, $\bar{J}_{hi}$ and s_{hi}^2 for $h = 0, 1, 2$, and 3, and for $i = 20, 21, \ldots, 30$ in Table III. It was observed that (i) all the measures tended to increase as the capacity increases, (ii) the initial sample-best system design was $\hat{b} = 27$, and (iii) the most critical constraint of the system design was $\hat{q}_{27} = 1$ (the expected maximum utilisation). Since we had only 10 replications for each system design, if the total capacity of the system is set to 27 which is the current sample-best system, then one may expect that the system can handle 605 boxes per hour, the maximum utilisation of 0.596, the time-averaged number of boxes in the main conveyer or 17.3, and the flow time of 2.578 min on average. Note that the variance of the throughput is 1078 and thus the average throughput may change if the number of simulation replications increases.

Table III: Sample means and variances of performance measure.

Design i		20	21	22	23	24	25
X_{0i}	$\bar{J}_{0i}$	482.705	502.320	519.742	538.777	556.195	574.013
	s_{0i}^2	688.775	711.014	820.595	1087.043	1340.738	1278.195
X_{1i}	$\bar{J}_{1i}$	0.485	0.493	0.505	0.532	0.551	0.563
	s_{1i}^2	0.004	0.003	0.002	0.003	0.005	0.004
X_{2i}	$\bar{J}_{2i}$	13.302	13.92	14.5	15.05	15.602	16.229
	s_{2i}^2	0.596	0.577	0.561	0.763	0.871	0.81
X_{3i}	$\bar{J}_{3i}$	2.425	2.442	2.47	2.488	2.512	2.531
	s_{3i}^2	0.011	0.01	0.012	0.015	0.022	0.021
Design i		26	**27**	28	29	30	
X_{0i}	$\bar{J}_{0i}$	583.234	**604.993**	617.994	632.715	644.616	
	s_{0i}^2	1681.412	**1077.591**	1648.788	1459.431	1590.645	
X_{1i}	$\bar{J}_{1i}$	0.576	**0.596**	0.609	0.629	0.639	
	s_{1i}^2	0.003	**0.004**	0.003	0.004	0.003	
X_{2i}	$\bar{J}_{2i}$	16.623	**17.299**	17.749	18.383	18.86	
	s_{2i}^2	1.17	**0.817**	1.294	0.939	0.95	
X_{3i}	$\bar{J}_{3i}$	2.575	**2.578**	2.612	2.632	2.672	
	s_{3i}^2	0.016	**0.011**	0.016	0.014	0.014	

Table IV: Allocation of weights and additional simulation replications.

Design i	20	21	22	23	24	25	26	**27**	**28**	29	30
α_i	0.0002	0.0003	0.0004	0.0010	0.0022	0.0053	0.0141	**0.7867**	**0.1634**	0.0179	0.0086
N_i	0.0180	0.0270	0.0360	0.0900	0.198	0.477	1.269	**70.803**	**14.706**	1.611	0.774
Δ_i	0	0	0	0	0	0	0	**61**	**5**	0	0

Table V: Comparison of the equal allocation and the OCBA-CO procedure.

	Equal Allocation			**OCBA-CO**		
Design i	26	**27**	28	**26**	27	28
$\bar{J}_{0i}$	583.330	**603.190**	616.872	**583.234**	562.727	618.019
$\bar{J}_{1i}$	0.576	**0.594**	0.612	**0.576**	0.641	0.605

After the OCBA-CO procedure is applied, as shown in Fig. 5, we calculate the number of additional simulation replications, as listed in Table IV. Only system designs 27 and 28 were allowed to have additional replications, but the other system designs did not require any additional simulation. This is reasonable because checking the feasibility of the sample-best system and a barely infeasible system design whose objective values are higher than the sample-best design while their feasibility may change owing to estimation errors.

Table V presents the results after obtaining additional observations, along with the equal allocation scheme and the OCBA-CO procedure. Note that the equal allocation scheme assigns six additional replications to all designs, whereas OCBA-CO assigns 61 replications for *i* = 27,

five replications for $i = 28$, and zero replication for the remaining designs. As a result, the equal allocation scheme determines that the system design $i = 27$ is still the best, but OCBA-CO decides that the system design $i = 26$ is the best, rather than $i = 27$ because the feasibility of the system $i = 27$ becomes clearer. Therefore, one may expect that the system may handle 583 boxes per hour with the maximum utilisation of 0.576 on average when the system design (i.e., the capacity) is set to $i = 26$ which is the new sample-best design. Nevertheless, the system can handle only 562 boxes per hour with the maximum utilisation of 0.641 (which is clearly infeasible) on average when the system design is set to $i = 27$. Therefore, OCBA-CO efficiently provides better results than the equal allocation in this manner.

5. CONCLUSION

In this paper, we considered a case study of a zone-picking system in a real distribution centre in South Korea. We first designed a simulation model using Simio and then connected the model to a MySQL database to handle large-sized order data. Subsequently, we formulated and solved a ranking and selection problem in the presence of stochastic constraints to determine the proper capacity of the system. We employed the OCBA-CO procedure to determine the capacity of the system under a restricted simulation budget.

The possible uses of the findings in this study are as follows: (1) the performance of the zone-picking system can be successfully assessed via a model developed using Simio and the MySQL database; (2) the steps and stored procedures newly developed and applied in Simio enable users to flexibly search and calculate the values of millions of rows in a database; and (3) the OCBA-CO procedure properly assigns the total simulation budget for the simulation model, which is practically useful, especially when one needs to make a decision according to the simulation model in a limited time or restricted computing budget. One may apply these findings to designing and implementing a whole distribution centre that includes multiple zone-picking systems in different sizes and structures and then calculate the long run average total profit to provide managerial implications. This work is ongoing.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIP) (Nos. 2019R1F1A1061256 and 2022R1F1A1063147).

REFERENCES

[1] Park, B. C. (2012). Order picking: issues, systems and models, Manzini, R. (Ed.), *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*, Springer-Verlag, London, 1-30

[2] Frazelle, E. H. (1996). *World-Class Warehousing*, Logistics Resources International Inc., Atlanta

[3] Coyle, J. J.; Bardi, E. J.; Langley Jr., C. J. (2003). *The Management of Business Logistics: A Supply Chain Perspective*, 7th edition, South-Western/Thomson Learning, Mason

[4] De Koster, R.; Le-Duc, T.; Roodbergen, K. J. (2007). Design and control of warehouse order picking: a literature review, *European Journal of Operational Research*, Vol. 182, No. 2, 481-501, doi:10.1016/j.ejor.2006.07.009

[5] Eisenstein, D. D. (2008). Analysis and optimal design of discrete order picking technologies along a line, *Naval Research Logistics*, Vol. 55, No. 4, 350-362, doi:10.1002/nav.20289

[6] Đurđević, D. B.; Miljuš, M. D. (2013). The procedure proposal for order pick area design, *Technical Gazette*, Vol. 20, No. 1, 85-91

[7] Frazelle, E. H.; Apple, J. M. (1994). Warehouse operations, Tompkins, J. A.; Harmelink, D. A. (Eds.), *The Distribution Management Handbook*, McGraw-Hill, New York, 22.1-22.36

[8] Parikh, P. J.; Meller, R. D. (2008). Selecting between batch and zone order picking strategies in a distribution center, *Transportation Research Part E: Logistics and Transportation Review*, Vol. 44, No. 5, 696-719, doi:10.1016/j.tre.2007.03.002
[9] Petersen, C. G. (2000). An evaluation of order picking policies for mail order companies, *Production and Operations Management*, Vol. 9, No. 4, 319-335, doi:10.1111/j.1937-5956.2000.tb00461.x
[10] Ho, Y.-C.; Lin, J.-W. (2017). Improving order-picking performance by converting a sequential zone-picking line into a zone-picking network, *Computers & Industrial Engineering*, Vol. 113, 241-255, doi:10.1016/j.cie.2017.09.014
[11] Li, W.; Miao, L.; Yang, P. (2021). Simulation analysis of robotic mobile fulfilment system based on cellular automata, *International Journal of Simulation Modelling*, Vol. 20, No. 3, 583-594, doi:10.2507/IJSIMM20-3-CO13
[12] Hwang, H. S.; Cho, G. S. (2006). A performance evaluation model for order picking warehouse design, *Computers & Industrial Engineering*, Vol. 51, No. 2, 335-342, doi:10.1016/j.cie.2005.10.002
[13] Yu, M.; de Koster, R. B. M. (2009). The impact of order batching and picking area zoning on order picking system performance, *European Journal of Operational Research*, Vol. 198, No. 2, 480-490, doi:10.1016/j.ejor.2008.09.011
[14] Yener, F.; Yazgan, H. R. (2023). Simulation of re-arrangement and healing in robotic compact bin-storage system, *International Journal of Simulation Modelling*, Vol. 22, No. 1, 100-109, doi:10.2507/IJSIMM22-1-635
[15] Liu, M. L.; Yao, X. Z.; Huang, J. Y.; Zhang, C. (2022). Optimization of unmanned vehicle scheduling and order allocation, *International Journal of Simulation Modelling*, Vol. 21, No. 3, 477-488, doi:10.2507/IJSIMM21-3-613
[16] Andradóttir, S.; Kim, S.-H. (2010). Fully sequential procedures for comparing constrained systems via simulation, *Naval Research Logistics*, Vol. 57, No. 5, 403-421, doi:10.1002/nav.20408
[17] Healey, C. M.; Andradóttir, S.; Kim, S.-H. (2013). Efficient comparison of constrained systems using dormancy, *European Journal of Operational Research*, Vol. 224, No. 2, 340-352, doi:10.1016/j.ejor.2012.08.012
[18] Healey, C.; Andradóttir, S.; Kim, S.-H. (2014). Selection procedures for simulations with multiple constraints under independent and correlated sampling, *ACM Transactions on Modeling and Computer Simulation*, Vol. 24, No. 3, 1-25, doi:10.1145/2567921
[19] Healey, C. M.; Andradóttir, S.; Kim, S.-H. (2015). A minimal switching procedure for constrained ranking and selection under independent or common random numbers, *IIE Transactions*, Vol. 47, No. 11, 1170-1184, doi:10.1080/0740817X.2015.1009198
[20] Lee, L. H.; Pujowidianto, N. A.; Li, L.-W.; Chen, C.-H.; Yap, C. M. (2012). Approximate simulation budget allocation for selecting the best design in the presence of stochastic constraints, *IEEE Transactions on Automatic Control*, Vol. 57, No. 11, 2940-2945, doi:10.1109/TAC.2012.2195931
[21] Hunter, S. R.; Pasupathy, R. (2013). Optimal sampling laws for stochastically constrained simulation optimization on finite sets, *INFORMS Journal on Computing*, Vol. 25, No. 3, 527-542, doi:10.1287/ijoc.1120.0519
[22] Pasupathy, R.; Hunter, S. R.; Pujowidianto, N. A.; Lee, L. H.; Chen, C.-H. (2015). Stochastically constrained ranking and selection via SCORE, *ACM Transactions on Modeling and Computer Simulation*, Vol. 25, No. 1, 1-26, doi:10.1145/2630066
[23] Law, A. M. (2015). *Simulation Modeling and Analysis*, 5th edition, McGraw-Hill, New York