Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9748378, 14 pages
http://dx.doi.org/10.1155/2016/9748378

Research Article

Formulations, Features of Solution Space, and Algorithms for Line-Pure *Seru* System Conversion

Wei Sun,[1] Qianqian Li,[1] Chunhui Huo,[1] Yang Yu,[2] and Ke Ma[2]

[1]*Business School, Liaoning University, Shenyang 110316, China*
[2]*Institute of Systems Engineering, State Key Laboratory of Synthetic Automation for Process Industries, Northeastern University, Shenyang 110819, China*

Correspondence should be addressed to Yang Yu; yuyang@ise.neu.edu.cn

Received 26 February 2016; Accepted 19 May 2016

Academic Editor: Yakov Strelniker

The line-*seru* conversion is usually used to improve productivity, especially in volatile business environment. Due to the simplicity, most researches focused on line-pure *seru* system conversion. We summarize the two existing models (i.e., a biobjective model and a single-objective model) of line-pure system conversion and formulate the three other usually used single-objective models in an integrated framework by combining evaluated performances and constraints. Subsequently, we analyze the solution space features of line-pure *seru* system conversion by dividing the whole solution space into several subspaces according to the number of *seru*s. We focus on investigating the features between $C_{\max}$ (and TLH) and subspaces. Thirdly, according to the distinct features between $C_{\max}$ (and TLH) and subspaces, we propose four effective algorithms to solve the four single-objective models, respectively. Finally, we evaluate the computational performance of the developed algorithms by comparing with enumeration based on extensive experiments.

1. Introduction

The *seru* production, conceived at Sony, is an innovation of assembly system used widely in the Japanese electronics industry and recognized a new production pattern. *Seru* production is proposed to overcome the less flexible shortcoming of assembly line, especially confronted with the dynamic and volatile business environments. *Seru* is an assembly unit including several simple equipment and one (or more) multiskilled operator(s). *Seru* is the pronunciation of cell in Japanese; therefore, *Seru* production is also called Japanese style cellular production. In *seru*, worker(s) must be multiskilled [1–4] because workers need to operate most or all the processes of production. In general, there are three types of *seru*: divisional *seru*, rotating *seru*, and *yatai* [5, 6]. In a divisional *seru*, tasks are divided into different sections and workers are partially cross-trained. Each section is operated by one or more workers. However, workers in rotating *seru* or *yatai* are completely cross-trained and do all tasks. In this research, rotating *seru*s or *yatai* are considered.

Seru system can be generally divided into two types: (1) pure *seru* system with only *seru*(s) and (2) hybrid system with *seru*(s) and short line. As Stecke et al. [5, 6] claimed, with combined strengths from Toyota's lean philosophy and Sony's one-person production organization, *seru* system is a more productive, efficient, and flexible system than conveyor assembly line. As we know, the productivity of the assembly line is decided by the worker with the worst skill. It is amazing that *seru* system can greatly decrease the influence of the worker with the worst skill on the productivity by worker reconfiguration. *Seru* is more flexible than assembly line, because the workers in *seru* system are multiskilled operators who can operate multiple tasks and process multiple products. *Seru*'s flexibility is different with cellular manufacturing, where workers operate the certain part/product family. *Seru* system can response to market change quickly, due to movable workstations, light equipment, and multiskilled workers. Moreover, *seru* system can be continuously improved, because workers in *seru* can study to process more tasks and products.

Due to the merit of *seru* system [5–7], *seru* system has successfully been applied by many leading Japanese companies such as Sony, Canon, Panasonic, NEC, Fujitsu, Sharp, and Sanyo. Presently, many companies converted assembly line into *seru* system (line-*seru* conversion) to increase the productivity [8–10]. By adopting line-*seru* conversion, Canon and Sony reduced 720,000 and 710,000 square meters of floor space, respectively [5, 6]. Moreover, Canon's costs were reduced significantly, by 55 billion yen in 2003, and by a total of 230 billion yen from 1998 to 2003. As a result, Canon emerged as a leading electronics maker [11, 12]. Other benefits [13, 14] include the reductions of throughput time, setup time, required labor hours, WIP inventories, and finished-product inventories. For example, the makespan was reduced by 53% at Sony Kohda and 35,976 required workers, that is equal to 25% of Canon's previous total workforce, have been saved. *Seru* systems also influence profits, product quality, and workforce motivation in a positive way.

The essence of line-*seru* conversion lies in how to convert traditional conveyor assembly line into a *seru* system to obtain the optimal conversion solution. It is a very hot and important decision making problem because the total productivity of manufacturers may be improved dramatically [15, 16]. Such technical and decision making problems were defined as line-*seru* (or line-cell) conversion problems [10, 16]. Kaku et al. [10, 16] built a biobjective line-*seru* conversion model with minimizing makespan (they use the term "total throughput time") and TLH. They claimed that their model considered three types of *seru* systems, that is, (1) pure *seru* system, (2) hybrid system with *seru*s and short line, and (3) assembly line. However, the third one, in fact, is not considered as a *seru* system, because it does not contain any *seru*. Therefore, there are only two types of *seru* systems, that is, pure *seru* system and hybrid system with *seru*s and short line.

Pure *seru* system is the *seru* system only containing *seru*(s) and is very simple and a special case of all other *seru* assembly systems. The results obtained from pure *seru* system models not only provide insights into the pure *seru* system environment but also provide a basis for heuristics that are applicable to more complicated *seru* system environments, for example, hybrid system with *seru*s and short line. In practice, problems in more complicated *seru* system environments are often decomposed into subproblems that deal with the pure *seru* system. Therefore, many literatures focused on the conversion of assembly line to pure *seru* system (line-pure *seru* system conversion), such as Yu et al. [17–20] and Sun et al. [21]. A simple case of line-pure *seru* system conversion is shown in Figure 1, where two *seru*s are constructed, that is, workers 2 and 5 in *seru* 1 and workers 1, 3, and 4 in *seru* 2.

To establish the main mathematical models is very important for the research on line-pure *seru* system conversion. Yu et al. [17] formulated the biobjective model of Min-$C_{\max}$ and TLH. Subsequently, the researches of Yu et al. [18, 20] were based on the biobjective model. Considering the biobjective model has higher computational complexity, Sun et al. [21] formulated the single-objective model of Min-$C_{\max}$ with TLH constraint. Additionally, there are several other usually used single-objective models in line-pure *seru* system conversion. We should not formulate one model in one research but integrate these models into a common framework. Therefore, the first objective of the research is to formulate the five models of line-pure *seru* system conversion in an integrated framework by combining the evaluated performances and constraints.

Moreover, the algorithms for line-pure *seru* system conversion are also the key. However, the existing algorithms for line-pure *seru* system conversion were based on enumeration (i.e., GA and local search) or metaheuristic and did not consider the distinct features of solution space of line-pure *seru* system conversion. For example, enumeration in Yu et al. [17] searched the whole solution space; Yu et al. [18, 20] and Sun et al. [21] used GA or local search to seek the optimal solution in the whole solution space. In fact, solution space of line-pure *seru* system conversion has the distinct features; that is, the solution space can be divided into several subspaces according to the number of *seru*s, and minimum $C_{\max}$ and minimum TLH usually exist in the special subspaces. That is to say, considering the features of solution space of line-pure *seru* system conversion, we maybe do not need to search the whole solution space to seek the optimal solution. Therefore, the second motivation of the research is to investigate the distinct features of solution space of line-pure *seru* system conversion. Subsequently, the third of objective is to develop the effective algorithms for the different models according to the obtained features of solution space of line-pure *seru* system conversion.

This paper, originally motivated by line-*seru* applications of Sony and Canon, has three purposes. First, two existing models (i.e., a biobjective model and a single-objective model) and the three other usually used single-objective models in line-pure system conversion are formulated in an integrated framework by combining evaluated performances with constraints. Subsequently, we analyze the solution space features of line-pure *seru* system conversion by dividing solution space into several subspaces according to the number of *seru*s. We focus on investigating the features between $C_{\max}$ (makespan) and TLH (total labor hours) and subspaces with different number of *seru*s. Thirdly, we propose four effective algorithms to solve the four single-objective model, respectively, according to the distinct features of line-pure *seru* system.

The remainder of this research is organized in the following ways. By combining the evaluated performances and constraints, Section 2 formulates five models usually used in line-pure *seru* system conversion in an integrated framework. Section 3 investigates the solution space feature of line-pure *seru* system conversion based on W divided subspaces according to the number of *seru*s. Subsequently, we focus on analyzing the features between $C_{\max}$ (and TLH) and subspaces with different number of *seru*s. In Section 4, according to the distinct features of solution space, especially the features between $C_{\max}$ (and TLH) and subspaces with different number of *seru*s, we propose four algorithms to solve the four single-objective models, respectively. Section 5 uses extensive experiments to evaluate the computational performance of the proposed algorithms by comparing with enumeration. Finally, we end the paper with conclusions and with suggestions for future research in Section 6 and give

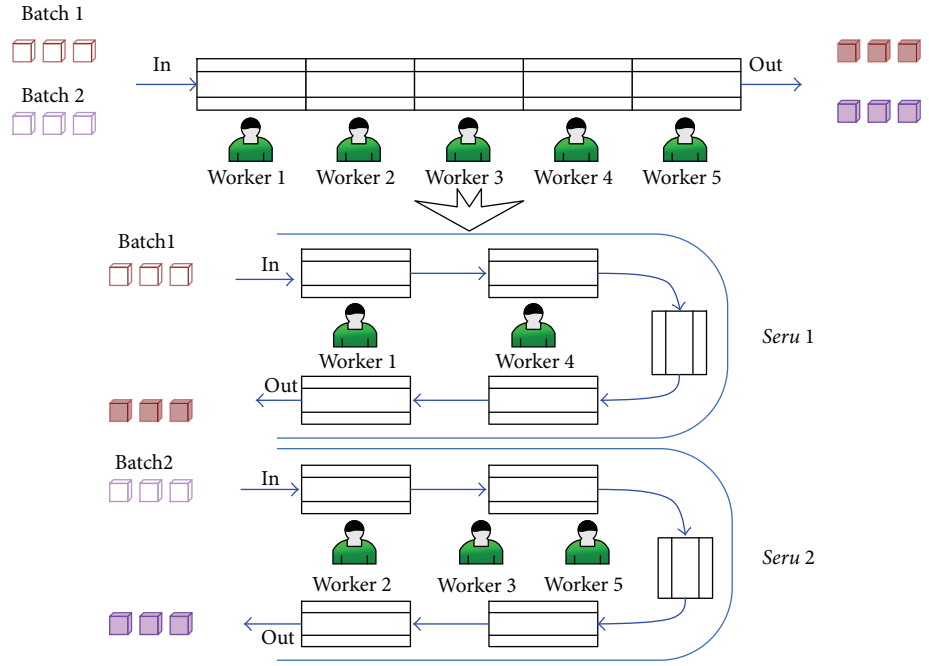

Figure 1: A case of line-pure *seru* system conversion [20].

the further research emphases, that is, the conversion from assembly line to the hybrid system with *seru*s and short line.

2. Formulation of Several Main Models of Line-Pure *Seru* System Conversion

2.1. Assumptions. The following assumptions are considered in line-pure *seru* system conversion:

(1) The types and batches of products to be processed are known in advance. There are N product types that are divided into M product batches. Each batch contains a single product type.

(2) In the line-pure *seru* system conversion, most assembly tasks within a *seru* are manual, so need for only simple and cheap equipment and the cost of duplicating equipment is ignored [5].

(3) A product batch needs to be assembled entirely within a single *seru*.

(4) All product types have the same assembly tasks. If tasks of some product were unique, we assume the task time for these unique tasks was zero.

(5) The assembly tasks within each *seru* are the same as the ones within the assembly line. Therefore, the total number of tasks equals W.

(6) In the assembly line, each task (or station) is in charge of a single worker. That means that a worker only performs a single assembly task in the assembly line. In contrast, a *seru* worker needs to perform all assembly tasks and assembles an entire product from-start-to-finish, and there is no disruption or delay between adjacent tasks.

2.2. Notations. We define the following terms:

(i) Indices

i: index of workers ($i = 1, 2, \ldots, W$). Also, the total number of tasks is W, according to assumption (5).

j: index of *seru*s ($j = 1, 2, \ldots, J$).

n: index of product types ($n = 1, 2, \ldots, N$).

m: index of product batches ($m = 1, 2, \ldots, M$).

k: index of the sequence of product batches in a *seru* ($k = 1, 2, \ldots, M$).

r: index of the sequence of product batches in the short line ($1, 2, \ldots, M$).

(ii) Parameters

$$V_{mn} = \begin{cases} 1, & \text{if product type of product batch } m \text{ is } n; \\ 0, & \text{othewise.} \end{cases} \quad (1)$$

B_m: size of product batch m.

T_n: cycle time of product type n in the assembly line.

SLP_n: setup time of product type n in the assembly line.

SCP_n: setup time of product type n in a *seru*.

η_i: upper bound on the number of tasks for worker i in a *seru*. If the number of tasks assigned to worker i

is more than η_i, worker i's average task time within a *seru* will be longer than her or his task time within the original assembly line.

ε_i: worker i's coefficient of influencing level of doing multiple assembly tasks.

β_{ni}: skill level of worker i for each task of product type n.

(iii) Decision Variables

$$X_{ij} = \begin{cases} 1, & \text{if worker } i \text{ is assigned to cell } j; \\ 0, & \text{othewise,} \end{cases}$$

$$Z_{mjk} = \begin{cases} 1, & \text{if product batch } m \text{ is assigned to cell } j \text{ in sequence } k; \\ 0, & \text{othewise.} \end{cases} \quad (2)$$

(iv) Variables

C_i: coefficient of variation of worker i's increased task time after line-*seru* conversion, that is, from a specialist to a completely cross-trained worker. If the number of worker i's tasks within a *seru* is over her or his upper bound η_i, that is, $W > \eta_i$, then the worker will cost more average task time than her or his task time within the original assembly line. C_i is given in (3).

TC_m: assembly task time of product batch m per station in a *seru*. In a *seru*, the task time of product type n is calculated by the average task time of workers in the *seru*. TC_m is represented as (4).

FC_m: flow time of product batch m in a *seru*. FC_m is represented as (5).

SC_m: setup time of product batch m in a *seru*. Setup time is considered when two different types of products are processed consecutively; otherwise, the setup time is zero. For example, in (6), two adjacent assembled products in a *seru* are expressed as m and m'. If the product type of m is different than that of m', that is, $V_{mn} = 1$, $V_{m'n} = 0$, then SC_m is SCP_nV_{mn}. However, if the product types of m and m' are identical, that is, $V_{mn} = V_{m'n} = 1$, then SC_m is 0.

FCB_m: begin time of product batch m in a *seru*. There is no waiting time between two product batches so that FCB_m is the aggregation of flow time and setup time of the product batches processed prior to product batch m in the same *seru*. FCB_m is represented as (7):

$$C_i = \begin{cases} 1 + \varepsilon_i (W - \eta_i), & W > \eta_i \\ 1, & W \le \eta_i, \end{cases} \quad \forall i, \quad (3)$$

$$TC_m = \frac{\sum_{n=1}^{N} \sum_{i=1}^{W} \sum_{j=1}^{J} \sum_{k=1}^{M} V_{mn} T_n \beta_{ni} C_i X_{ij} Z_{mjk}}{\sum_{i=1}^{W} \sum_{j=1}^{J} \sum_{k=1}^{M} X_{ij} Z_{mjk}}, \quad (4)$$

$$FC_m = \frac{B_m TC_m W}{\sum_{i=1}^{W} \sum_{j=1}^{J} \sum_{k=1}^{M} X_{ij} Z_{mjk}}, \quad (5)$$

$$SC_m = \begin{cases} SCP_n V_{mn}, & V_{mn} = V_{m'n} = 1 \\ 0, & V_{mn} = 1,\ V_{m'n} = 0, \end{cases} \quad (6)$$

$$\left(m' \mid Z_{mjk} = 1,\ Z_{m'j(k-1)} = 1,\ \forall j, k\right),$$

$$FCB_m = \sum_{p=1}^{m-1} \sum_{j=1}^{J} \sum_{k=1}^{m} \left(FC_p + SC_p\right) Z_{mjk} Z_{pj(k-1)}. \quad (7)$$

2.3. Evaluated Performances. The following two performances are usually used to evaluate *seru* system's productivity.

2.3.1. Makespan. Makespan ($C_{\max}$) of the pure *seru* system is the due time of the last completed product batch and can be expressed as

$$C_{\max} = \max_{m=1}^{M} \left(FCB_m + FC_m + SC_m\right). \quad (8)$$

2.3.2. Total Labor Hour. The total labor hours (TLH) are the work time of all workers assembling the total product batches:

$$\text{TLH} = \sum_{m=1}^{M} \sum_{i=1}^{W} \left(\sum_{j=1}^{J} \sum_{k=1}^{M} FC_m X_{ij} Z_{mjk} \right). \quad (9)$$

2.4. Constraints. Each decision variable determines one process. Therefore, the line-pure *seru* system conversion is a two-stage decision process, that is, *seru*-formation and *seru*-load, decided by X_{ij} and Z_{mjk}, respectively.

According to the two decision processes and the two evaluated performances, we divide the constraints into the following categories.

2.4.1. Seru-Formation Constraints. Consider

$$1 \le \sum_{i=1}^{W} X_{ij} \le W, \quad \forall j, \quad (10)$$

$$\sum_{j=1}^{J} X_{ij} = 1, \quad \forall i, \quad (11)$$

$$\sum_{j=1}^{J} \sum_{i=1}^{W} X_{ij} = W. \quad (12)$$

Equation (10) is the *seru*'s worker constraint which ensures that each formatted *seru* contains least one worker and most W workers. Equation (11) is the worker assignment constraint which guarantees that each worker must be only assigned to a *seru*. Equation (12) is the worker number constraint which guarantees that the total number of workers

in all formatted *seru*s equals W, that is, the total number of workers in the assembly line.

2.4.2. Seru-Load Constraints. Consider

$$\sum_{j=1}^{J}\sum_{k=1}^{M} Z_{mjk} = 1, \quad \forall m, \tag{13}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk} = 0, \quad \left(\forall j \mid \sum_{i=1}^{W} X_{ij} = 0\right). \tag{14}$$

Equation (13) is the *seru*-load constraint; that is, a product batch is only loaded to a *seru*. Equation (14) guarantees a product batch must be assigned to a *seru* in which at least one worker is assigned.

2.4.3. $C_{\max}$ Constraint. Consider

$$C_{\max} \text{ of pure } seru \text{ system} \le \text{given } C_{\max}. \tag{15}$$

Equation (15) constrains that pure *seru* system's $C_{\max}$ is not worse than the given value. For example, we can set the given $C_{\max}$ as 90% of the assembly line's $C_{\max}$.

2.4.4. TLH Constraint. Consider

$$\text{TLH of pure } seru \text{ system} \le \text{given TLH}. \tag{16}$$

Equation (16) constrains that total labor hours (TLH) of pure *seru* system are not worse than the given value.

2.5. Several Main Models of Line-Pure Seru System Conversion. By combining the above two evaluated performances of $C_{\max}$ and TLH with four constraints of *seru*-formation, *seru*-load, $C_{\max}$, and TLH, we formulate several main mathematical models of line-pure *seru* system conversion.

2.5.1. Model of Min-$C_{\max}$ (Model of Min-$C_{\max}$: min (8), s.t. (10)–(14)). Model of Min-$C_{\max}$ is to minimize makespan (i.e., (8)) of pure *seru* system. Therefore, objective is expressed as min (8) and constraints include *seru*-formation constraints and *seru*-load constraints, that is, (10)–(14).

2.5.2. Model of Min-$C_{\max}$ with TLH Constraint (Model of Min-$C_{\max}$ with TLH Constraint: min (8), s.t. (10)–(14), (16)). TLH constraint sometimes needs to be considered in model of Min-$C_{\max}$, and so model of Min-$C_{\max}$ with TLH constraint is formulated as above.

2.5.3. Model of Min-TLH (Model of Min-TLH: min (9), s.t. (10)–(14)). Model of Min-TLH is to minimize TLH (i.e., (9)) of pure *seru* system. Therefore, objective is expressed as min (9) and constraints include *seru*-formation constraints and *seru*-load constraints.

2.5.4. Model of Min-TLH with $C_{\max}$ Constraint (Model of Min-TLH with $C_{\max}$ Constraint: min (9), s.t. (10)–(14), (15)). $C_{\max}$ constraint sometimes can be considered in model of Min-TLH, and so model of Min-TLH with $C_{\max}$ constraint is formulated as above.

2.5.5. Biobjective Model of Min-$C_{\max}$ and TLH (Model of Min-$C_{\max}$ and TLH: min ((8) and (9)), s.t. (10)–(14)). Model of Min-$C_{\max}$ and TLH is to minimize $C_{\max}$ and TLH simultaneously. Therefore, biobjective is expressed as min ((8) and (9)) and constraints include *seru*-formation constraints and *seru*-load constraints.

2.6. Researches on the Five Models' Formulations. So far, most researches focused on the biobjective model of Min-$C_{\max}$ and TLH in Section 2.5.5. For example, Yu et al. [17, 20] formulated the biobjective model of Min-$C_{\max}$ and TLH and investigated the mathematical analysis and influence factors. In addition, as the biobjective model has higher computational complexity, Sun et al. [21] formulated the single-objective model of Min-$C_{\max}$ with TLH constraint in Section 2.5.2.

Although the three other models were not researched until now, they are usually considered in line-pure *seru* system conversion. In fact, they are similar to model of Min-$C_{\max}$ with TLH constraint in Section 2.5.2, because they are single-objective models. It is unacceptable to formulate one model in one research. Therefore, the first contribution of the research is to summarize the existing two models and to formulate the three other usually used single-objective models in a framework by combining the above two evaluated performances and four constraints.

In addition, previous researches did not investigate the distinct features of solution space of line-pure *seru* system conversion. As a result, the existing algorithms for line-pure *seru* system are not developed according to the distinct features.

3. Features of Solution Space of Line-Pure *Seru* System Conversion

3.1. Complexity of Solution Space Satisfying Seru-Formation and Seru-Load Constraints. In line-pure *seru* system conversion, *seru*-formation is the first step. It is to determine how many *seru*s to be formed and how to assign workers into the *seru*s [17]and it is decided by decision variable X_{ij}.

A feasible solution of *seru*-formation must satisfy *seru*-formation constraint, that is, (10)–(12). Yu et al. [17] proved that *seru*-formation is an instance of the unordered set partition and an NP-hard problem. The number of all feasible solutions of *seru*-formation can be expressed recursively as the following:

$$F(W) = \sum_{J=1}^{W} P(W, J), \tag{17}$$

where $P(W, J)$ is the count of partitioning W workers in assembly line into J *seru*s and can be expressed as the Stirling numbers of the second kind [22]. The numbers of $F(1)$ to $F(10)$ are 1, 2, 5, 15, 52, 203, 877, 4140, 21147, and 115975, respectively [23–25]. Obviously, $F(W)$ increases exponentially with W.

Seru-load is the second step of line-pure *seru* system conversion and is decided by decision variable Z_{mjk}. It

determines which product batches are dispatched to the *seru*s formed in *seru*-formation [26, 27]. A feasible solution of *seru*-load must satisfy *seru*-load constraint, that is, (13) and (14). *Seru*-load is NP-hard. Therefore, most researches used typical dispatching rules, such as FCFS (first-come, first-served) and SPT (shortest processing time). Even though the typical dispatching rules are used in *seru*-load, line-pure *seru* system conversion still is NP-hard, because it contains *seru*-formation which is NP-hard.

Yu et al. [20] proved that line-pure *seru* system conversion with SPT is an instance of the unordered set partition and the complexity of solution space can be expressed as (17). Line-pure *seru* system conversion with FCFS is an instance of the ordered set partition and the complexity of solution space can be expressed as

$$T(W) = \sum_{J=1}^{W} P(W, J) * J!. \tag{18}$$

The numbers of $T(1)$ to $T(10)$ are 1, 3, 13, 75, 541, 4,683, 47,293, 545,835, 7,087,261, and 102,247,563, respectively. In the research, we use FCFS rule to assign product batches to *seru*s.

3.2. Solution Spaces of the Five Models. In the above five models, model of Min-$C_{\max}$, model of Min-TLH, and biobjective model of Min-$C_{\max}$ and TLH have the identical solution space described in Section 3.1, because all of them only include *seru*-formation constraints and *seru*-load constraints.

However, model of Min-$C_{\max}$ with TLH constraint and model of Min-TLH with $C_{\max}$ constraint have the different solution space, because the two models contain the other constraints except *seru*-formation constraints and *seru*-load constraints. Therefore, the solution spaces of the two models must be not more than that of Section 3.1.

3.3. Existing Algorithms for the Five Models. Most researches focused on the algorithms for the biobjective model of Min-$C_{\max}$ and TLH in Section 2.5.5. Yu et al. [20] proposed a based-NSGA-II algorithm to solve the biobjective model; Yu et al. [18] combined local search into NSGA-II to improve the running time and solutions qualification; Yu et al. [17] used enumeration and based-NSGA-II algorithm to solve the biobjective model to analyze the impact factor on $C_{\max}$ and TLH improvements by line-pure *seru* system conversion. Sun et al. [21] developed a variable neighborhood search (VNS) for model of Min-$C_{\max}$ with TLH constraint in Section 2.5.2.

However, the algorithms for the three other models were not researched until now.

The existing algorithms for line-pure *seru* system conversion were based on the metaheuristic (i.e., GA and local search) and did not consider the distinct features of solution space. For example, enumeration in [17] searched the whole solution space; Yu et al. [18, 20] and Sun et al. [21] used GA or local search to seek the optimal solution in the whole solution space. In fact, solution space of line-pure *seru* system conversion has the distinct features; that is, the solution space can be divided into several subspaces according to the number of *seru*s, and minimum $C_{\max}$ and minimum TLH usually exist in the special subspaces. That is to say, considering the features of solution space of line-pure *seru* system conversion, we maybe do not need to search the whole solution space to seek the optimal or suboptimal solution.

Therefore, we investigate the distinct features of solution space of line-pure *seru* system conversion and then propose effective algorithms to solve the four single-objective models based on the distinct features of solution space.

3.4. Subspaces with Different Number of Serus

Definition 1. Subspace in solution space of line-pure *seru* system conversion is the set of solutions with the same number of *seru*s.

For example, of (18), the whole solution space can be divided into W subspaces according to the number of *seru*s, that is, the subspaces with $1, 2, \ldots$, and W *seru*s. Therefore, (18) can be expressed as the following:

$$T(W) = P(W, 1) * 1! + \cdots + P(W, J) * J! + \cdots + P(W, W) * W!. \tag{19}$$

Table 1 summarizes the number of solutions in subspace with J *seru*s for the lines with 6–9 workers.

The value in column of "workers" represents the number of workers in the assembly line. For the row "6," the numbers of solutions in subspaces with 1, 2, 3, 4, 5, and 6 *seru*s are $1 = P(6,1) * 1!$, $62 = P(6,2) * 2!$, $540 = P(6,3) * 3!$, $1560 = P(6,4) * 4!$, $1800 = P(6,5) * 5!$, and $720 = P(6,6) * 6!$, respectively.

To show the proportion of solutions in each subspace to the total solutions in the whole solution space, Table 2 normalizes the data in Table 1.

From Table 2, we can see that the number of solutions in subspaces with few or more *seru*s is very small. Therefore, if minimum $C_{\max}$ and minimum TLH exist in such subspaces, we can dramatically decrease the searching space.

3.5. Value of $C_{\max}$ and TLH in Subspaces. Yu et al. [17] performed 64 arrays of full factorial experiment to analyze influence factors and proposed some managerial insights. Two insights on $C_{\max}$ and TLH are as follows: (1) to minimize $C_{\max}$, the pure *seru* system with fewer *seru*s should be created and assign the workers with similar skill levels for product types to the same *seru*; and (2) to minimize TLH, the pure *seru* system with more *seru*s should be created.

Yu et al. [17] used the solutions with 1 *seru* and W *seru*s to briefly explain the two insights: (1) they proved that when $W \leq \eta_i$, if the *seru* system with a *seru* is formatted, then the $C_{\max}$ performance of pure *seru* system must be better than that of the assembly line. Additionally, they stated that the pure *seru* system with fewer *seru*s is easy to balance $C_{\max}$ among *seru*s than the pure *seru* system with more *seru*s. Therefore, the pure *seru* system with fewer *seru*s can produce better $C_{\max}$; and (2) they proved that when $W \leq \eta_i$ and $W \leq M$, the pure *seru* system with W *seru*s should be formatted in order to minimize TLH, because this makes full use of each worker's skill.

To further investigate the two insights, in the research, we analyze the features of $C_{\max}$ and TLH in each subspace in

Table 1: Number of solutions in subspace with *J seru*s.

Workers	Number of solutions in subspace with *J seru*s									Total solutions
	1	2	3	4	5	6	7	8	9	
6	1	62	540	1560	1800	720	—	—	—	4683
7	1	126	1806	8400	16800	15120	5040	—	—	47293
8	1	254	5796	40824	126000	191520	141120	40320	—	545835
9	1	510	18150	186480	834120	1905120	2328480	1451520	362880	7087261

Table 2: Proportion of solutions in subspace with *J seru*s.

Workers	Proportion of solutions in subspace with *J seru*s								
	1	2	3	4	5	6	7	8	9
6	0.02	1.32	11.53	33.31	38.44	15.37	—	—	—
7	0.00	0.27	3.82	17.76	35.52	31.97	10.66	—	—
8	0.00	0.05	1.06	7.48	23.08	35.09	25.85	7.39	—
9	0.00	0.01	0.26	2.63	11.77	26.88	32.85	20.48	5.12

detail. Figures 2–4 show $C_{\max}$ and TLH of all solutions in each subspace for the cases with 6–8 workers, respectively. The results of Figures 2–4 are exactly obtained by enumeration. The detailed data of the used cases with 6–8 workers are described in Section 5.1.

3.6. Distinct Features between $C_{\max}$ (and TLH) and Subspaces. By observing $C_{\max}$ and TLH in Figures 2–4, we can investigate the distinct features between $C_{\max}$ (and TLH) and subspaces as follows.

Feature 1. Minimum $C_{\max}$ usually exists in the subspaces with few *seru*s.

Explanation. As shown in Figures 2–4, the minimum $C_{\max}$ always exists in the subspaces with 2 *seru*s.

Feature 2. Except the subspace with 1 *seru*, the minimum $C_{\max}$ of the subspaces with fewer *seru*s usually is less than that of the subspaces with more *seru*s.

Explanation. As shown in Figure 3, the minimum $C_{\max}$ of the subspaces with J ($1 < J < 7$) *seru*s always is less than that of the subspaces with $J + 1$ *seru*s. The trend can be found in all of Figures 2 and 3. Table 3 shows the detailed information on the minimum $C_{\max}$ of the subspaces with *J seru*s for the cases with 6–8 workers.

Features 1 and 2 mean that it not necessary to search the whole solution space to obtain minimum $C_{\max}$, especially in single-objective models. To fast obtain minimum $C_{\max}$, therefore, we can search the subspaces from fewer *seru*s to more *seru*s. Consequently, we propose effective algorithms to fast find minimum $C_{\max}$. The detailed procedure is described in Section 4.1.

Feature 3. Minimum TLH usually exists in the subspaces with more *seru*s.

Explanation. Assume *M_J* express the maximum J. In Figures 2–4, the minimum TLH always exists in the subspace with *M_J seru*s.

Feature 4. Minimum TLH of the subspaces with more *seru*s usually is less than that of the subspaces with fewer *seru*s.

Explanation. As shown in Figures 2–4, the minimum TLH of the subspaces with *J seru*s always is less than that of the subspaces with $J - 1$ *seru*s. Table 4 shows the detailed information on the minimum TLH of the subspaces with *J seru*s for the cases with 6–8 workers.

Features 3 and 4 mean that it not necessary to search the whole feasible solution space to obtain the minimum TLH, especially in single-objective models. To fast obtain minimum TLH, therefore, we can search the subspaces from more *seru*s to few *seru*s. Consequently, we propose effective algorithms to fast find the minimum TLH. The detailed procedure is described in Section 4.2.

4. Algorithms for the Single-Objective Line-Pure *Seru* System Conversion

As mentioned above, most existing algorithms were for the biobjective line-pure *seru* system conversion. The algorithm of Sun et al. [21] was for single-objective model. However, these algorithms were not developed based on the features of solution space of line-pure *seru* system conversion, which decreases their running efficiency. Therefore, we propose algorithms to solve the 4 single-objective models based on the distinct features of solution space.

4.1. Algorithms for Models of Min-$C_{\max}$ and Min-$C_{\max}$ with TLH Constraint. To avoid the obtained solution with minimum $C_{\max}$ being locally optimal, the algorithm does not search the subspace with the least *seru* (i.e., 1 *seru*) but searches subspaces from fewer *seru*s to more *seru*s to seek

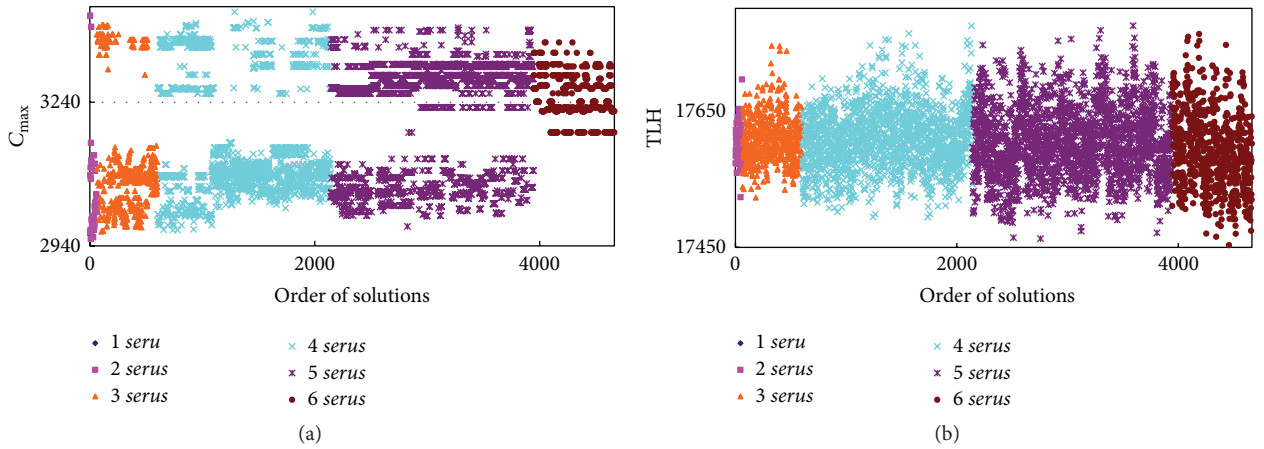

Figure 2: C_{max} and TLH in subspaces for the line with 6 workers.

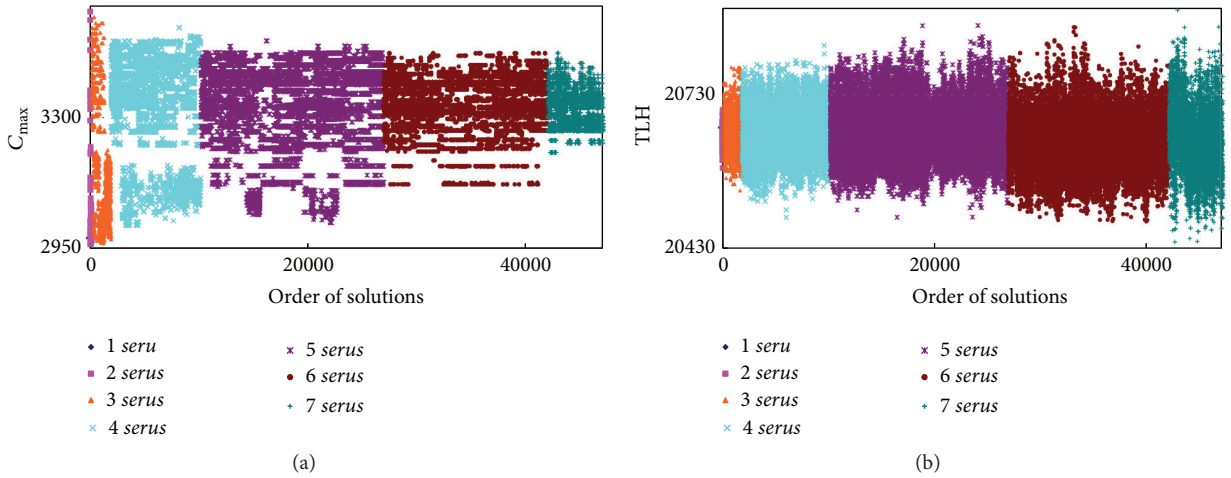

Figure 3: C_{max} and TLH in subspaces for the line with 7 workers.

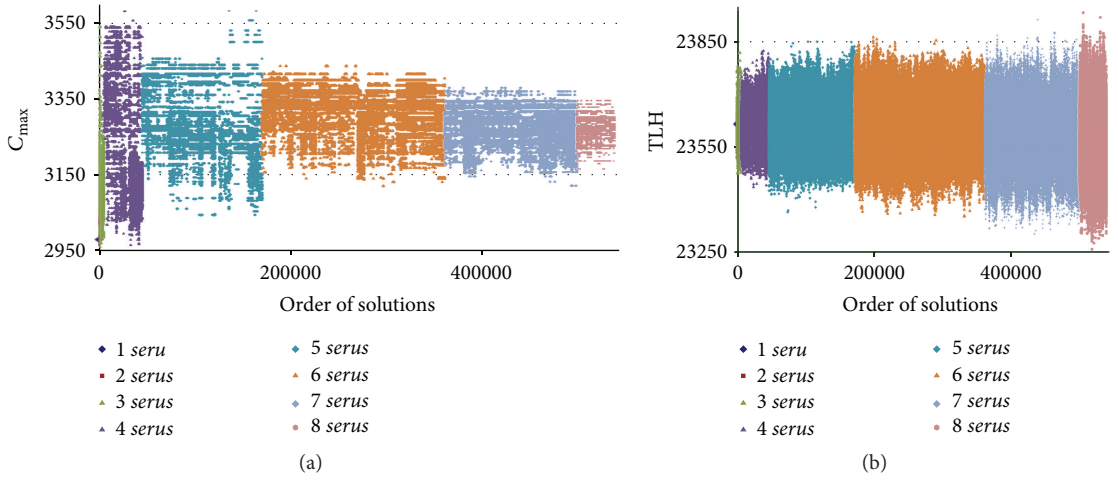

Figure 4: C_{max} and TLH in subspaces for the line with 8 workers.

Table 3: Minimum $C_{\max}$ in subspace with J *seru*s.

Workers	Minimum $C_{\max}$ in subspace with J *seru*s							
	1	2	3	4	5	6	7	8
6	2963	2954	2970	2973	2981	3177	—	—
7	2980	2963	2966	3014	3021	3123	3208	—
8	2980	2962	2971	2971	3045	3121	3122	3163

Table 4: Minimum TLH in subspace with J *seru*s.

Workers	Minimum TLH in subspace with J *seru*s							
	1	2	3	4	5	6	7	8
6	17610	17559	17534	17494	17463	17453	—	—
7	20665	20586	20506	29505	20504	20482	20442	—
8	23641	23519	23475	23437	23365	23354	23303	23260

minimum $C_{\max}$. Therefore, the minimum $C_{\max}$ may be obtained by searching only a part of solution space.

For an instance of Min-$C_{\max}$ with W workers, after obtaining the optimal solution in subspace with J *seru*, the algorithm searches the optimal solution in subspace with $J+1$ *seru*s. If the latter is not better than the former, then the algorithm will stop; otherwise, continue, until the optimal solution with $J+1$ *seru*s is not better than the optimal solution with J *seru*s. The procedure can be described as shown in Algorithm 1.

In Steps (2) and (3-3), $J = J + 1$ means to search solution space from few *seru*s to more *seru*s. Step (3-3) is to judge if the algorithm stops. If yes, output O; otherwise, continue to search the optimal $C_{\max}$ in the next subspace.

Similar to Algorithm 1, algorithm for models of Min-$C_{\max}$ with TLH constraint can be expressed as shown in Algorithm 2.

Distinguished from Algorithm 1, Algorithm 2 needs to calculate TLH and to judge if TLH constraint is satisfied. They are described in steps (1-1) and (1-2).

4.2. Algorithm for Model of Min-TLH and Min-TLH with $C_{\max}$ Constraint. To avoid the obtained solution with minimum TLH being locally optimal, the algorithm does not search the subspace with the most *seru* (i.e., W *seru*) but searches subspaces from more *seru*s to few *seru*s to seek the minimum TLH. Therefore, the minimum $C_{\max}$ may be obtained by searching only a part of solution space.

For an instance of Min-TLH with W workers, after obtaining the optimal solution in subspace with J *seru*, the algorithm searches the optimal solution in subspace with $J-1$ *seru*s. If the latter is not better than the former, then the algorithm will stop; otherwise, continue, until the optimal solution with $J-1$ *seru*s is not better than the optimal solution with J *seru*s. The procedure can be described as shown in Algorithm 3.

In Steps (2) and (3-3), $J = J - 1$ means to search solution space from more *seru*s to few *seru*s. Step (3-3) is to judge if the algorithm stops. If yes, output O; otherwise, continue to search the optimal TLH in the next subspace.

Table 5: The parameters used in experiments.

Product types	Batch size	ε_i	SL_n	SCP_n	T_n	η_i
5	$N(50, 5)$	$N(0.2, 0.05)$	2.2	1.0	1.8	10

$N(50, 5)$: normal distribution ($\mu = 50$, $\sigma = 5$).

Table 6: Data distribution of worker's level of skill (β_{ni}).

Product types				
1	2	3	4	5
$N(1, 0.1)$	$N(1.05, 0.1)$	$N(1.1, 0.1)$	$N(1.15, 0.1)$	$N(1.2, 0.1)$

Similar to Algorithm 3, algorithm for models of Min-TLH with $C_{\max}$ constraint can be expressed as shown in Algorithm 4.

Distinguished from Algorithm 3, Algorithm 4 needs to calculate $C_{\max}$ and to judge if $C_{\max}$ constraint is satisfied. They are described in steps (1-1) and (1-2).

4.3. Combining Features 1 and 2 into the Existing Algorithm. As mentioned above, Sun et al. [21] developed a variable neighborhood search (VNS) for model of Min-$C_{\max}$ with TLH constraint. However, their VNS seeks the optimal solution in the whole solution space. Therefore, considering Features 1 and 2, that is, to search minimum $C_{\max}$ satisfying TLH constraint in the subspaces with fewer *seru*s by redefining the neighborhood strategies, Sun's algorithm running time will be improved.

5. Computational Experiments

5.1. Test Instances. Tables 5–9 show the parameters, data distribution and detailed data of level of skill of workers, coefficient of influencing level of skill to multiple stations for workers, and data of batches used in experiments, respectively. From Table 5, it can be observed that the lot size of each batch is $N(50, 5)$ and the ability of workers is also different with stations and $N(0.2, 0.05)$. Table 6 shows that the mean of skill level (β_{ni}) of each worker for processing n product types

Input: W (the number of workers in assembly line).
Output: The optimal solution of Min-$C_{\max}$.
(1) Initialize. Set O = *null* (record the optimal solution), $J = 1$.
    (1-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's $C_{\max}$.
    (1-2) Obtain the optimal $C_{\max}$ in S_J.
    (1-3) O = optimal $C_{\max}$ in S_J;
(2) $J = J + 1$;
(3) While $J < W$ Do
    (3-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's $C_{\max}$.
    (3-2) Obtain the optimal $C_{\max}$ in S_J.
    (3-3) If optimal $C_{\max}$ in S_J < optimal $C_{\max}$ in S_{J-1} Then
        O = optimal $C_{\max}$ in S_J;
        $J = J + 1$;
        Continue;
    Else
        Break;
(4) Output O.

Algorithm 1: Algorithm for models of Min-$C_{\max}$.

Input: W (the number of workers in assembly line).
Output: The optimal solution of Min-$C_{\max}$ with TLH constraint.
(1) Initialize. Set O = *null* (record the optimal solution), $J = 1$.
    (1-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's $C_{\max}$ and TLH.
    (1-2) Produce the feasible solution set (F_J) of S_J, that is, the set of solutions satisfying TLH constraint.
    (1-3) Obtain the optimal $C_{\max}$ in F_J.
    (1-4) O = optimal $C_{\max}$ in F_J;
(2) $J = J + 1$;
(3) While $J < W$ Do
    (3-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's $C_{\max}$ and TLH.
    (3-2) Produce the feasible solution set (F_J) of S_J, that is, the set of solutions satisfying TLH constraint.
    (3-3) Obtain the optimal $C_{\max}$ in F_J.
    (3-3) If optimal $C_{\max}$ in F_J < optimal $C_{\max}$ in F_{J-1} Then
        O = optimal $C_{\max}$ in F_J;
        $J = J + 1$;
        Continue;
    Else
        Break;
(4) Output O.

Algorithm 2: Algorithm for models of Min-$C_{\max}$ with TLH constraint.

Input: W (the number of workers in assembly line).
Output: The optimal solution of Min-TLH.
(1) Initialize. Set O = *null* (record the optimal solution), $J = W$.
    (1-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's TLH.
    (1-2) Obtain the optimal TLH in S_J.
    (1-3) O = optimal TLH in S_J;
(2) $J = J - 1$;
(3) While $J > 1$ Do
    (3-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's TLH.
    (3-2) Obtain the optimal TLH in S_J.
    (3-3) If optimal TLH in S_J < optimal TLH in S_{J-1} Then
        O = optimal TLH in S_J;
        $J = J - 1$;
        Continue;
    Else
        Break;
(4) Output O.

Algorithm 3: Algorithm for models of Min-TLH.

Input: W (the number of workers in assembly line).
Output: The optimal solution of Min-TLH with $C_{\max}$ constraint.
(1) Initialize. Set O = *null* (record the optimal solution), $J = W$.
    (1-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's TLH and $C_{\max}$.
    (1-2) Produce the feasible solution set (F_J) of S_J, that is, the set of solutions satisfying $C_{\max}$ constraint.
    (1-3) Obtain the optimal TLH in S_J.
    (1-4) O = optimal TLH in S_J;
(2) $J = J - 1$;
(3) While $J > 1$ Do
    (3-1) Generate the solution set (S_J) of sub-space with J *serus* according to $P(W, J) * J!$ and calculate each solution's TLH and $C_{\max}$.
    (3-2) Produce the feasible solution set (F_J) of S_J, that is, the set of solutions satisfying $C_{\max}$ constraint.
    (3-3) Obtain the optimal TLH in S_J.
    (3-4) If optimal TLH in S_J < optimal TLH in S_{J-1} Then
        O = optimal TLH in S_J;
        $J = J - 1$;
        Continue;
    Else
        Break;
(4) Output O.

Algorithm 4: Algorithm for models of Min-TLH with $C_{\max}$ constraint.

Table 7: Data of worker's level of skill (β_{ni}).

Worker/product type	1	2	3	4	5
1	0.92	0.96	1.04	1.09	1.2
2	0.95	0.97	1.09	1.12	1.18
3	0.99	1.01	1.05	1.09	1.21
4	1.03	1.07	1.09	1.12	1.25
5	0.96	1.02	1.05	1.1	1.18
6	1.01	1.1	1.1	1.15	1.23
7	1.04	1.07	1.09	1.17	1.24
8	0.98	1.02	1.1	1.11	1.2
9	0.97	1.03	1.12	1.19	1.26
10	0.98	1.06	1.13	1.18	1.28

Table 8: Coefficient of influencing level of skill to multiple stations for workers (ε_i).

Worker	1	2	3	4	5	6	7	8	9	10
ε_i	0.18	0.19	0.2	0.21	0.2	0.2	0.2	0.22	0.19	0.19

ranges from 1 to 1.2 and the standard deviations are fixed to 0.1. The detailed data of β_{ni} are given in Table 7. The detailed data of ε_i and batches are given in Tables 8 and 9, respectively. In fact, data in Tables 5–9 is the part of data used in Yu et al. [20].

For the instance with W workers, we use the following data set from Tables 5–9: entire Table 5, first W rows of Table 7, first W columns of Table 8, and entire Table 9.

5.2. Hardware and Software Specifications. The four algorithms were coded in C# and executed on an Intel Core™ 2 processors at 2.66 GHz under Windows XP using 3.49 GB of RAM.

5.3. Comparative Results with Enumeration. Considering the similarity between Algorithm 1 (or 3) and Algorithm 2 (or 4), we used the Algorithms 1 and 3 to solve the models of Min-$C_{\max}$ and Min-TLH, respectively. The performance comparative results to the enumeration are shown in Table 10.

From Table 10, we can see the optimal solutions obtained by Algorithms 1 and 3 are same as the results from enumeration. Moreover, the running time is much less than that of enumeration, because algorithms search the corresponding subspaces according to the distinct features between $C_{\max}$ (and TLH) and subspaces instead of searching the whole solution space. In addition, the running time of Algorithm 3 is more than that of Algorithm 1. That is because the subspace with more *seru*s has more feasible solutions than the subspace with few *seru*s, as shown in Table 1.

6. Conclusions and Future Research

In the research, three main contributions can be summarized as follows.

Firstly, by combining two common evaluated performances of $C_{\max}$ and TLH with four constraints of *seru*-formation, *seru*-load, $C_{\max}$, and TLH, we summarize the two existing models and formulate three other usually used models in line-pure *seru* system conversion. The two evaluated performances, four classified constraints, and formulated models are summarized in Table 11.

Table 9: Data of batches.

Batch number	Product type	Batch size (B_m)
1	3	55
2	5	53
3	3	54
4	4	49
5	1	49
6	4	55
7	1	54
8	2	48
9	2	48
10	3	48
11	2	46
12	4	58
13	3	48
14	4	52
15	5	48
16	5	51
17	1	54
18	4	57
19	2	54
20	5	49
21	1	53
22	3	46
23	4	45
24	5	46
25	2	45
26	3	44
27	1	53
28	4	47
29	2	53
30	3	52

Secondly, we investigate the solution space feature of line-pure *seru* system conversion by dividing them into W subspaces according to the number of *seru*s (J). Subsequently, we focus on analyzing the features between $C_{\max}$ (and TLH) and subspaces in detail. The obtained distinct features are shown in Table 12.

Thirdly, according to the distinct features of solution space, especially the features between $C_{\max}$ (and TLH) and subspaces, we propose four algorithms to solve four single-objective models, respectively. Table 13 summarizes each proposed algorithm, based-on-features, and solved model.

The research on the line-*seru* conversion is relatively lacking. A thorough research problem list can be found in Yin et al. [20], such as partially cross-trained workers (i.e., a worker cannot perform all assembly tasks), different products

Table 10: Performance comparative results to the enumeration.

Workers	Line		Enumeration			Algorithm 1		Algorithm 3	
	$C_{\max}$	TLH	$C_{\max}$	TLH	Time (in s)	$C_{\max}$	Time (in s)	TLH	Time (in s)
6	3394	19995	2954	17602	6.43	2954	0.72	17602	3.12
7	3468	23844	2963	20627	22	2963	1.03	20627	9.08
8	3522	27686	2962	23542	157	2962	5.1	23542	51.3
9	3608	31917	2974	26582	1859	2974	9.2	26582	463.2

Table 11: Evaluated performances, constraints, and formulated models.

Evaluated performances	Expression
Makespan ($C_{\max}$)	$C_{\max} = \max_{m=1}^{M}(FCB_m + FC_m + SC_m)$, (8)
Total labor hours (TLH)	$\text{TLH} = \sum_{m=1}^{M}\sum_{i=1}^{W}(\sum_{j=1}^{J}\sum_{k=1}^{M} FC_m X_{ij} Z_{mjk})$, (9)
Constrains	Equations
Seru-formation	(10), (11), and (12)
Seru-load	(13) and (14)
$C_{\max}$	(15)
TLH	(16)
Models	Expression
Min-$C_{\max}$	min (8), s.t. (10)–(14)
Min-$C_{\max}$ with TLH constraint	min (8), s.t. (10)–(14), (16)
Min-TLH	min (9), s.t. (10)–(14)
Min-TLH with $C_{\max}$ constraint	min (9), s.t. (10)–(14), (15)
Min-$C_{\max}$ and TLH	min ((8) and (9)), s.t. (10)–(14)

Table 12: Distinct features between $C_{\max}$ (and TLH) and subspaces.

Features	Content
1	Minimum $C_{\max}$ usually exists in the subspaces with few *seru*s.
2	Except for the subspace with 1 *seru*, the minimum $C_{\max}$ of the subspaces with fewer *seru*s usually is less than that of the subspaces with more *seru*s.
3	Minimum TLH usually exists in the subspaces with more *seru*s.
4	Minimum TLH of the subspaces with more *seru*s usually is less than that of the subspaces with fewer *seru*s.

have different assembly tasks, cost of *karakuri* (a Japanese word meaning duplication of equipment), and human and psychology factors.

In addition, line-hybrid *seru* system conversion (i.e., the hybrid system with *seru*s and short line is the second type of the line-*seru* conversion defined by Kaku et al. [10]) is more complex and more real than line-pure *seru* system conversion. Therefore, the further research should be focused on the line-hybrid *seru* system conversion.

Table 13: Algorithm features and solved model.

Algorithms	Based-on-features	Solved model
Algorithm 1	Features 1 and 2	Min-$C_{\max}$
Algorithm 2	Features 1 and 2	Min-$C_{\max}$ with TLH constraint
Algorithm 3	Features 3 and 4	Min-TLH
Algorithm 4	Features 3 and 4	Min-TLH with $C_{\max}$ constraint

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (71420107028 and 71571037) and the National Social Science Foundation of China (13CGL045).

References

[1] T. Kimura and M. Yoshita, "*Seru* systems run into trouble when nothing is done," *Nikkei Monozukuri*, vol. 7, pp. 38–61, 2004 (Japanese).

[2] C. Liu, N. Yang, W. Li, J. Lian, S. Evans, and Y. Yin, "Training and assignment of multi-skilled workers for implementing *seru* production systems," *International Journal of Advanced Manufacturing Technology*, vol. 69, no. 5–8, pp. 937–959, 2013.

[3] C. Liu, J. Lian, Y. Yin, and W. Li, "*Seru* Seisan-an innovation of the production management mode in Japan," *Asian Journal of Technology Innovation*, vol. 18, no. 2, pp. 89–113, 2010.

[4] C. Liu, K. E. Stecke, J. Lian, and Y. Yin, "An implementation framework for *seru* production," *International Transactions in Operational Research*, vol. 21, no. 1, pp. 1–19, 2014.

[5] K. E. Stecke, Y. Yin, I. Kaku, and Y. Murase, "*Seru*: the organizational extension of JIT for a super-talent factory," *International Journal of Strategic Decision Sciences*, vol. 3, no. 1, pp. 105–118, 2012.

[6] K. E. Stecke, Y. Yin, and I. Kaku, "*Seru* production: an extension of just-in-time approach for volatile business environments," *Analytical Approaches to Strategic Decision-Making: Interdisciplinary Considerations*, pp. 45–58, 2014.

[7] N. Takeuchi, *Seru Production System*, JMA Management Center, Tokyo, Japan, *Seru* Seisan, 2006 (Japanese).

[8] J. Bukchin, E. Darel, and J. Rubinovitz, "Team-oriented assembly system design: a new approach," *International Journal of Production Economics*, vol. 51, no. 1-2, pp. 47–57, 1997.

[9] S. Cao, *Production reform: Seru cases in Japan and China [M.S. thesis]*, Yamagata University, Yamagata, Japan, 2008 (Japanese).

[10] I. Kaku, J. Gong, J. Tang, and Y. Yin, "Modeling and numerical analysis of line-cell conversion problems," *International Journal of Production Research*, vol. 47, no. 8, pp. 2055–2078, 2009.

[11] Y. Yin, "The direction of Samsung style next generation production methods," A Speech given at the Samsung Production Methods Innovation Forum, October 17. Samsung Electronics at Suwon city, Korea, 2006.

[12] Y. Yin, K. E. Stecke, and I. Kaku, "The evolution of *seru* production systems throughout Canon," *Operations Management Education Review*, vol. 2, pp. 27–40, 2008.

[13] Y. Sakazume, "Is Japanese cell manufacturing a new system? A comparative study between Japanese cell manufacturing and cell manufacturing," *Journal of Japan Industrial Management Association*, vol. 12, pp. 89–94, 2005.

[14] D. I. Miyake, "The shift from belt conveyor line to work-cell based assembly systems to cope with increasing demand variation in Japanese industries," *International Journal of Automotive Technology and Management*, vol. 6, no. 4, pp. 419–439, 2006.

[15] D. J. Johnson, "Converting assembly lines to assembly cells at Sheet Metal Products: insights on performance improvements," *International Journal of Production Research*, vol. 43, no. 7, pp. 1483–1509, 2005.

[16] I. Kaku, J. Gong, J. Tang, and Y. Yin, "A mathematical model for converting conveyor assembly line to cellular manufacturing," *International Journal of Industrial Engineering and Management Science*, vol. 7, no. 2, pp. 160–170, 2008.

[17] Y. Yu, J. Gong, J. Tang, Y. Yin, and I. Kaku, "How to carry out assembly line-cell conversion? A discussion based on factor analysis of system performance improvements," *International Journal of Production Research*, vol. 50, no. 18, pp. 5259–5280, 2012.

[18] Y. Yu, J. Tang, W. Sun, Y. Yin, and I. Kaku, "Combining local search into non-dominated sorting for multi-objective line-cell conversion problem," *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 4, pp. 316–326, 2013.

[19] Y. Yu, J. Tang, W. Sun, Y. Yin, and I. Kaku, "Reducing worker(s) by converting assembly line into a pure cell system," *International Journal of Production Economics*, vol. 145, no. 2, pp. 799–806, 2013.

[20] Y. Yu, J. Tang, J. Gong, Y. Yin, and I. Kaku, "Mathematical analysis and solutions for multi-objective line-cell conversion problem," *European Journal of Operational Research*, vol. 236, no. 2, pp. 774–786, 2014.

[21] W. Sun, Y. Yu, J.-F. Tang, Y. Yin, and I. Kaku, "Variable neighborhood search for line-cell conversion towards increasing productivity," *Computer Integrated Manufacturing Systems*, vol. 20, no. 12, pp. 3040–3047, 2014.

[22] B. C. Rennie and A. J. Dobson, "On Stirling numbers of the second kind," *Journal of Combinatorial Theory*, vol. 7, no. 2, pp. 116–121, 1969.

[23] M. Williams, "Back to the past: some plants, especially in Japan, are switching to craft work from assembly lines," *The Wall Street Journal*, 1994.

[24] M. Klazar, "Bell numbers, their relatives, and algebraic differential equations," *Journal of Combinatorial Theory. Series A*, vol. 102, no. 1, pp. 63–87, 2003.

[25] A. Knopfmacher and M. Mays, "Ordered and unordered factorizations of integers," *Mathematica Journal*, vol. 10, no. 1, pp. 72–89, 2006.

[26] C. Miao and J. Zou, "Parallel-machine scheduling with time-dependent and machine availability constraints," *Mathematical Problems in Engineering*, vol. 2015, Article ID 956158, 6 pages, 2015.

[27] Y. Zhou and Q. Zhang, "Multiple-machine scheduling with learning effects and cooperative games," *Mathematical Problems in Engineering*, vol. 2015, Article ID 197123, 7 pages, 2015.