D B S －19－07

「Simulated annealing and genetic algorithm based method for a bi-Level *seru* loading problem with worker assignment in seru production systems」

Lan Luo and Zhe Zhang_

School Economics and Management, Nanjing University of Science and Technology

Nanjing 210094, P. R. China

Yong Yin

Graduate School of Business, Doshisha University

September 2019

DBS Discussion Paper Series supported by the OMRON Foundation.DBS

(summary)

Seru production is one of the latest manufacturing modes arising from Japanese production practice. *Seru* can achieve efficiency, flexibility, and responsiveness simultaneously. To accommodate the current business environment with volatile demands and fierce competitions, *seru* has attracted more and more attention both from researchers and practitioners. A new planning management system, just-in-time organization system (JIT-OS), is used to manage and control a seru production system. The JIT-OS contains two decisions: *seru* formation and seru loading. By *seru* formation, a *seru* system with one or multiple appropriate *serus* is configured; by *seru* loading, customer ordered products are allocated to *serus* to implement production plans. In the process of seru formation, workers have to be assigned to serus. In this paper, a *seru* loading problem with worker assignment is constructed as a bi-level programming model, and the worker assignment on the upper level is to minimize total idle time while the lower level is to minimize the makespan by finding out optimal product allocation. A product lot can be splitted and allocated to different *serus*. The problem of this paper is shown to be NP-hard. Therefore, a simulated annealing and genetic algorithm (SA-GA) is developed. The SA is for the upper level programming and the GA is for the lower level programming. The practicality and effectiveness of the model and algorithm are verified by two numerical examples, and the results show that the SA-GA algorithm has good scalability.

Keywords: Assembly systems

Manuscript submitted to

SIMULATED ANNEALING AND GENETIC ALGORITHM BASED METHOD FOR A BI-LEVEL *SERU* LOADING PROBLEM WITH WORKER ASSIGNMENT IN *SERU* PRODUCTION SYSTEMS

LAN LUO AND ZHE ZHANG*

School Economics and Management, Nanjing University of Science and Technology
Nanjing 210094, P. R. China

YONG YIN

Graduate School of Business, Doshisha University
Karasuma-Imadegawa, Kamigyo-ku, Kyoto, 602-8580, Japan

ABSTRACT. *Seru* production is one of the latest manufacturing modes arising from Japanese production practice. *Seru* can achieve efficiency, flexibility, and responsiveness simultaneously. To accommodate the current business environment with volatile demands and fierce competitions, *seru* has attracted more and more attention both from researchers and practitioners. A new planning management system, just-in-time organization system (JIT-OS), is used to manage and control a *seru* production system. The JIT-OS contains two decisions: *seru* formation and *seru* loading. By *seru* formation, a *seru* system with one or multiple appropriate *serus* is configured; by *seru* loading, customer ordered products are allocated to *serus* to implement production plans. In the process of *seru* formation, workers have to be assigned to *serus*. In this paper, a *seru* loading problem with worker assignment is constructed as a bi-level programming model, and the worker assignment on the upper level is to minimize total idle time while the lower level is to minimize the makespan by finding out optimal product allocation. A product lot can be splitted and allocated to different *serus*. The problem of this paper is shown to be NP-hard. Therefore, a simulated annealing and genetic algorithm (SA-GA) is developed. The SA is for the upper level programming and the GA is for the lower level programming. The practicality and effectiveness of the model and algorithm are verified by two numerical examples, and the results show that the SA-GA algorithm has good scalability.

1. **Introduction.** In the past three decades, volatile and diversified demands, high capital, and labor cost as well as the rapid revolution in technology have posed great challenges to manufacturing industries, especially to the high-tech like electronics. Such tough environment motivates enterprises to modify traditional production systems with high automated equipment and low worker involvement and to create

---

2010 *Mathematics Subject Classification.* Primary: 58F15, 58F17; Secondary: 53C35.

Key words and phrases. *Seru* loading, bi-level programming, lot-splitting, worker assignment, hybrid intelligent algorithm.

* Corresponding author: Zhe Zhang. This research was sponsored by Natural Science Foundation of China (NSFC Grant no. 71401075). We thank Professor Xiuli Wang and Ding Zhang for their valuable discussions and comments, and we would like to express our appreciation to all the reviewers and editors who contributed to this research.

new production modes. In 1992, under the background of the economic crisis in Japan since 1990s, a new production system named *seru* was created at a factory in Sony. Later on, many manufacturing enterprises such as Canon, Sharp, Panasonic, etc. also adopted *seru* production systems (Liu et al., 2014 [25]). Taking Canon as an example, the company dismantled assembly lines of its 54 factories with a total length of about 20,000 meters and replaced them with *serus*. This replacement saved 720,000 square meters of workshop space and reduced the average working time of WIP from 3 days to 6 hours (D&M Nikkei Mechanical, 2003 [6]), and the total cost was reduced by 230 million yen (Weekly Toyo Keizen, 2003 [45]) From 1999 to 2005, its subsidiary, Canon Electronics Corporation, increased profit from 1.1 billion yen to 11.8 billion yen, reduced factory space by 70%, reduced energy demand (water, electricity, etc.) and carbon dioxide emissions by 50% and increased average productivity by 4 times (Hisashi Sakamaki, 2006 [11]; Yu and Tang, 2019 [58]). Yin et al. (2017) provided more detailed data on the benefits of implementing *seru* production in Canon and Sony [51]. Researches show that *seru* is more adaptive and competitive in unpredictable environment with multiple product models, fluctuated volumes, and short product life cycles (Sakamaki, 2006 [36]; Zhang et al., 2017 [59]). *Seru* is considered as a potential production system for Industry 4.0 (Yin et al., 2018 [52]).

Seru production system is a relatively new production mode arising from Japanese production practice (Yin et al., 2017 [51]), the research on *seru* is still few because of its short history. Recently, *seru* has attracted attention from leading scholars. For example, Roth et al. (2016) summed up the development of operations management over the past 25 years, pointing out 8 possible future research directions. They listed *seru* as one of the new research fields worthy of attention, i.e., "*seru* production systems are more flexible than Toyota production system, and they represent the next generation of lean production that has recently been introduced to operations" [34]. Similarly, Treville mentioned that although there is little research on *seru* production systems outside Japan, its progressiveness has been verified in practice. By applying *seru* production, some Japanese electronics enterprises have been able to respond quickly to market demands and confront with rapid development and upgrading of products (Treville et al., 2017 [5]). Although *seru* production mode has been successfully implemented in many manufacturing enterprises, the academic research on it is not sufficient enough to guide production practice. In this paper, we take a *seru* loading problem with worker assignment and lot-sppliting into consideration, and hopefully this research could promote the practice of *seru* production.

Seru is the Japanese pronunciation for cell. *Seru* is defined as a low automated assembly system that is converted from traditional conveyor line. A *seru* system consists one or more *serus*. A *seru* consists of simple equipment and multi-skilled workers to complete one or more product types (Yin, Stecke and Kaku, 2008 [48]; Stecke et al., 2012 [39]). The difference between *seru* production and celluar manufacturing can be found in Yin et al. (2018) [52], Sakazume (2005) [35], and Liu et al. (2010) [24]. They stated that these two types of production systems are similar in layout such as U-sharped. Manufacturing cells are converted from functional layout job shops into product layout flow shops using group technology. *Seru* is a conversion of conveyor lines. Cells are mainly used in machining processes but *seru* is mainly adapted in the assembly process. Yin et al. (2017) [51] added that *serus* are reconfigurable while cells are usually fixed.

There are three types of *seru*, namely divisional *seru*, rotating *seru* and *yatai* (Akino, 1997 [2]), which reveal the evolution of *seru* production system, see Fig. 1. A divisional *seru* is configured at the first stage when converting a line into several *serus*. As *seru* implementation and worker training carry on, the workers possess the whole skill required for assembly, rotating *serus* can be constructed. The equipment is shared by several fully skilled workers in rotating *serus*, they move from one workstation to another to complete all the operations from start to end one by one in each *seru*. The operations on a single product are completed by a fully skilled worker and not shared with other workers. When a product is finished, the worker returns to the first workstation and begins a new round (Liu et al., 2014 [25]). The efficiency of each *seru* is mainly decided by the slowest worker in a rotating *seru*, thus causing waste in processing time. Sometimes, a supervisor is required to take charge of managerial tasks in rotating *serus*. The benefit of rotating *seru* is its flexibility to fluctuated volumes. The high the production volume, the more the number of workers within a rotating seru. On the contrary, the low the production volume, the less the number of workers are assigned. Some rotating *serus* can finally evolve into *yatais*. A *yatai* is a highly self-disciplined *seru* that contains only one fully cross-trained worker who takes full charge of the processing procedure (Liu et al., 2010 [24]). There are hybrid *seru* systems in which different *seru* types and conveyor lines exist (Iwamuro, 2004 [15]; Miyake, 2006 [32]). Researches on *seru* production usually focus on specific *seru* types and/or hybrid *seru* systems. Liu et al. (2014) [23] discussed the production planning problem in multi-stage multi-option *seru* systems. Yu et al. (2017) [56] took line-hybrid *seru* conversion problem into account. Some research on the practice of *seru* production is ongoing, including *seru*-line conversion (Shao et al., 2016 [37]; Yu et al., 2017 [55]; Aboelfotoh et al., 2018 [1]; Zhang et al., 2019 [62]), *seru* formulation (Yu et al., 2018 [57]; Wang and Tang, 2018 [46]) and reliability of *seru* systems (Han et al., 2018 [7]; Han et al., 2019 [8]). This paper will focus on the *seru* loading problem with worker assignment in the context of rotating *serus*.

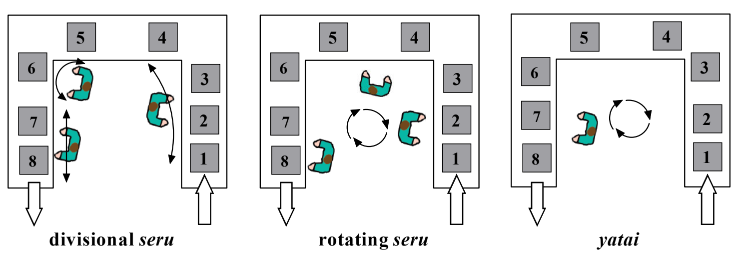

FIGURE 1. Three types *seru*

A new planning system, JIT-OS, is used to manage and control a *seru* production system. The industrial cases of JIT-OS can be found in Yin et al. (2008) [48] and Stecke et al. (2012) [39]. JIT-OS is an extension, or upgrade, of traditional JIT material system (JIT-MS). Their mechanisms are similar: the correct materials/*serus*, in the right place, at the appropriate time, in the exact amount. The difference is the focus from materials to organizations (i.e., *serus*). According to Stecke et al. (2012) [39], the application of JIT-OS is as follows. First, apply the principles of the correct *serus*, in the right place, at the appropriate time for product model changes. This involves the relocation or relayout of either current *serus* or the creation of new *serus* for both new models or model changes. Second, determine the appropriate number of *serus* and/or number of workers within *serus* to handle production volume fluctuation. The JIT-OS contains two decisions: *seru* formation and *seru* loading. By *seru* formation, a *seru* system with one or multiple appropriate *serus* is configured; by *seru* loading, customer ordered products are allocated to *serus* to implement production plans. In the process of *seru* formation, workers have to be assigned to *serus*. In this paper, a *seru* loading problem with worker assignment is constructed as a bi-level programming model.

Seru loading is included in production planning problem, which plays an essential role in deciding the performance of a *seru* system. It contains two aspects: one is to assign products to feasible *serus* (or assign feasible *serus* to products), and the other is to give a preliminary order for products in each *seru* (Süer and Dagli, 2005 [40]). Tao et al. (2010) [43] developed a semi-online algorithm for scheduling problem with bounded processing time. Yin et al. (2013) [50] studied scheduling problem with past-sequence-dependent delivery times. Zhao et al. (2014) [63] considered scheduling and assignment problem with rejection and position dependent processing time. Lot-splitting and setup time are two main factors when making *seru* loading decision. On one hand, parallel production may reduce total makespan and increase the flexibility of the whole *seru* system. On the other hand, too small lot size brings about frequent setups which may in turn cause an increase in makespan. Lot-splitting means the demand of a product can be divided and the product can be allocated to more than one *seru*, it is usually neglected in researches about *seru* on account of complexity. However, other researches have been carried on with lot-splitting in the scenarios of jobshop, CM and traditional conveyor line. Low et al. (2004) [27] studied the benefits of lot-splitting in job-shop production system. Huang and Yu (2017) [14] designed an ant colony algorithm for solving multi-objective job-shop scheduling problem with equal-size lot-splitting. What's more, setup time is another deciding factor in loading problem which is usually considered to be zero in *seru* context. Some researches under other production modes provide us with similar understandings. Hsu et al. (2010) [13] regarded setup time to be proportional to the length of the already processed jobs. Luo et al. (2015) [28] solved scheduling problem for hybrid flowshop with family setup time. Pei et al. (2017) [33] considered time-dependent setup time which is a liner function of starting time. Luo et al. (2017) [29] estimated setup time by the number of different operations between two adjacent products in one *seru*. Apart from these two main factors above, assignment of multi-skilled workers are the key factor that deciding the performance of *seru* production system. Babayigit and Süer (2003) [3] considered minimizing tardy jobs with limited manpower. Liu et al. (2013) [26] investigated training and assignment problem of workers with the aim of balance workers' workload. Ying and Tsai (2017) [53] worked on training and assigning multi-skilled workers in *seru* system to minimize total cost. Lian et al. (2018) [22] considered a multi-skilled worker assignment problem with worker heterogeneity in *seru* systems. Sun et al. (2019) [41] developed a cooperative coevolution algorithm which combining generic algorithm and local search, and the ant colony optimization algorithm for solving *seru* formulation and *seru* scheduling problem at the same time with the objective of minimizing makespan. Sun et al. (2019) [42] used a cooperative coevolution algorithm to solve *seru* production problem aim at reducing the total tardiness. Although the researches above took worker assignment into account, few researches has considered the hierarchy of worker assignment decision and production allocation decision. In that case, the *seru* loading problem with worker assignment in this paper is abstracted as a bi-level programming model.

Because of the complexity of practical production decision-making problems which may be in the charge of different hierarchical decision makers, bi-level programming has been widely utilized in the field of engineering optimization. There are mainly two aspects of methods for solving bi-level programming, one is analytical methods, and the other is heuristics (Sinha et al. 2018) [38]. Due to the non-convexity and non-differentiability of bi-level programming, the heuristics including meta-heuristic methods are widely used. Kasemset and Kachitvichyanukul (2010) [19] built a bi-level multi-objective model under Theory of Constraints (TOC) for job shop scheduling. Kasemset and Kachitvichyanukul (2012) [20] developed a PSO-based procedure for bi-level job-shop scheduling problem. Yang et al. (2013) [47] designed an electromagnetism-like optimization algorithm for bi-level programming problems. Han et al. (2013) [9] proposed a bi-level model for scheduling problem with lot-splitting in virtual cell manufacturing system. Zhang et al. (2014) [61] introduced bi-level programming with MOBL-APSO algorithm to resource-constrained multiple project scheduling problems in hydropower station. Behnia et al. (2017) [4] built a bi-level mathematical model for cell formulation problem considering workers' interest. Zhang and Xu (2016) [60] developed a bi-level multi-objective MRCPSP (multi-mode resource-constrained projects scheduling problem) model with fuzzy random coefficients and bi-random coefficients. In this paper, we will design a simulated annealing and genetic algorithm (SA-GA) to solve the proposed bi-level seru loading model, where SA is for upper level programming and the GA is for lower level programming.

The remainder of this paper is organized as follows: a detailed description for the *seru* loading problem with worker assignment is presented in Section 2. Then in Section 3, a bi-level model is formulated based on the problem description above. To solve this NP-hard problem, a simulated annealing and genetic algorithm (SA-GA) is designed in Section 4. Section 5 provides two numerical examples to test the effectiveness of the proposed model and algorithm. The conclusions and future research are finally shown in Section 6.

2. Problem description.

2.1. *Seru* loading problem with worker assignment.

In this section, a *seru* loading problem with worker assignment will be discussed. Since multi-skilled worker is the core of *seru* production system, so its assignment is one of the most important issues. Normally, when the orders are received, the worker assignment decision is firstly made by the first-line managers. Then, the *serus* are constructed and the ability of each *seru* (whether a product can be produced and its producing time) is also determined. According to this worker assignment, the product allocation decision is made by the production planning department. Subsequently, the production planning department feedbacks his result to the first-line managers to check it is satisfied or not. If it is not, the process above will be repeated until the optimal worker assignment and loading results are obtained. In this decision making process, although the first-line managers cannot make product allocation decision of lower level directly, they can guide the decision. Besides, the product allocation decision could in turn affects worker allocation on the upper level. Hence, the output of worker assignment in the upper level is the input of product allocation in the lower level. Meanwhile, the output of lower level is also the input of the upper level. Therefore, the *seru* loading problem in this paper is formulated as a bi-level problem, see Fig. 2.

With regard to worker assignment, the workers' processing time for each product are known and which *seru* the workers should be assigned to is needed to be decided. The *serus* discussed in this paper are all rotating *serus*, the divisional *serus* and *yatais* are left for the future research. In a rotating *seru*, the workers are in charge of the whole processing procedures of a single product but the equipments are shared with each other. They move along with the materials from one workstation to another to complete the processing procedures. The ability of *serus* are decided by workers assigned to them, and if at least one worker in the seru is capable of processing a product, then the *seru* can process this product. And the efficiency of each *seru* is mainly decided by the slowest worker in the *seru*, and the other workers in the *seru* should stop and wait for the slowest one. Thus, the waiting time is their idle time. The aim of worker assignment is to balance the ability of workers in each *seru*, i.e. to minimize the total idle time of the workers.

When it comes to *seru* loading, the demands of each product are known, and which *seru* the product should be allocated to along with the allocation quantity is to be decided. The products are sorted by due dates in advance, so that the products with the earlier due date will be prioritized. Lot-splitting is allowed in this paper to ensure the solutions be more balanced and flexible. The aim of product allocation is to minimize the makespan. Makespan means the maximum processing time of all the *serus* in the whole *seru* production system. The demands of the products should be satisfied and there is an upper limit of processing time for each *seru*. Except for the objective which is to minimize the maximum of several values, the constraints of product allocation present the characteristics of transportation problem.

2.2. **Assumptions.** The following assumptions are made to formulate the bi-level programming model of *seru* loading problem with worker assignment:

(1) The *seru* system has already been configured, and the reconfiguration is not taken into consideration.
(2) Each product can be produced in at least one *seru*.
(3) The demand of each product have already been known.
(4) The manpower assignment is decided before product allocation.
(5) The raw materials are already in place, so the product can be assembled immediately after setup.
(6) The *serus* are *kanketsu*, i.e. all the procedures required for producing a product can be completed within the assigned *serus*.

2.3. **Notation.** *Indices*

i	*seru* index	$i = 1, 2, \cdots, I$
j	product index	$j = 1, 2, \cdots, J$
w	worker index	$w = 1, 2 \cdots, W$

Parameters

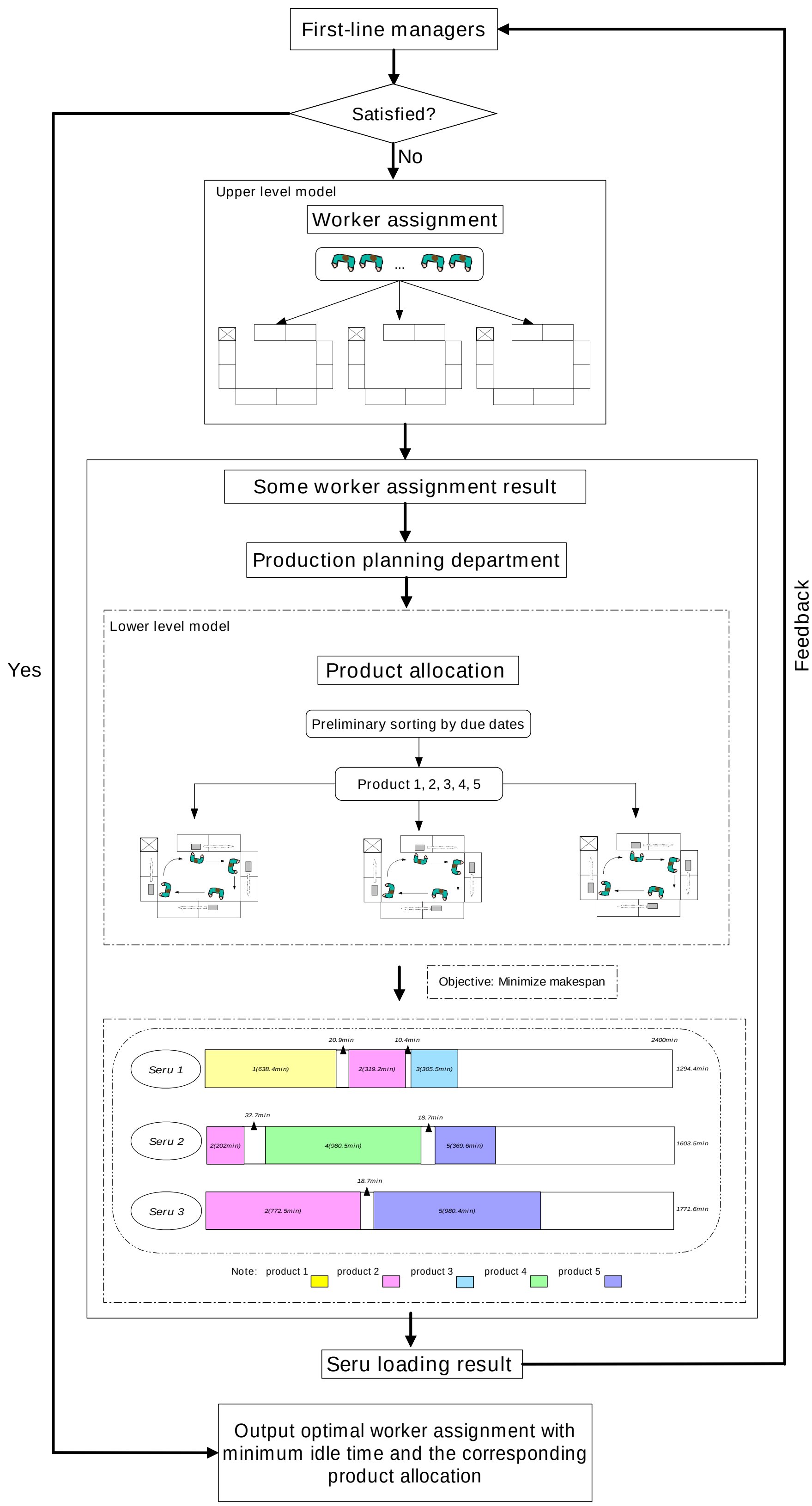

FIGURE 2. Whole bi-level decision procedure

Q_j the demand for product j
s_j setup time before producing product j
t_j^w worker w's producing time of product j
t_{ij} the processing time of product j in *seru* i
T_i the maximum producing time of *seru* i in this period
W_i^l the minmum number of workers can be assigned to *seru* i
W_i^u the maximum number of workers can be assigned to *seru* i
$e_w^j = \begin{cases} 1, & \text{if worker } w \text{ can assemble product } j, \\ 0, & \text{otherwise.} \end{cases}$

Decision variables
$y_i^w = \begin{cases} 1, & \text{if worker } w \text{ is assigned to } seru\ i, \\ 0, & \text{otherwise.} \end{cases}$

$f_{ij} = \begin{cases} 1, & \text{if } seru\ i \text{ can process product } j, \\ 0, & \text{otherwise.} \end{cases}$

$x_{ij} = \begin{cases} 1, & \text{if product } j \text{ is allocated to } seru\ i, \\ 0, & \text{otherwise.} \end{cases}$

Q_{ij} the quantity of product j being assigned to *seru* i

There are some connections among these variables:
The efficiency of a rotating *seru* is decided by the slowest worker, i.e.:

$$t_{ij} = max_{\{w=1,\cdots,W\}}\{y_i^w t_j^w\}$$

If worker w can assemble product j, then his/her processing time for product j cannot be zero, i.e.:

$$e_w^j = \begin{cases} 1, & t_j^w \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

If one of the worker in *seru* i can produce product j, then the *seru* can produce product j, i.e.:

$$f_{ij} = max_{\{w,\cdots,W\}}\{y_i^w e_w^j\}$$

Besides, the total number of workers that can produce product j in *seru* i is:

$$\sum_{w=1}^{W} y_i^w e_w^j$$

If product j is allocated to *seru* i, then the allocated quantity cannot be zero, i.e.:

$$x_{ij} = \begin{cases} 1, & Q_{ij} \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, let $v_{ij} = \begin{cases} 1, & \sum_{j'=1}^{j-1} x_{ij'} = 0 \\ 0, & \text{otherwise.} \end{cases}$, then $v_{ij}x_{ij} = 1$ means that product j will be firstly produced in *seru* i.

3. **Modeling.**

3.1. **Upper level model.** The objective of upper level model is to minimize the total idle time of all the workers in this *seru* production system, and it is the waiting time of all the workers on average workload in their assigned *seru*, i.e.:

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{w=1}^{W} (t_{ij} - t_j^w) y_i^w \frac{Q_{ij}}{\sum_{w=1}^{W} y_i^w e_w^j} \tag{1}$$

Since a worker can only be allocated to one *seru*, thus:

$$\sum_{i=1}^{I} y_i^w = 1 \qquad \forall w \tag{2}$$

Because of the space and equipment limits as well as the increase in management difficulties as workers adding, the number of workers in each *seru* is supposed not to exceed the maximum number. What's more, the minimum number of workers in each *seru* should be guaranteed in case that the workload of workers in *serus* which have small number of workers is too heavy, i.e.:

$$W_i^l \leq \sum_{w=1}^{W} y_i^w \leq W_i^u \qquad \forall i \tag{3}$$

Besides, there are also some logical constraints in upper level model:

$$y_i^w,\ f_{ij} \in \{0,1\} \tag{4}$$

3.2. **Lower level model.** The objective of lower level model is to minimize the total makespan, which is the finishing time of the latest product in the whole system, i.e.:

$$\min\ MS = max_{\{i\in\{1,\cdots,I\}\}} \sum_{j=1}^{J} \left(\frac{Q_{ij}t_{ij}}{\sum_{w=1}^{W} y_i^w e_w^j} + s_j x_{ij} v_{ij} \right) \tag{5}$$

Because a product can only be assigned to feasible *serus*, so:

$$x_{ij} \leq f_{ij} \qquad \forall i, j \tag{6}$$

And, the total quantity of demand should be fulfilled when splitting into lots:

$$\sum_{i}^{I} Q_{ij} = Q_j \qquad \forall j \tag{7}$$

In addition, the total producing time of each *seru* cannot exceed its available time in the period,i.e.:

$$\sum_{j=1}^{J} \left(\frac{Q_{ij}t_{ij}}{\sum_{w=1}^{W} y_i^w e_w^j} + s_j x_{ij} \right) \leq T_i \tag{8}$$

Finally, there are also some logical constraints in lower model, i.e.:

$$Q_{ij} \geq 0 \tag{9}$$

To sum up, the bi-level programming model formulated for the *seru* loading problem in this paper is as follows:

$$
\left\{
\begin{array}{ll}
\min \ \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{w=1}^{W}(t_{ij}-t_j^w)y_i^w\frac{Q_{ij}}{\sum_{w=1}^{W}y_i^w e_w^j} & \\
s.t. & \left\{
\begin{array}{l}
\sum_{i=1}^{I} y_i^w = 1, \quad \forall w\\
W_i^l \le \sum_{w=1}^{W} y_i^w \le W_i^u, \quad \forall i\\
t_{ij} = max_{\{w=1,\cdots,W\}}\{y_i^w t_j^w\}, \quad \forall i,j\\
f_{ij} = max_{\{w=1,\cdots,W\}}\{y_i^w e_w^j\}, \quad \forall i,j\\
e_w^j = \left\{\begin{array}{ll} 1, & t_j^w \neq 0\\ 0, & \text{otherwise.}\end{array}\right.\\
y_i^w,\ f_{ij} \in \{0,1\}\\
i=1,2,\cdots,I; j=1,2,\cdots,J; w=1,2,\cdots,W;\\
\min \ MS = max_{\{i\in\{1,\cdots,I\}\}}\sum_{j=1}^{J}\left(\frac{Q_{ij}t_{ij}}{\sum_{w=1}^{W}y_i^w e_w^j}+s_j x_{ij} v_{ij}\right)\\
s.t. \quad \left\{
\begin{array}{l}
x_{ij} \le f_{ij}, \quad \forall i,j\\
\sum_{i=1}^{I} Q_{ij} = Q_j, \quad \forall j\\
\sum_{j=1}^{J}\left(\frac{Q_{ij}t_{ij}}{\sum_{w=1}^{W}y_i^w e_w^j}+s_j x_{ij}\right) \le T_i, \quad \forall i\\
x_{ij} = \left\{\begin{array}{ll} 1, & Q_{ij} \neq 0\\ 0, & \text{otherwise.}\end{array}\right.\\
Q_{ij} \ge 0\\
i=1,2,\cdots,I; j=1,2,\cdots,J; w=1,2,\cdots,W
\end{array}\right.
\end{array}\right.
\end{array}
\right. \tag{10}
$$

4. **Bi-level simulated annealing and genetic algorithm (SA-GA).** Bi-level programming problem has been proven to be NP-hard (Jeroslow, 1985 [16]; Hansen, Jaumard and Savard, 1992 [10]; Vicente, Savard and Judice, 1994 [44]). What's more, the upper level programming of worker assignment has been proven by Yu et al. (2014) [54] as an exact cover problem which is one of the Karp's 21 NP-complete problems (Karp, 1972 [18]) . Similarly, the *seru* loading problem has also been proven to be NP-hard by Yin et al. (2011) [49]. Therefore, it can be concluded that the bi-level *seru* loading problem with worker assignment is an NP-hard problem. Considering the complexity of bi-level programming, especially with mixed integer variables and minimax objective in lower level programming, a non-numerical stochastic method i.e. SA-GA algorithm is designed to solve this problem. In this algorithm, the SA is dedicated for the upper level programming because its ability to jump out of the local optimum due to Metropolis rule, while GA is for lower level programming due to its robustness and global optimization. In the process of finding the optimal solution, when a particular worker assignment is decided in upper level programming, then the GA is used to find out the optimal product allocation for this particular worker assignment. The outline of the SA-GA algorithm which can briefly show the connection between the upper and lower programming is presented in Fig. 3.

4.1. **Simulated annealing for the upper level programming.** Simulated annealing (SA) is firstly introduced to combinatorial optimization by Kirk- patrick,

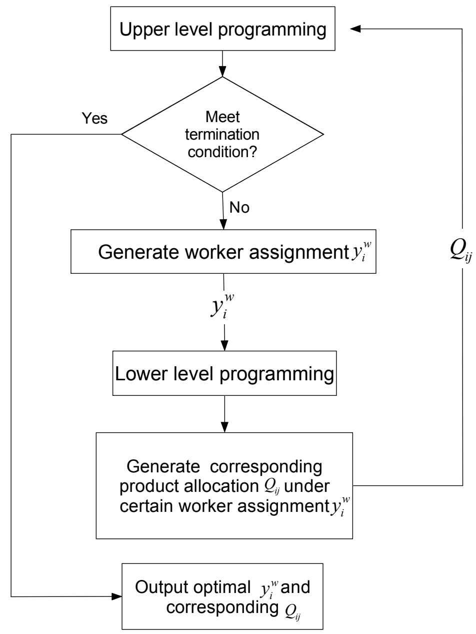

FIGURE 3. The outline of SA-GA algorithm

Gelatt and Vecchi (1983) [21]. The simulation of annealing procedure in solids provides a new method for solving complex problems with large number of variables. Compared with hill-climbing method, SA provides a mechanism which is called Metropolis for inferior solution being accepted. The Metropolis procedure, proposed by Metropolis et al. (1953) [30], is used to generate a set of states under a certain temperature. Take minimization problem for example, let ΔE be the changing quantity of evaluation i.e. $\Delta E = E_{new} - E_{old}$. If $\Delta E \leq 0$, the old state will be replaced by the new state. In other case, when $\Delta E > 0$, the new state will be accepted by the probability of

$$P = exp(-\Delta E/T) \tag{11}$$

It can be concluded from Eq. (11) that the acceptance is close to 1 when the temperature is at a higher level in the beginning. SA in this stage is similar to simple random search. As the iteration goes on, the temperature drops, as well as the acceptance of inferior solution. At the end of the iteration, the temperature is at a very low level and the acceptance of inferior solution is close to 0, which makes the SA has the same effect as iterated hill-climbing methods. In that case, it can be quickly convergent. SA is a local research method, but the Metropolis rule makes it possible to jump out of the local optimum. Considering the global optimization and rapid convergence SA has, it is utilized in this paper for upper level programming.

In this paper, a kind of permutation encoding method is used to represent the solution. A chromosome is combined with W genes which correspond to W workers. Thus, the position of a gene represents the worker's number, and the value of a gene means the *seru*'s number which the worker is assigned to. By this way of encoding, the constraint in Eq. (2) which let a worker can only be assigned to one *seru* will always be satisfied. An indicator variable y^w to represent the value of a gene is defined as follows:

$$y^w = i \quad \text{when} \quad y_i^w = 1, \forall i, w \tag{12}$$

A specific example of the encoding method is shown in Fig. 4. The numbers in the first row which note the locations represent the number of a certain worker, and the number (i.e. y^w) in the second row is the *seru* where the worker is assigned. For instance, in Fig. 4 the first worker is assigned to *seru* 1 , the second worker is assigned to *seru* 2 and the third worker is assigned to *seru* 2, etc. The encoding vector in programming is the second row.

worker (location)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
seru	1	2	2	1	1	1	3	2	3	1	3	2	2	3	2

FIGURE 4. An example of SA encoding

The procedure of SA for upper level programming is summarized as follows:

Step 1: Initialize T_max, T_min, II, α, and set initial iteration $T = T\_max$, $t = 1$;
Step 2: Generate initial random solution of worker assignment;
Step 3: Check constraints and repair the solution;
Step 4: Solve lower level programming by GA and get the best product allocation quantities;
Step 5: Evaluate the solution by the objective of total idle time;
Step 6: Set initial iteration $ii = 1$;
Step 7: Generate a random disturbance to the solution;
Step 8: Repeat **Step 3-5**;
Step 9: If $eval\_new < eval\_old$, receive the new solution; else receive the new solution by Metropolis rule, $ii = ii + 1$;
Step 10: If $ii \leq II$, then go to **Step 7**, else go to **Step 11**;
Step 11: Let $T = \alpha T$, $t = t+1$, If $T \geq T\_min$, go to **Step 6**, else output the best solution of worker assignment and its idle time, as well as the corresponding product allocation quantities and makespan obtained by GA in lower level programming.

Besides, the detailed procedure for generating initial solution and repairing is presented by the following pseudo codes in Algorithm 1, and the way of generating a random disturbance to the current solution is shown in Algorithm 2. The pseudo codes are based on MATLAB. And solving lower level programming by GA in step 4 will be explained in detail in the following section.

4.2. **Genetic algorithm for the lower level programming.** Since being proposed by Holland in 1975 [12], the genetic algorithm (GA) has been widely used in engineering optimization problem. GA is inspired by the natural evolution procedure, it starts from an initial population of random solutions. The offspring population is generated from the parent population by a series of genetic operations i.e. selection, crossover and mutation. In each generation, the population consists of a certain number of individuals, and each individual corresponds to a potential solution. The individuals will be evaluated by a certain fitness function. After obtaining the fitness value of each individual, the parents will be selected. The individuals with higher fitness value are more likely to be chosen, according to the assumption that better parents will generate better offspring. Hopefully the

Algorithm 1: Initialize and repair

```
Input: seru, worker, worker_uplim, worker_lowlim
Output: worker_seru_row
worker_seru_row = fix(rand(1, worker) * seru) + 1;
seru_worker = zeros(seru, worker_uplim);
for w = 1 : worker do
    if the number of assigned workers in seru: worker_seru_row(w) is less
    than worker_uplim then
        assign worker w to the seru;
    else
        while worker w has not been assigned do
            a = fix(rand * seru) + 1;
            if the number of workers in seru a is less than worker_uplim then
                assign worker w to seru a;
            end
        end
    end
    while the seru where the number of workers is less than worker_lowlim
    exists do
        find the first unsatisfied seru noted as c;
        b = fix(rand * seru) + 1;
        if b ≠ c and the number of workers in seru b meets upper and lower
        limits then
            assign the worker with maximum number in seru b to seru c;
        end
    end
end
for w=1:worker do
    [worker_seru_row(w), no_use] = find(seru_worker == w);
end
```

Algorithm 2: Generate disturbance

```
Input: seru, worker, worker_seru_row_old
Output: worker_seru_row_new
worker_seru_row_dis = worker_seru_row_old;
k = fix(rand * worker) + 1;
if worker_seru_row_dis(k) < seru then
    change worker k to the next seru;
else
    change worker k to seru 1;
end
check constraints and repair worker_seru_row_dis;
worker_seru_row_new = worker_seru_row_dis;
```

algorithm will converge to a best solution which may represent the optimal or suboptimal solution. GA is developed for solving lower level model due to its robustness and global optimization (Jones and Soule, 2016 [17]). The genetic algorithm for lower level model is called repeatedly under different worker assignment results, the robustness of genetic algorithm ensures that the results of product allocation won't be greatly affected by parameters compared with other algorithms. Besides, GA is a global search method, and it can process multiple individuals at the same, which speed up the velocity of finding solutions.

To meet the demand constraint in Eq. (7), allocation ratios are used in this paper to represent the solution by real-number encoding. The genetic operators all work on the allocation ratios. Fig. 5 intuitively shows the real-number encoding way by allocation ratios. The algorithms for initialization and repairing solutions

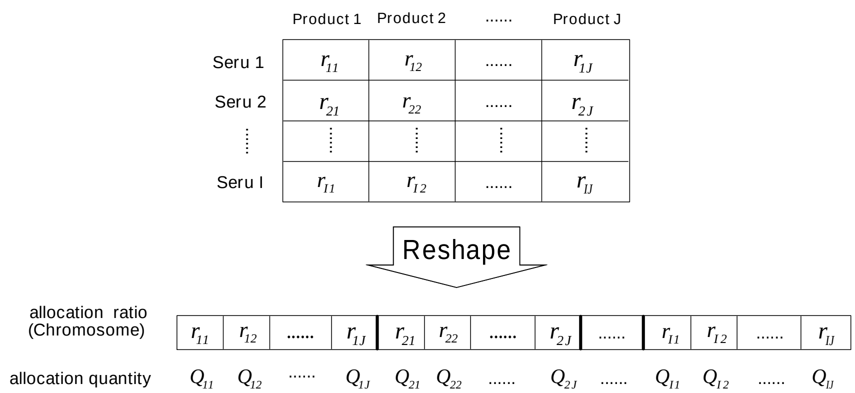

FIGURE 5. The genetic encoding based on allocation ratios

are as follows:

In this paper, a binary tournament selection method is used to select parents. It chooses two chromosomes from the old population every time, the one with higher fitness value will be introduced to the new population, the procedure will be repeated until the number of individuals in the new population meets the population size. After selection, the crossover operator used in the algorithm is arithmetical crossover. Let chromosome $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be the parents, and introducing a random number of λ, then the two children are:

$$\boldsymbol{x}_1^{'} = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \tag{13}$$

$$\boldsymbol{x}_2^{'} = (1-\lambda)\boldsymbol{x}_1 + \lambda \boldsymbol{x}_2 \tag{14}$$

The mutation method used in the GA is nonuniform mutation, it is firstly raised by Michalewicz (1996) [31]. For a chromosome $\boldsymbol{x}$, if x_k is the gene to be mutated, there are two alternative ways of mutation, i.e.

$$x_k^{'} = \begin{cases} x_k + \Delta(g, x_k^U - x_k), & rand < 0.5 \\ x_k - \Delta(g, x_k - x_k^L), & otherwise \end{cases} \tag{15}$$

x_k^U and x_k^L are the upper and lower limits of x_k. In this paper, because the solutions are represented by allocation ratios, the upper limit of allocation ratio is 1, and the lower limit is 0. The function $\Delta(g, y)$ is defined as follows:

$$\Delta(g, y) = y(1 - r^{(1-\frac{g}{G})b}) \tag{16}$$

The parameter g represents the present genetic algebra when mutation occurs, and G notates the maximum genetic algebra. Besides, r is the random number in [0,1] interval while b marks the degree of nonuniformity (in this paper b is set to be 2 as normal). It can be known from the Eq. (16) that at early stage when g is very small, $\Delta(g, y)$ is close to y, such that the x_k can mutate in the whole solution space. While at the late stage, $\Delta(g, y)$ approaches to 0, x_k only mutate in a very small neighbourhood. Therefore, the nonuniform mutation method can avoid prematurity of the GA as well as improve the speed of convergence.

The detailed procedure of GA for lower level programming is as follows:

Step 1: Initialize pop_size, GEN;

Step 2: Generate initial population by allocation ratios;

Step 2.1: Set initial iteration a=1;

Step 2.2: Generate the allocation ratio of product j in *seru* i (r_{ij}) randomly, set $r_{ij} = 0$ under the probability of p_zero1, and normalize r_{ij} by $r_{ij} = \frac{r_{ij}}{\sum_{i=1}^{I} r_{ij}}$;

Step 2.3: Let $a = a + 1$, If $a > pop\_size$, then go to **step 3** , else go to *Step 2.2*;

Step 3: Set initial iteration $g = 1$;

Step 4: Set initial iteration $b = 1$;

Step 5: Repair the solutions to meet the constraints of demands in Eq. (7);

Step 5.1: Calculate the allocation quantities (Q_{ij}) of product j in *seru* i by $Q_{ij} = round(r_{ij} * Q_j)$;

Step 5.2: Adjust the allocation quantity of product j in an allocated *seru* s which is randomly found by $Q_{sj} = Q_j - \sum_{i=1}^{I} Q_{ij} + Q_{sj}$;

Step 6: Evaluate the solutions;

Step 6.1: Calculate the total processing time of each *seru* i: PT_i and the makespan MS is $MS = max_{\{i \cdots I\}} PT_i$;

Step 6.2: If for each *seru*, $PT_i <= T_i$, then $fitness = max(T_i) - MS$, else $fitness = -MS$;

Step 6.3: Let $b = b + 1$, If $b > pop\_size$, then go to **step 7** , else go to **step 5**;

Step 7: Select the parents from the population by binary tournament selection;

Step 6: Generate the child population by arithmetical crossover operator as Eq.(13 - 14);

Step 8: Mutate the chromosomes by nonuniform mutation as Eq.(15 - 16), and set some of the genes to be 0 under the probability of p_zero2;

Step 9: Let $g = g + 1$, If $g \leq GEN$, then go to **Step 4**, else output the best solution of Q_{ij} and its MS.

The probability of p_zero1 and p_zero2 decides the degree of lot-splitting in the *seru* production system.

To sum up, the overall procedure of the whole SA-GA algorithm is show in Fig. 6.

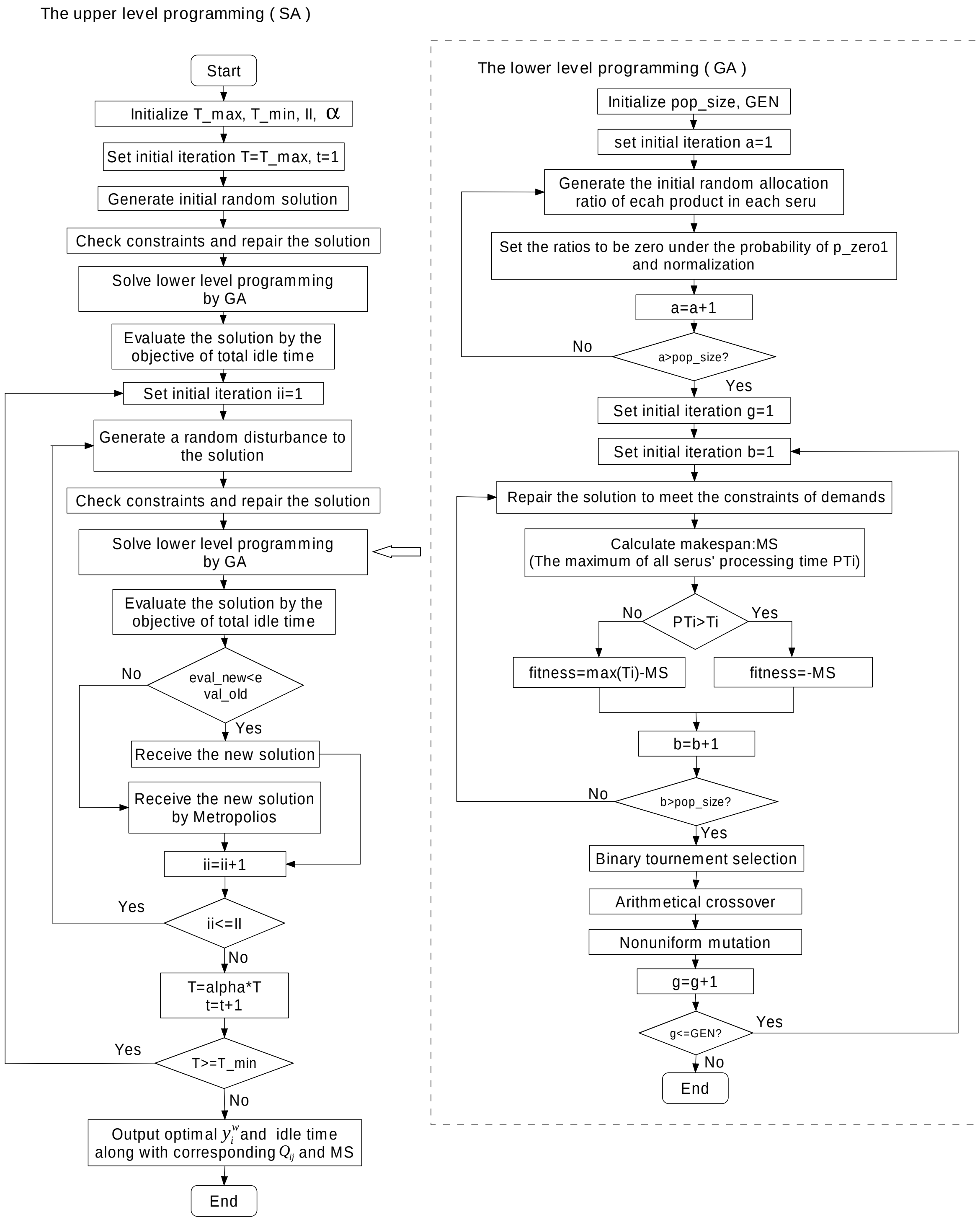

Figure 6. The flowchart of SA-GA algorithm

5. **Numerical examples and analysis.** In this section, numerical examples are presented to verify the practicality and effectiveness of the bi-level model and SA-GA algorithm proposed above. The parameter settings of SA-GA algorithm for solving these two examples are presented in Table 1 based on various tests. The algorithm is implemented by MATLAB R2015b on an Inter Core I5-7200U (basic frequency of 2.5GHz) with 8G memory.

Table 1. The parameter setting of SA-GA algorithm

Level	Algorithm	Parameters	
Upper	SA	$T\_max = 10000$	$T\_min = 0.1$
		$II = 20$	$\alpha = 0.9$
Lower	GA	$pop\_size = 300$	$GEN = 500$
		$p\_zero1 = 0.75$	$p\_zero2 = 0.25$
		$p\_cross = 0.9$	$p\_muta = 0.1$

5.1. **Data collection.** The example in this subsection is about a *seru* production system of 15 workers, and 3 *serus* are to be configured for producing 8 types of products. Besides, the maximum number of workers in a *seru* is 6 and the minimum number is 4. The whole period is of 5 weekdays and each day has 8 working hours, such that the available producing time for each *seru* (T_i) is 2400 minutes. They work from 8:00 to 12:00 in the morning and from 14:00 to 18:00 in the afternoon. Table 2 lists the parameters about products, including each worker's producing time for each product (t_j^w), the demand of each product (Q_j) and the setup time before producing each product (s_j). The number of the products are based on the order of their due dates, and the one with earlier due date has smaller number, which makes it been arranged before the one with later due date in the allocated *seru*.

5.2. **Results and analysis.** Fig. 7 is the minimum idle time in every iteration of SA. It can be seen from the figure that the SA begins to converge around the 100th iteration. Fig. 8 reveals the relationship between the two objectives of the bi-level model. It is clear that there is a high linear correlation between the total idle time and the makespan. In that case, the realization of lower level objective to some extent has a positive effect on the upper level objective. The dots close to the origin have deeper color, which means that the SA-GA algorithm gradually converges to the optimal solution.

Moreover, Fig. 9 and 10 present the two optimal solutions of this problem respectively, the former has minimum idle time but with slightly longer makespan, and the latter is the opposite. Because the model is a bi-level model and the worker assignment is decided in the first place, so the first-line manager is more likely to choose the solution with minimum idle time in Fig. 9. The worker assignment decision with minimum idle time is shown in Fig. 11, and the corresponding product allocation decision made by product planning department is presented in Table 3. Fig. 11 shows that 6 workers are assigned to *seru* 1, 5 workers are assigned to *seru* 2 and 4 workers are assigned to *seru* 3. With the minimum idle time of all the workers, the working pace of the workers in each *seru* is relatively consistent. Besides, according to Table 3, product 5 is allocated to *seru* 1 and *seru* 3, and product 6 is allocated to *seru* 2 and *seru* 3. Therefore, lot-splitting is occurred to product 5 and 6 in this loading decision. Fig. 12 illustrates the loading results intuitively. It can

TABLE 2. Data about products

Product	Worker's processing time (min)															Demand	Setup (min)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
1	23	23	21	22	21	24	22	–	21	24	22	–	24	24	23	95	4
2	–	32	37	32	–	34	37	31	34	31	–	31	36	36	37	100	9
3	41	43	–	–	44	47	42	42	–	41	47	45	44	42	–	130	8
4	29	28	29	28	26	27	26	27	27	28	26	31	31	–	28	105	6
5	17	–	17	16	19	17	–	18	16	16	20	20	18	16	17	120	5
6	42	23	20	33	38	33	27	29	–	34	33	29	30	36	19	145	6
7	–	68	48	63	43	71	49	21	66	59	53	–	–	70	83	50	4
8	14	15	14	20	–	19	19	17	22	19	17	18	–	15	10	115	1

[1] The '–' means that the worker cannot produce the product.

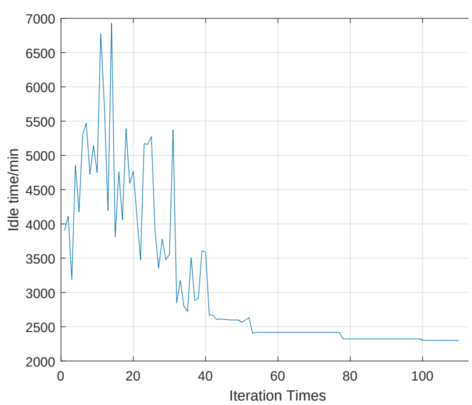

FIGURE 7. The minimum idle time in each iteration of SA

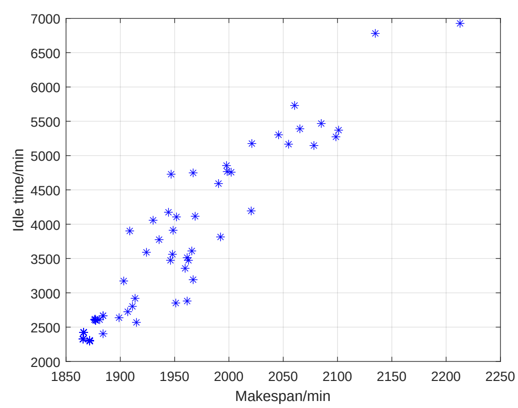

FIGURE 8. Idle time and makespan

be seen from the figure that *seru* 2 has the longest processing time of 1872 minutes among the 3 *serus*, so the makespan of the whole *seru* production system is 1872 minutes. The processing time of *seru* 2 is only 43 minutes longer than *seru* 3 which has the shortest processing time. In that case, the loading of the whole production system is relatively balanced. Table 4 lists the detailed production timetable of each product in this *seru* system. Hopefully, the practical production will be guided by this table. It can be seen from Table 4 that on Friday the *serus* do not have any tasks, so it is possible for the *seru* production system to receive more orders.

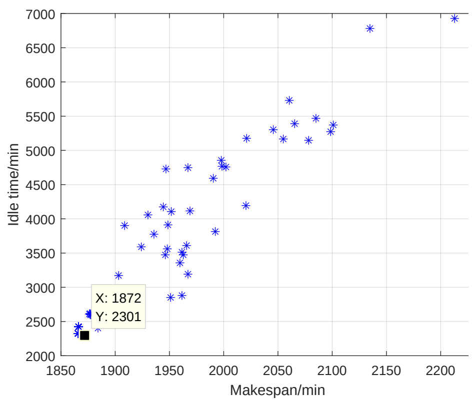

FIGURE 9. Minimum idle time

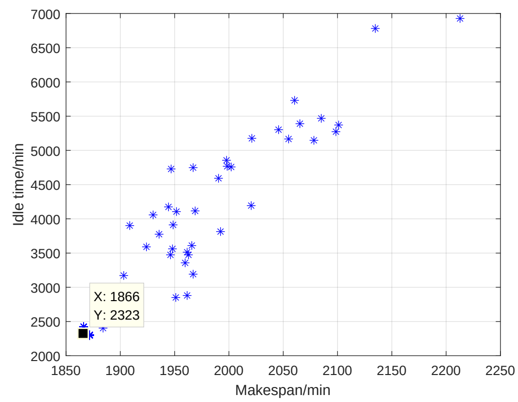

FIGURE 10. Minimum makespan

worker	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Seru	3	2	1	1	3	1	2	2	1	1	3	2	2	1	3

FIGURE 11. The worker assignment decision

TABLE 3. Data about products

Seru \ product	1	2	3	4	5	6	7	8
1	–	100	–	–	80	–	50	115
2	–	–	130	–	–	116	–	–
3	95	–	–	105	40	29	–	–

[1] The '–' means that the product is not allocated to the *seru*.

TABLE 4. Production timetable

Product	1	2	3	4	5
Seru	3	1	2	3	1
Starting time	Monday 8:00	Monday 8:00	Monday 8:00	Tuesday 9:13	Tuesday10:22
Finishing time	Tuesday 9:07	Tuesday 10:17	Wednesday 11:30	Wednesday 15:54	Tuesday 16:09
Product	5	6	6	7	8
Seru	3	2	3	1	1
Starting time	Wednesday 15:59	Wednesday 11:36	Thursday 9:25	Tuesday 16:13	Thursday 8:05
Finishing time	Thursday 9:19	Thursday 17:12	Thursday 16:29	Thursday 8:04	Thursday 17:07

Apart from the detailed result illustrated above, other results of 10 repeated runs for this example are listed in Table 5.

5.3. **Comparison with GA-GA algorithm.** To better analyse the effectiveness of the SA-GA algorithm, a GA-GA algorithm which solve upper and lower level model all by genetic algorithm is designed and the results of the two algorithms are compared.

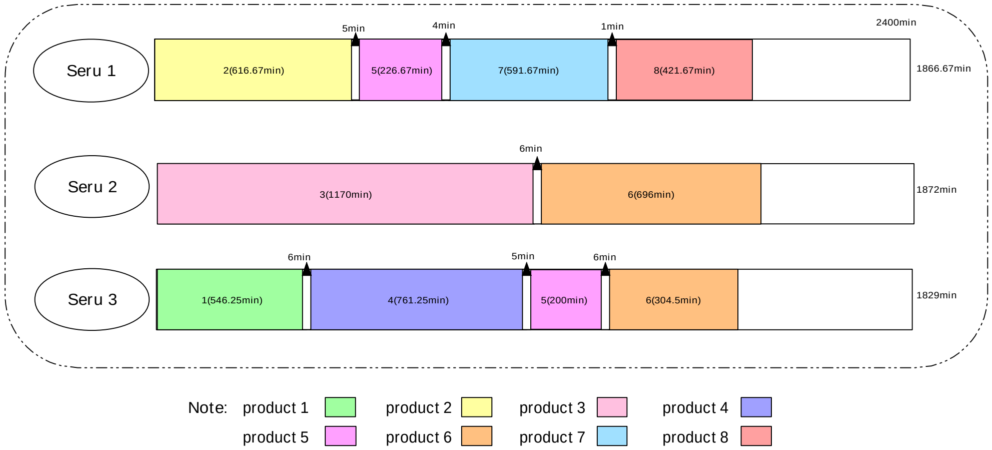

FIGURE 12. Loading results

TABLE 5. Results of the small case

No.	Idle time (min)	Makespan (min)	CPU time (s)
1	2381.7	1910	8242.5
2	2629.4	1907.8	8193.5
3	2438.6	1932.3	8222.2
4	2446	1936	8228
5	2723.1	1933	8228.6
6	2022.3	1893	8064.8
7	2461.2	1895	8095.2
8	2300	1906.3	8098.5
9	2819	1859.5	8051.9
10	2566.8	1874.1	8135
Average	2478.81	1904.7	8156.02
SD	214.16	24.07	70.94

The numerical example used to test the GA-GA algorithm is the same case above, and the parameter of GEN is set to 110 which is the same as the iteration number of SA in this example and the parameter of pop_size is set to be 20 which equals to the number of neighbors in SA for each iteration. After ten runs for GA-GA algorithm, we can get that the average and SD CPU time are 8353.08 (s) and 278.59 (s), where both of them are larger than SA-GA's result. Hence, it can be concluded that SA-GA algorithm has faster calculation than GA-GA. The results of ten runs by GA-GA algorithm are listed in Table 6.

5.4. **Test on the large case.** To show the superiority of SA-GA algorithm, a large case which contains 20 products, 10 *serus* and 50 workers are tested. In this large case, the workers' processing time for each product is listed in Table. 7, and the setup time and demand of each product is shown in Table. 8. The other data is the same as the small case above.

The parameter settings of large case are the same as those of small case in Table 1, which is also based on various tests. The results of five runs are shown in Table

TABLE 6. Results of GA-GA algorithm for small case

No.	Idle time (min)	Makespan (min)	CPU time (s)
1	2519.5	1913.8	8220.8
2	nonconvergent		
3	2384.4	1851.8	8278.8
4	2375.6	1898.3	8230.3
5	nonconvergent		
6	2409.5	1829	8284.8
7	2882	1894.2	9085.2
8	2213.2	1941.3	8244.5
9	2526	1941	8190.9
10	2526.9	1918	8289.3
Average	2479.64	1898.43	8353.08
SD	181.55	37.55	278.59

8, which verified the designed SA-GA algorithm are also adapt to the large case. In addition, the average CPU time only takes 7859.98s more than the small case above, so the scalability of the SA-GA algorithm in this paper are also proved.

6. **Conclusions and further research.** This paper studies a *seru* loading problem with worker assignment. The worker assignment is decided before the product allocation. Considering the hierarchy of decision-making, a bi-level model is proposed for this problem. In the upper level model, a best worker assignment should be decided to minimize the total idle time, while the aim of the lower model is to find best product allocation to minimize the total makespan under the certain worker assignment decided by upper level model. Then, a SA-GA algorithm is designed for solving this model. The SA is for upper level model and GA is for lower level model. The model and SA-GA algorithm are tested by a small case and a large case, and the results show that the SA-GA algorithm has good scalability. Finally, comparison with GA-GA algorithm is also presented to prove the superiority of SA-GA algorithm.

Future research should concentrate on developing more efficient and scalable algorithms. Although SA and GA are all effective meta-heuristic methods for combinatorial optimization problem, the GA will be called repeatedly while solving upper level programming by SA, which contributes to an increase in computation time. Hence, it is of great significance to introduce other mathematical methods for solving this MIP and bi-level problem. The mathematical feature of the bi-level model should be carefully examined for developing the algorithm. Besides, considering the complexity of decision-making in practical production and the conflicts of different decision goals, the multi-objective model should also be introduced to *seru* loading problem, as well as the multilevel programming. What's more, studies on *seru* loading problems in other *seru* types like divisional *serus* and *yatais* should also be carried out. Lastly, software development for real application scenario is supposed to be taken into account based on this research.

REFERENCES

[1] A. Aboelfotoh, G. A. Süer and M. Abdullah, Selection of Assembly Systems; Assembly Lines vs. *Seru* Systems, *Procedia Computer Science*, **140** (2018), 351-358.

TABLE 7. Workers' processing time for each product

Worker	Workers' processing time for each product (min)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	22	39	47	29	–	34	64	23	50	71	20	21	11	32	39	15	20	27	29	24
2	23	37	47	29	20	35	54	21	55	77	24	20	12	30	38	19	21	21	26	20
3	22	38	46	–	22	35	50	26	56	77	24	15	11	34	43	–	17	23	29	21
4	–	37	46	32	–	35	56	–	50	78	22	19	12	34	42	17	21	21	25	23
5	23	40	47	27	–	34	59	20	50	81	–	17	10	34	42	15	17	22	28	–
6	22	38	47	29	23	33	61	25	–	–	21	20	10	32	39	18	17	27	28	21
7	21	37	–	28	19	32	59	–	52	73	23	21	11	30	35	16	22	22	25	19
8	23	38	49	27	18	–	–	22	48	78	21	20	10	31	44	18	–	23	26	22
9	21	40	50	–	21	33	56	26	53	87	–	18	10	32	40	17	22	21	27	20
10	24	–	49	28	19	38	58	26	52	–	24	18	10	37	41	–	17	25	26	23
11	22	38	47	30	18	33	61	27	53	77	21	19	–	33	41	–	19	21	29	23
12	23	36	49	29	21	34	57	22	53	86	21	20	10	33	36	17	21	21	27	24
13	23	39	47	29	22	–	65	21	53	86	22	19	12	30	38	15	19	22	27	21
14	23	39	–	31	18	30	50	29	57	85	24	–	10	37	36	19	21	27	27	21
15	21	36	49	30	–	36	59	24	50	81	24	17	11	37	35	18	19	26	26	22
16	25	36	49	31	19	37	58	22	54	82	24	19	12	39	39	17	18	23	29	22
17	–	37	45	30	22	38	60	23	55	–	21	18	–	37	44	–	21	22	27	–
18	23	37	49	31	21	37	61	–	48	–	21	21	12	34	–	17	17	–	26	20
19	21	35	48	–	19	37	61	28	48	69	20	19	10	33	36	18	22	25	25	20
20	21	38	–	30	23	32	55	29	53	72	22	16	10	–	35	16	17	21	29	23
21	21	36	50	30	22	35	64	29	53	86	22	17	11	39	–	17	18	24	26	23
22	22	38	50	29	–	33	61	22	48	69	23	17	11	40	–	15	–	22	28	21
23	25	39	47	29	19	37	59	26	–	–	24	16	12	39	–	17	22	25	25	23
24	20	37	49	29	22	30	62	22	47	71	21	18	12	40	42	19	21	27	25	21
25	23	39	47	30	21	38	63	–	55	–	23	15	11	31	38	19	22	27	27	24
26	–	37	–	32	23	32	58	28	50	72	24	16	11	32	44	17	19	22	25	–
27	24	37	49	30	22	–	56	30	51	78	24	19	11	34	40	–	19	–	29	21
28	24	39	47	–	18	37	–	24	47	85	23	16	10	39	35	17	22	20	26	20
29	25	–	47	29	19	36	54	20	49	79	24	16	11	35	41	18	–	23	25	–
30	20	37	47	28	22	40	51	22	51	78	–	21	11	37	39	16	18	–	25	–
31	23	38	48	29	19	35	53	20	56	72	22	16	11	–	37	16	20	25	25	20
32	23	36	45	29	–	37	63	29	50	79	20	16	11	37	–	18	21	27	29	25
33	23	38	47	30	23	40	59	26	56	78	23	18	11	41	40	–	21	22	–	23
34	21	35	50	27	23	38	65	22	47	71	24	16	10	38	36	16	20	27	27	24
35	20	35	48	32	21	33	61	25	–	–	21	21	10	38	–	19	18	24	29	19
36	–	38	45	30	19	31	63	24	56	85	23	–	10	41	37	19	19	24	26	21
37	24	36	–	30	22	38	55	24	50	87	23	19	12	–	39	17	20	26	–	25
38	24	39	49	32	21	37	52	–	–	71	24	19	12	35	38	15	19	23	25	22
39	21	39	49	31	19	35	57	29	55	77	21	19	11	40	43	15	19	–	28	19
40	25	35	47	30	20	34	59	25	48	72	23	–	10	41	35	18	20	20	26	18
41	22	36	48	32	20	–	55	25	49	71	23	19	12	31	43	17	19	22	26	24
42	23	39	47	27	19	39	64	24	53	74	24	21	12	32	–	19	18	–	29	22
43	23	36	46	32	20	35	–	21	55	80	21	16	12	32	40	18	20	23	26	22
44	–	–	45	31	22	37	52	–	57	72	22	16	11	37	–	18	22	20	27	24
45	20	40	47	29	21	32	52	29	55	82	21	15	10	33	–	16	20	27	–	19
46	20	40	–	30	20	38	58	23	50	82	23	20	11	31	38	19	–	–	25	19
47	23	39	49	–	21	39	61	25	–	78	24	19	11	38	39	17	22	27	27	25
48	22	38	49	28	18	33	–	25	49	74	23	18	12	30	43	16	20	21	29	–
49	20	39	47	29	21	–	60	22	52	81	23	21	12	30	40	16	20	24	25	19
50	22	–	47	28	20	32	64	27	49	77	–	18	10	33	37	17	20	20	25	24

[1] The '–' means that the worker cannot produce the product.

Table 8. Setup time and demand of products

Product	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Setup time (min)	4	9	8	6	5	6	4	1	10	12	24	2	5	7	11	3	15	4	2	7
Demand	145	107	134	105	140	145	115	87	145	126	125	150	118	106	75	80	132	83	65	89

Table 9. Results of large case

No.	Idle time (min)	Makespan (min)	CPU time (s)
1	5587.3	1932	16236
2	5457.4	1848.30	16376
3	5164.9	1865.4	16221
4	5258.9	1919	15754
5	5757.2	1806	15743
Average	5445.14	1874.14	16066
SD	214.92	46.36	264.84

[2] S. Akino, Internationalization of Japanese company and change of production system, (Japanese), *Rikkyo Economic Review*, **51** (1997), 29-55.

[3] C. Babayigit, G. A. Süer, Cell loading to minimize the number of tardy jobs subject to the manpower restriction, *Proceedings of Group Technology/Cellular Manufacturing Symposium* OH, USA, 2003.

[4] B. Behnia, I. Mahdavi, B. Shirazi, M. M. Paydar, A bi-level mathematical programming for cell formation problem considering workers interest, *International Journal of Industrial Engineering & Production Research*, **28** (2017), 267-277.

[5] S. Treville, M. Ketokivi, V. Singhal, Competitive manufacturing in a high-cost environment: introduction to the special issue, *Journal of Operations Management*, **49-51** (2017), 1-5.

[6] D&M Nikkei Mechanical. The Challenge of Canon-Part 3, *D&M Nikkei Mechanical*, **588** (2003), 70-73.

[7] X. Han, Z. Zhang and Y. Yin, *"Reliability Analysis for a Divisional Seru Production System with Stochastic Capacity,"* 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, (2018), 710-714.

[8] X. Han, Z. Zhang and Y. Yin, Reliability-oriented multi-resource allocation for seru production system with stochastic capacity, *International Journal of Manufacturing Research*, (2019), inpress.

[9] W. M. Han, J. Chen, X. Z. Bu, Batch splitting activity in scheduling of virtual cell with capacity constraints based on bi-level mathematical model, *Applied Mechanics & Materials*, **263-266** (2013), 1257-1264.

[10] P. Hansen, B. Jaumard and G. Savard, New branch-and-bound rules for linear bilevel programming, *SIAM Journal on Scientific & Statistical Computing*, **13** (1992), 1194-1217.

[11] Sakamaki Hisashi . The Change of Consciousness and Company by Cellular Manufacturing in Canon Way, (Japanese), Tokyo: JMAM, 2006.

[12] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence, *Quarterly Review of Biology*, **6** (1975), 126-137.

[13] C. J. Hsu, W. H. Kuo and D. L. Yang, Unrelated parallel machine scheduling with past-sequence-dependent setup time and learning effects, *Applied Mathematical Modeling*, **35** (2010), 1492-1496.

[14] R. H. Huang and T. H. Yu, An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting, *Applied Soft Computing.*, **57** (2017), 642-656.

[15] H. Iwamuro, An easy book about *seru* production, (Japanese), *Nikkan Kogyo Shimbun* Tokyo, 2004.

[16] R. G. Jeroslow, The polynomial hierarchy and a simple model for competitive analysis, *Mathematical Programming*, **32** (1985), 146-164.

[17] J. Jones and T. Soule, Comparing genetic robustness in generational vs. steady state evolutionary algorithms. *In Proceedings of the 8th annual conference on genetic and evolutionary computation*, ACM (2006), 143-150.

[18] R. M. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations* Now York: Plenum, (1972), 85-103.

[19] C. Kasemset and V. Kachitvichyanukul, Bi-level multi-objective mathematical model for job-shop scheduling: the application of theory of constraints, *International Journal of Production Research*, **48** (2010), 6137-6154.

[20] C. Kasemset and V. Kachitvichyanukul, A PSO-based procedure for a bi-level multi-objective TOC-based job-shop scheduling problem, *International Journal of Operational Research*, **14** (2012), 50-69.

[21] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science*, **220** (1983), 671-680.

[22] Jie Lian, ChenGuang Liu, WenJuan Li, Yong Yin, A multi-skilled worker assignment problem in seru production systems considering the worker heterogeneity, *Computers & Industrial Engineering*, **118** (2018), 366-382,

[23] C. G. Liu, F. Dang, W. J. Li, J. Lian, S. Evans and Y. Yin, Production planning of multi-stage multi-option *seru* production systems with sustainable measures, *Journal of Cleaner Production*, **105** (2014), 285-299.

[24] C. G. Liu, J. Lian, Y. Yin and W. J. Li, *Seru* seisan-an innovation of the production management mode in Japan, *Asian Journal of Technology Innovation*, **18** (2010), 89-113.

[25] C. G. Liu, K. E. Stecke, J. Lian and Y. Yin, An implementation framework for *seru* production, *International Transactions in Operational Research*, **21** (2014), 1-19.

[26] C. G. Liu, N. Yang, W. J. Li, J. Lian, S. Evans and Y. Yin, Training and assignment of multi-skilled workers for implementing *seru* production systems, *The International Journal of Advanced Manufacturing Technology*, **69** (2013), 937-959.

[27] C. Low, C. M. Hsu, K. I. Huang, Benefits of lot splitting in job-shop scheduling, *The International Journal of Advanced Manufacturing Technology*, **24** (2004), 773-780.

[28] H. Luo, A. Zhang, G. Q. Huang, Active scheduling for hybrid flowshop with family setup time and inconsistent family formation, *Journal of Intelligent Manufacturing*, **26** (2015), 169-187.

[29] L. Luo, Z. Zhang and Y. Yin, Modelling and numerical analysis of*seru* loading problem under uncertainty, *European J of Industrial Engineering*, **11** (2017), 185-204.

[30] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, Equation of state calculations by fast computing machines, *Journal of Chemical Physics*, **21** (1953), 1087-1092.

[31] Z. Michalewicz, Genetic Algorithm + Data Structure = Evolution Programs", 3rd edition, *Springer-Verlag* New York, 1996.

[32] 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*, **6** (2006), 419C439.

[33] J. Pei, X. Liu, P. M. Pardalos, A. Migdalas and S. Yang, Serial-batching scheduling with time-dependent setup time and effects of deterioration and learning on a single-machine, *Journal of Global Optimization*, **67** (2017), 1-12.

[34] A. Roth, J. Singhal, K. Singhal and C. S. Tang, Knowledge creation and dissemination in operations and supply chain management, *Production and Operations Management*, **25** (2016), 1473-1488.

[35] Y. Sakazume, Is Japanese Cell Manufacturing a New System? : A Comparative Study between Japanese Cell Manufacturing and Cellular Manufacturing, *Journal of Japan Industrial Management Association*, **55** (2005), 341-349.

[36] H. Sakamaki, The Change of Consciousness and Company by Cellular Manufacturing in Canon Way, (Japanese), *Japan Management AssociationManagement Center* Tokyo, 2006.

[37] L. M. Shao, Z. Zhang and Y. Yin, A bi-objective combination optimisation model for line-*seru* conversion based on queuing theory, *International Journal of Manufacturing Research*, **11** (2016), 322-338.

[38] A. Sinha, P. Malo and K. Deb, A review on bilevel optimization: from classical to evolutionary approaches and applications, *IEEE Transactions on Evolutionary Computation*, **22** (2018), 276-295.

[39] 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*, **3** (2012), 105-118.

[40] G. A. Süer and C. Dagli, Intra-cell manpower transfers and cell loading in labor-intensive manufacturing cells, *Computers & Industrial Engineering*, **48** (2005), 643-655.

[41] W. Sun, Y. Wu, Q. Lou and Y. Yu, A Cooperative Coevolution Algorithm for the Seru Production With Minimizing Makespan, *IEEE Access*, **7** (2019), 5662-5670.

[42] W. Sun, Y. Yu, J. W. Wang, Reducing the total tardiness by Seru Production: Model, exact and cooperative coevolution solutions, *International Journal of Production Research*, in press.

[43] J. Tao, , Z. Chao and Y. Xi, A semi-online algorithm and its competitive analysis for a single machine scheduling problem with bounded processing times, *Journal of Industrial and Management Optimization*, **6** (2010), 269-282.

[44] L. Vicente, G. Savard and J. Judice, Descent approaches for quadratic bilevel programming, *Journal of Optimization Theory and Applications*, **81** (1994), 379 - 399.

[45] Not only Toyota-Miraculous Canon Manufacturing System, (Japanese), Weekly Toyo Keizen, 2003.

[46] Y. Wang, J. F. Tang, Cost and service-level-based model for a seru production system formation problem with uncertain demand, *Journal of Systems Science and Systems Engineering*, **27** (2018), 519C537.

[47] X. Yang, D. Qiu and J. Shen, Solving bi-level programming problem based on electromagnetism-like algorithm, *Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering* (2013), 923-927.

[48] Y. Yin, K. E. Stecke and I. Kaku, The evolution of *seru* production systems throughout canon, *Operations Management Education Review*, **2** (2008), 27-40.

[49] Y. Yin, K. E. Stecke, M. Li and I. Kaku, Prospering in a volatile market: Meeting uncertain demand with *seru*, *Working paper, Yamagata University*, 2011.

[50] Y. Yin, T. C. E. Cheng, J. Xu, S. R. Cheng and C. C. Wu, (2013). Single-machine scheduling with past-sequence-dependent delivery times and a linear deterioration, *Journal of Industrial and Management Optimization*, **9** (2013), 323-339.

[51] Y. Yin, K. E. Stecke, M. Swink and I. Kaku, Lessons from *seru*, production on manufacturing competitively in a high cost environment, *Journal of Operations Management*, **49-51** (2017), 67-76.

[52] Y. Yin, K. E. Stecke and D. Li, The evolution of production systems from Industry 2.0 through Industry 4.0, *International Journal of Production Research*, **56** (2018), 848-861.

[53] K. C. Ying and Y. J. Tsai, Minimising total cost for training and assigning multiskilled workers in *seru* production systems, *International Journal of Production Research*, **55** (2017), 2978-2989.

[54] 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*, **236** (2014), 774-786.

[55] Y. Yu, W. Sun, J. Tang, I. Kaku and J. W. Wang, Line-*seru* conversion towards reducing worker(s) without increasing makespan: models, exact and meta-heuristic solutions, *International Journal of Production Research*, **55** (2017), 2990-3007.

[56] Y. Yu, W. Sun, J. Tang, J. Wang, Line-hybrid *seru* system conversion, *Computers & Industrial Engineering*, **103** (2017), 282-299.

[57] Y. Yu, J. W. Wang, K. Ma, W. Sun, *Seru* system balancing: Definition, formulation, and solution, *Computers & Industrial Engineering*, **122** (2018), 318C325.

[58] Y. Yu , J. F. Tang, Review of *seru* production, *Frontiers of Engineering Management*, **6** (2019), 183-192.

[59] X. L. Zhang, C. G. Liu, W. J. Li, S. Evans and Y. Yin, Effects of key enabling technologies for *seru* production on sustainable performance, *Omega*, **66** (2017), 290-307.

[60] Z. Zhang and J. Xu, Bi-level multiple mode resource-constrained project scheduling problems under hybrid uncertainty, *Journal of Industrial & Management Optimization*, **12** (2016), 565-593.

[61] Z. Zhang, J. Xu, H. Yang and Y. Wang, Bi-level optimization of resource-constrained multiple project scheduling problems in hydropower station construction under uncertainty, *Scientia Iranica A*, **22** (2014), 650-667.

[62] Z. Zhang, L. Shao, and Y. Yin, PSO-based algorithm for solving lot splitting in unbalanced seru production systems, *International Journal of Industrial and Systems Engineering*, (2019), inpress.

[63] C. Zhao, Y. Yin, T. C. E. Cheng and C. C. Wu, Single-machine scheduling and due date assignment with rejection and position-dependent processing times, *Journal of Industrial and Management Optimization*, **10** (2014), 691-700.

[64] P. Zwierzyski and H. Ahmad, *Seru* production as an alternative to a traditional assembly line, *Engineering Management in Production and Services*, **10** (2018), 62-69.

Appendix. Further Reading – Recent Papers on *Seru* Production Systems

Abdullah, M. (2018). *Impact of Skill: Seru vs Classical Assembly Line* (Doctoral dissertation, Ohio University).

Beber, J. (2019). *Seru* production system: Assembly cells implementation in a house appliance factory. *Journal of Lean Systems*, *4*(3), 23-43.

Han, X., Zhang, Z., & Yin, Y. (2018, December). Reliability Analysis for a Divisional *Seru* Production System with Stochastic Capacity. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 710-714). IEEE.

Kaku, I. (2016). A Fundamental Positive Investigation into Japanese *Seru* Production Systems. *IFAC-PapersOnLine*, *49*(12), 337-342.

Kaku, I. (2017). Is *Seru* a Sustainable Manufacturing System?. *Procedia Manufacturing*, *8*, 723-730.

Kaku, I., Zhang, X., & Yin, Y. (2017). DESCRIPTION AND EVALUATION OF *SERU* PRODUCTION SYSTEM WITH SF SCHEME. *DEStech Transactions on Engineering and Technology Research*, (icpr).

Li, X., Li, D., Wu, X., Zheng, H., & Yin, Y. (2017, June). A cooperative co-evolution approach for a line-*seru* conversion problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1406-1411). IEEE.

Lian, J., Liu, C., Li, W., & Yin, Y. (2018). A multi-skilled worker assignment problem in *seru* production systems considering the worker heterogeneity. *Computers & Industrial Engineering*, *118*, 366-382.

Liu, C., Dang, F., Li, W., Lian, J., Evans, S., & Yin, Y. (2015). Production planning of multi-stage multi-option seru production systems with sustainable measures. *Journal of Cleaner Production*, *105*, 285-299.

Liu, C., Li, W., Lian, J., & Yin, Y. (2012). Reconfiguration of assembly systems: From conveyor assembly line to *serus*. *Journal of Manufacturing Systems*, *31*(3), 312-325.

Liu, C., Stecke, K. E., Lian, J., & Yin, Y. (2014). An implementation framework for *seru* production. *International Transactions in Operational Research*, *21*(1), 1-19.

Liu, C., Yang, N., Li, W., Lian, J., Evans, S., & Yin, Y. (2013). Training and assignment of multi-skilled workers for implementing *seru* production systems. *The International Journal of Advanced Manufacturing Technology*, *69*(5-8), 937-959.

Luo, L., Zhang, Z., & Yin, Y. (2016, December). *Seru* loading with worker-operation assignment in single period. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 1055-1058). IEEE.

Luo, L., Zhang, Z., & Yin, Y. (2017). Modelling and numerical analysis of *seru* loading problem under uncertainty. *European Journal of Industrial Engineering*, *11*(2), 185-204.

Manupati, V. K., Deepthi, T. V., Ramakotaiah, K., & Rao, S. S. (2015, March). Reconfiguration of Networked *seru* production systems in an Indian Perspective. In *2015 International Conference on Industrial Engineering and Operations Management (IEOM)* (pp. 1-7). IEEE.

Ren, H., & Wang, D. (2019). Analysis of the Effect of the Line-*Seru* Conversion on the Waiting Time with Batch Arrival. *Mathematical Problems in Engineering*, *2019*.

Shao, L., Zhang, Z., & Yin, Y. (2016). A bi-objective combination optimisation model for line-*seru* conversion based on queuing theory. *International Journal of Manufacturing Research*, *11*(4), 322-338.

Shao, L., Zhang, Z., & Yin, Y. (2017). Production System Performance Improvement by Assembly Line-seru Conversion. In *Proceedings of the Tenth International Conference on Management Science and Engineering Management* (pp. 1165-1180). Springer, Singapore.

Singh, S. (2017, July). A study on *seru* production system. In *3rd International Conference on Emerging Technologies in Engineering, Biomedical, Management and Science [3rd ETEBMS-2017]* (Vol. 9).

Sun, W., Li, Q., Huo, C., Yu, Y., & Ma, K. (2016). Formulations, features of solution space, and algorithms for line-pure *seru* system conversion. *Mathematical Problems in Engineering*, *2016*.

Sun, W., Wu, Y., Lou, Q., & Yu, Y. (2019). A cooperative coevolution algorithm for the *seru* production with minimizing makespan. *IEEE Access*, *7*, 5662-5670.

Villa, A., & Taurino, T. (2013). From JIT to *Seru*, for a production as lean as possible. *Procedia Engineering*, *63*, 956-965.

Wang, J., Liu, H., Qu, P., & Yin, Y. (2013). Design and operations of *seru* manufacturing: case study. In *The 6th PSU-UNS International Conference on Engineering and Technology (ICET-2013)* (pp. 15-17).

Wang, Y., & Tang, J. (2017, June). Multi-objective optimization model for seru production system formation under uncertain condition. In *2017 International Conference on Service Systems and Service Management* (pp. 1-6). IEEE.

Wang, Y., & Tang, J. (2018). Cost and service-level-based model for a *seru* production system formation problem with uncertain demand. *Journal of Systems Science and Systems Engineering*, *27*(4), 519-537.

Wang, Y., & Tang, J. (2018, June). Is full skill the best configuration for *seru* production system?. In *2018 Chinese Control And Decision Conference (CCDC)* (pp. 3904-3909). IEEE.

Wu, L., Chan, F. T., Niu, B., & Li, L. (2018). Cross-trained worker assignment and comparative analysis on throughput of divisional and rotating *seru*. *Industrial Management & Data Systems*, *118*(5), 1114-1136.

Yin, Y., Kaku, I., & Liu, C. (2016). Management of overlapped cross-training: with or without a supervisor?. *Asian Journal of Management Science and Applications*.

Yin, Y., Stecke, K. E., & Li, D. (2018). The evolution of production systems from Industry 2.0 through Industry 4.0. *International Journal of Production Research*, *56*(1-2), 848-861.

Yin, Y., Stecke, K. E., Swink, M., & Kaku, I. (2017). Lessons from *seru* production on manufacturing competitively in a high cost environment. *Journal of Operations Management*, *49*, 67-76.

Ying, K. C., & Tsai, Y. J. (2017). Minimising total cost for training and assigning multiskilled workers in *seru* production systems. *International Journal of Production Research*, *55*(10), 2978-2989.

Yu, Y., Gong, J., Tang, J., Yin, Y., & Kaku, I. (2012). How to carry out assembly line–cell conversion? A discussion based on factor analysis of system performance improvements. *International Journal of Production Research*, *50*(18), 5259-5280.

Yu, Y., Sun, W., Tang, J., Kaku, I., & Wang, J. (2017). Line-*seru* conversion towards reducing worker (s) without increasing makespan: models, exact and meta-heuristic solutions. *International Journal of Production Research*, *55*(10), 2990-3007.

Yu, Y., Sun, W., Tang, J., & Wang, J. (2017). Line-hybrid *seru* system conversion: Models, complexities, properties, solutions and insights. *Computers & Industrial Engineering*, *103*, 282-299.

Yu, Y., & Tang, J. (2019). Review of *seru* production. *Frontiers of Engineering Management*, *6*(2), 183-192.

Yu, Y., Tang, J., Gong, J., Yin, Y., & Kaku, I. (2014). Mathematical analysis and solutions for multi-objective line-cell conversion problem. *European Journal of Operational Research*, *236*(2), 774-786.

Yu, Y., Tang, J., Sun, W., Yin, Y., & Kaku, I. (2013). Reducing worker (s) by converting assembly line into a pure cell system. *International Journal of Production Economics*, *145*(2), 799-806.

Yu, Y., Tang, J., Sun, W., Yin, Y., & Kaku, I. (2013). Combining local search into non-dominated sorting for multi-objective line-cell conversion problem. *International Journal of Computer Integrated Manufacturing*, *26*(4), 316-326.

Yu, Y., Tang, J., Yin, Y., & Kaku, I. (2015). Comparison of two typical scheduling rules of line-*seru* conversion problem. *Asian Journal of Management Science and Applications*, *2*(2), 154-170.

Yu, Y., Wang, J., Ma, K., & Sun, W. (2018). Seru system balancing: Definition, formulation, and exact solution. *Computers & Industrial Engineering*, *122*, 318-325.

Yu, Y., Wang, S., Tang, J., Kaku, I., & Sun, W. (2016). Complexity of line-*seru* conversion for different scheduling rules and two improved exact algorithms for the multi-objective optimization. *SpringerPlus*, *5*(1), 809.

Zhang, X., Liu, C., Li, W., Evans, S., & Yin, Y. (2017). Effects of key enabling technologies for *seru* production on sustainable performance. *Omega*, *66*, 290-307.

Zwierzyński, P., & Ahmad, H. (2018). *Seru* production as an alternative to a traditional assembly line. *Engineering Management in Production and Services*, *10*(3), 62-69.

吴旭辉, 杜劭峰, 郝慧慧, 于洋, 殷勇, & 李冬妮. (2018). 一种基于协同进化的流水线向 *Seru* 系统转化方法. *自动化学报*, *44*(6), 1015-1027.