Computers & Industrial Engineering 122 (2018) 318–325

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Seru system balancing: Definition, formulation, and exact solution

Yang Yu[a,b], Junwei Wang[b,*], Ke Ma[a], Wei Sun[c]

[a] State Key Laboratory of Synthetic Automation for Process Industries, Institute of Intelligent Systems, Northeastern University, Shenyang 110819, PR China
[b] Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
[c] Business School, Liaoning University, Shenyang 110316, PR China

ARTICLE INFO

Keywords:
Manufacturing
Seru production
Workload
Balancing

ABSTRACT

Seru production can reduce makespan, labor hours and manpower by improving workers' workload balance based on the reconfiguration of workers. Therefore, this study focuses on the fundamental principles of *seru* system balancing. For a *seru*, we define and formulate *seru* balance (*SB*) to describe workloads balance of the workers in the *seru*. For a *seru* system containing more *serus*, the *seru* system balance (*SSB*) needs to be evaluated from workloads balance of all workers and workloads balance among all *serus*. Consequently, we define and formulate *intra-seru* system balancing (*Intra-SSB*) and *inter-seru* system balancing (*Inter-SSB*) to evaluate the two perspectives respectively. We theoretically develop the lower and upper bounds of *Intra-SSB and Inter-SSB* respectively. In addition, we define and formulate the *seru* system balancing problem (*SSBP*) as a bi-objective model with maximizing *Intra-SSB* and *Inter-SSB* simultaneously. The property of solution space for *SSBP* is analyzed. Finally, we develop an improved algorithm based on ε-constraint for *SSBP*. The algorithm saves computation time by cutting non-Pareto-optimal solutions before performing ε-constraint.

1. Introduction

Production line balancing is a key performance indicator for any production system (such as Assembly line production system and Toyota Production System). A good production line balancing can prohibit low efficiency of a production system by balancing workload and process time, and minimizing labor slack.

Assembly line production system, as one of the most important production systems in history, brought by Ford Motor Company, was recognized and appreciated by many engineers and scholars. Since Henry Ford created the first revolutionary automobile assembly line, known as the model T in 1913, assembly line production system was used for massive production.

The well-known Toyota Production System (TPS/JIT/Lean Production) was a revolutionary exploration in improving assembly line production system. TPS integrated Just-In-Time and Automation, and was widely used in automobile production industries. TPS assembles similar products and is used to satisfy diversified customer need.

Nowadays, markets have the following characteristics: short product life cycles, uncertain product types (Muriel, Somasundaram, & Zhang, 2006; Wang, Dou, Muddada, & Zhang, 2017; Wang et al., 2016), and fluctuating production volumes (Fu, Ding, Wang, & Wang, 2017; Fu, Wang, Huang, & Wang, 2017; Wang, Fu, Huang, Huang, & Wang, 2017; Yin, Stecke, Swink, & Kaku, 2017). Traditional assembly line, which was developed for mass-production of standardized products, is not suitable for the turbulent markets. *Seru* production was developed to cope with the turbulent markets (Liu, Stecke, Lian, & Yin, 2014; Yin et al., 2017). *Seru* production is implemented based on at least a *seru* system. *Seru* system is a more productive, efficient, and flexible system than the conveyor assembly line due to combining strengths from Toyota's lean philosophy and Sony's one-person production organization. *Seru* system contains at least one *seru*. *Seru* is flexible, because the workers in *seru* system are multi-skilled operator who can perform multiple tasks and process multiple products. *Seru* production has many benefits such as reducing lead-time, setup time, required workforce, WIP inventories, finished-product inventories, cost, and shop floor space (Stecke, Yin, Kaku, & Murase, 2012; Yin et al., 2017). However, *seru* system balance, as a key performance indicator of *seru* production, has not been investigated yet.

Therefore, this study focuses on the fundamental principles of *seru* system balancing. For a *seru*, we define and formulate *seru* balance (*SB*) to describe workloads balance of the workers in the *seru*. For a *seru* system containing more *serus*, we evaluate the *seru* system balance (*SSB*) from two perspectives: workloads balance of all workers and workloads balance among all *serus*. We define *intra-seru* system balancing (*Intra-SSB*) and *inter-seru* system balancing (*Inter-SSB*) to evaluate the two perspectives respectively. We theoretically develop the lower and upper bounds of *Intra-SSB and Inter-SSB* respectively. We define the

* Corresponding author.
E-mail address: jwwang@hku.hk (J. Wang).

https://doi.org/10.1016/j.cie.2018.05.048
Received 5 July 2017; Received in revised form 1 March 2018; Accepted 28 May 2018
Available online 30 May 2018
0360-8352/

seru system balancing problem (*SSBP*) as a bi-objective mathematical model with maximizing *Intra-SSB* and *Inter-SSB* simultaneously. The solution space of *SSBP* is analyzed. An improved exact algorithm based on ε-constraint is developed to obtain the Pareto-optimal solutions of *SSBP*.

2. Literature review

2.1. Assembly line balancing and TPS balancing

Assembly line balancing (ALB) is of vital importance to assembly line production system. By increasing assembly line balance, production efficiency of assembly line can be greatly improved. Salveson (1955) formulated assembly line balancing. Assembly line balancing is to arrange the individual processing and assembly tasks at the workstations so that the total time required at each workstation is approximately the same (Baybars, 1986). By reaching good assembly line balancing, the total slack can be reduced to an optimized level so that the assembly line can work more effectively and efficiently. Assembly line balancing problem (ALBP) proposed by Helgeson and Birnie (1961) is used to subsume optimization models that seek to support this decision process.

Boysen, Fliedner, and Scholl (2007) classified ALBP into two types: simple assembly line balancing problem (SALBP) and general assembly line balancing problem (GALBP). Salveson (1955) established a simplified mathematical model solving assembly line balancing problem, which is labeled as the SALBP. Boysen et al. (2007) made a classification on SALBP: (1) SALBP-I minimizing the number of stations for a given cycle time; (2) SALBP-II minimizing the cycle time for a given number of stations (Uğurdağ, Rachamadugu, & Papachristou, 1997); (3) SALBP-E (E expresses efficiency) maximizing the line efficiency by minimizing the number of stations and the cycle time simultaneously. Balancing in SALBP-E is expressed as $ALB = \frac{sum\ of\ all\ tasktime}{number\ of\ workstations * cycletime}$ (Esmaeilbeigi, Naderi, & Charkhgard, 2015); and (4) SALBP-F finding a feasible balancing given the number of stations and the cycle time. In addition, Baybars (1986) reviewed the exact algorithms in solving SALBP. Klein and Scholl (1996) proposed a branch and bound procedure to solve the problem of maximizing the production rate in simple assembly line balancing. Becker and Scholl (2006) and Scholl and Becker (2006) surveyed the exact and heuristic algorithms in solving SALBP. Otto, Otto, and Scholl (2013) proposed a systematic data generation and test design for solution algorithms on the example of SALBP. Otto and Otto (2014) designed effective priority rules for simple assembly line balancing. Chica, Bautista, Cordón, and Damas (2016) established a multi-objective model for robust time and space assembly line balancing under uncertain demand and developed evolutionary algorithms to solve it.

Other researches considered the assembly line balancing with some practical aspects, such as processing alternatives (Pinto, Dannenbring, & Khumawala, 1983), parallel stations, two sided, U-shaped line (Aase et al., 2003), and two-sided u-shaped lines. They are called as general assembly line balancing (GALB). Boysen et al. (2007) reviewed GALB problems including mathematical models, bound computations, exact solutions, and heuristic solutions. Battaïa and Dolgui (2012) proposed a reduction approaches for a GALB. In addition, Bentaha, Battaïa, and Dolgui (2014) propose an approximation method for disassembly line balancing problem under uncertainty.

In TPS/JIT, balancing problem is considered during the implementation process. Plenert (1997) described the balancing problem of JIT. During the initial stages of JIT implementation (the first few years), the focus is on inventory waste elimination and causing workload imbalances. Subsequently, the focus is on labor waste elimination by fine tuning the workload balancing on the production line. Li, Chang, Ni, and Biller (2009) performed a case study on an automotive JIT system. They proposed a control method that adjusts the maintenance prioritization and the initial buffer to balance the line. They defined balancing of JIT by the blockage time and starvation time of all stations.

2.2. Seru production

For *seru* production, a new production system, the balance has not been investigated yet. This study focuses on the fundamental principle on *seru* system balance.

Seru, conceived by Sony in Japan, is lean and agile. *Seru* is an assembly unit including several simple equipments and one or more multi-skilled workers. Workers in a *seru* are cross-trained (Tekin, Hopp, & Oyen, 2002) and implement most or all of tasks in the *seru*. There are three types of *serus* (Stecke et al., 2012; Yin et al., 2017). A divisional *seru* is a short line staffed with several partially cross-trained workers, where several tasks are grouped and each group of tasks is performed by one or more workers. A rotating *seru* is where every worker assembles an entire product from-start-to-finish without disruption. A *yatai* represents a special rotating *seru* with only one worker. In this study, only divisional *seru* and *yatai* are considered.

A *seru* system contains one or more *serus*. *Seru* system can obtain a better performance by the reconfiguration of workers than the assembly line. A detailed introduction of the *seru* system and its managerial mechanism can be found in Yin et al. (2017) and Stecke et al. (2012). *Seru* production has several stunning advantages, as Liu et al. (2014) concluded that *seru* has high flexibility, low inventory, and good morale. In addition, *seru* production can reduce makespan, setup time, required workers, cost, and shop floor space. *Seru* system has successfully been applied by many leading Japanese companies such as Sony, Canon, Panasonic, NEC, Fujitsu, Sharp, and Sanyo (Yin et al., 2017).

Liu et al. (2014) explained the implementation framework for *seru* production and stated that balance of *seru* production should be considered. Kaku, Gong, Tang, and Yin (2009) established the mathematical model of line-*seru* conversion. They used the model to illustrate a *seru* system can obtain a better makespan performance than an assembly line. Generally, the following several performances are used to evaluate *seru* system. Kaku et al. (2009), Yu, Tang, Gong, Yin, and Kaku (2014), Yu, Sun, Tang, Kaku, and Wang (2017), and Yu, Sun, Tang, & Wang (2017) used makespan and total labor time to evaluate the efficiency of *seru* system.

However, previous researches did not investigate why makespan, total labor time, and number of workers can be improved by *seru* production. In fact, *seru* production decreases the influences of the bottleneck worker (i.e., the worker with the worst skill) by worker reconfiguration. Therefore, the balancing of a *seru* system needs to be investigated. However, up to now the fundamental principles of *seru* system balancing are not investigated still.

3. Definition and formulation of *seru* system balancing

3.1. Assumptions

Product types and batches to be assembled are known in advance. N product types that are divided into M product batches. Each batch contains a single product type. Obviously, $M \geqslant N$.

1. Each product batch needs to be assembled entirely in a *seru*. In other words, a batch cannot be shared by *serus*.
2. The tasks in each *seru* are identical.

3.2. Notations

We define the following notations.

- Indices

  i: Index of workers ($i = 1, 2\ldots, W$). The number of tasks equals the number of workers because in the assembly line a task is performed by a worker.

j: Index of *serus* ($j = 1, 2..., J$).

n: Index of product types ($n = 1, 2..., N$).

m: Index of product batches ($m = 1, 2..., M$).

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

- Parameters

$V_{mn} = \begin{cases} 1, & \text{if product type of product batch } m \text{ is } n \\ 0, & \text{otherwise} \end{cases}$

TP_{ni}: Worker *i*'s average task time for product type *n*.

TM_{mi}: Worker *i*'s average task time for batch *m*, therefore, $TM_{mi} = \sum_{n=1}^{N} V_{mn} TP_{ni}$.

B_m: Size of product batch *m*.

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 in a *seru* will be longer than her or his average task time (TP_{ni}).

ε_i: Worker *i*'s coefficient of influencing level of doing multiple tasks.

C_i: Coefficient of variation of worker *i*'s increased task time because of assembling all tasks in a *seru*, as shown in Eq. (1).

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

T_{mi}: Worker *i*'s average task time for batch *m* in a *seru*, $T_{mi} = TM_{mi} * C_i$. If the number of worker *i*'s tasks in a *seru* is over her or his upper bound η_i, i.e., $W > \eta_i$, then the worker will cost more average task time than her or his task time, i.e., $T_{mi} > TM_{mi}$.

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

$Z_{mjk} = \begin{cases} 1, & \text{if product batch } m \text{ is assigned to } seru\ j \text{ in sequence } k \\ 0, & \text{otherwise} \end{cases}$

3.3. Definition and formulation of seru balancing (SB) & seru system balancing (SSB)

A case of a *seru* is shown in Fig. 1. For the *seru*, the workloads balancing of the two workers in the *seru* should be evaluated. Therefore, we propose *seru* balancing (*SB*). Subsequently, we formulate *SB* by making reference to SALBP-E.

Definition1. *Seru* balancing (*SB*) is to evaluate the workloads balancing of works in a *seru*.

For a *seru* with *W* workers assembling batch *m*, *SB_m* is formulated as the following equation.

$$SBm = \frac{\sum_{i=1}^{W} T_{mi}}{\max_i(T_{mi}) * W}, \quad \forall\, m \tag{2}$$

where

$\sum_{i=1}^{W} T_{mi}$ is the sum of task time of all workers in the *seru* assembling batch *m*;

$\max_i(T_{mi})$ expresses the maximum task time of workers in the *seru*

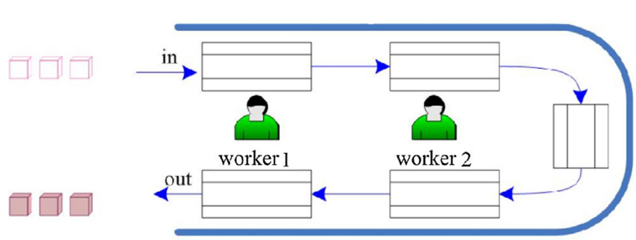

Fig. 1. A case of a *Seru*.

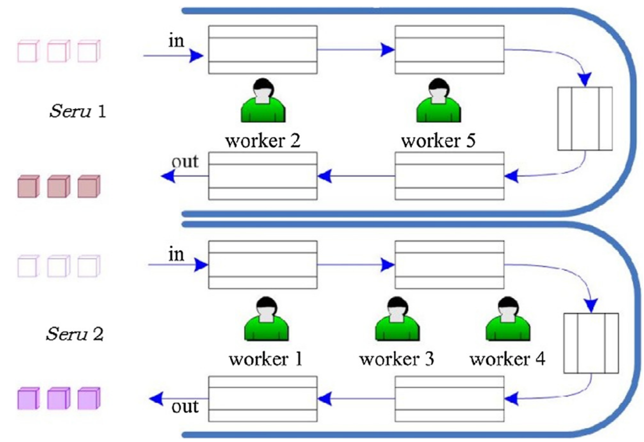

Fig. 2. A case of a *Seru* system.

assembling batch *m*;

W is the number of workers in the *seru*.

For a *seru* with *W* workers assembling *M* batcher, *SB* is calculated by the average balancing of assembling the *M* batches, formulated as the following equation.

$$SB = \frac{1}{M} \sum_{m=1}^{M} \frac{\sum_{i=1}^{W} T_{mi}}{\max_i(T_{mi}) * W} \tag{3}$$

However, a *seru* system is composed of one or more *serus*. A case of a *seru* system is given in Fig. 2, where the *seru* system includes two *serus*. Therefore, not only workloads balancing of all workers in the two *serus* needs to be evaluated, but workloads balancing between the two *serus* needs to be evaluated. Thus, for a *seru* system, we propose *intra-seru* system balancing (*Intra-SSB*) to evaluate workloads balancing of all workers in all *serus*, and *inter-seru* system balancing (*Inter-SSB*) to evaluate workloads balancing of all *serus*.

Definition 2. *Intra-seru* system balancing (*Intra-SSB*) is to evaluate the workload balancing of all workers in a *seru* system.

For a *seru* system with *J serus*, *Intra-SSB* is calculated by the average *seru* balancing (*SB*) of all *serus* in the *seru* system, formulated as the following equation.

$$Intra\text{-}SSB = \frac{1}{J} \sum_{j=1}^{J} \frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W} \sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}}{\max_i\left(\sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}\right) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M} \sum_{k=1}^{M} Z_{mjk}} \tag{4}$$

where

$\frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W} \sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M} \sum_{k=1}^{M} Z_{mjk}}$ expresses the *seru* balancing (*SB*) of *seru j*;

$\sum_{m=1}^{M} \sum_{k=1}^{M} Z_{mjk}$ is the number of batches assembled in *seru j*;

$\frac{\sum_{i=1}^{W} \sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}$ expresses the balancing of *seru j* assembling batch *m*;

$\sum_{i=1}^{W} \sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk}$ is the sum of task time of all workers in *seru j* assembling batch *m*;

$\max_i(\sum_{k=1}^{M} T_{mi} X_{ij} Z_{mjk})$ is the maximum task time of workers in *seru j* assembling batch *m*;

$\sum_{i=1}^{W} X_{ij}$ is the number of workers in *seru j*.

Definition 3. *Inter-seru* system balancing (*Inter-SSB*) is to evaluate the workload balancing of all *serus* in a *seru* system.

For a *seru* system with *J serus*, *Inter-SSB* is calculated by the following equation.

$$Inter\text{-}SSB = \frac{\sum_{j=1}^{J} makespan_j}{J * maxMakespan} \tag{5}$$

In Eq. (5), $makespan_j$ is the makespan of *seru j*, as expressed in Eq. (6); *maxMakespan* is the maximal makespan of all *serus* in the *seru* system.

$$makespan_j = \sum_{m=1}^{M} (FSB_m + FS_m + SS_m) | Z_{mjk} = 1, \quad \forall\, j \tag{6}$$

3.4. Lower and upper bounds for Intra-SSB and Inter-SSB

We theoretically develop the lower and upper bounds for *Intra-SSB* and *Inter-SSB*.

Theorem 1. Upper bound of *Intra-SSB* is 1.

Proof. Obviously, $\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk} \leqslant \max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk})$, so, $\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk} \leqslant \max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}$. Therefore, $\frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}} \leqslant 1, \quad \forall\, m, j$. This means that for any *seru j* assembling any batch *m*, the *seru* balance is at most 1. Therefore, for any *seru j* assembling $\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}$ batches, $\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}} \leqslant \sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}, \quad \forall\, j$.

So, $\frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}} \leqslant 1, \quad \forall\, j$, and $\sum_{j=1}^{J} \frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}} \leqslant J$. Thus, the upper bound of Intra-SSB is 1, i.e., $\frac{1}{J}\sum_{j=1}^{J} \frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}} \leqslant 1$.

When the workers in a *seru* have the same skill, i.e., the T_{mi} is identical, *Intra-SSB* is 1. □

Theorem 2. Lower bound of *Intra-SSB* is $\frac{1}{J}\sum_{j=1}^{J}\frac{1}{|S_j|}$, where $|S_j|$is the number of workers in *seru j*.

Proof. Let $|S_j| = \sum_{i=1}^{W} X_{ij}$ expressing the number of workers in *seru j*, and then

$$\frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}$$
$$= \frac{1}{|S_j|}\frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk})}$$
$$= \frac{1}{|S_j|}\frac{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) + \sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk} - \max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk})}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk})}$$

Obviously, $\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk} - \max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk})_i \geqslant 0$, therefore, $\frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}} \geqslant \frac{1}{|S_j|}$. This means that for any *seru j* assembling any batch *m*, the *seru* balance is at least $\frac{1}{|S_j|}$. Therefore, for any *seru j* assembling $\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}$ batches, $\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}} \geqslant \frac{1}{|S_j|} * \sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}, \quad \forall\, j$.

So, $\frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}} \geqslant \frac{1}{|S_j|}, \quad \forall\, j$, and $\sum_{j=1}^{J} \frac{\sum_{m=1}^{M} \frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}} \geqslant \sum_{j=1}^{J}\frac{1}{|S_j|}$. Therefore, the lower bound is $\frac{1}{J}\sum_{j=1}^{J}\frac{1}{|S_j|}$.

When each *seru* has the worst balance between workers, i.e., $\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk} = \max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk})$, *Intar-SSB* is $\frac{1}{J}\sum_{j=1}^{J}\frac{1}{|S_j|}$. □

Theorem 3. Upper bound of *Inter-SSB* is 1.

Proof. Obviously, $makespan_j \leqslant maxMakespan, \quad \forall\, j$, so, $\sum_{j=1}^{J} makespan_j \leqslant J * maxMakespan$, therefore $\frac{\sum_{j=1}^{J} makespan_j}{J * maxMakespan} \leqslant 1$.

When all *serus* have the same makespan, *Inter-SSB* is 1. □

Theorem 4. Lower bound of *Inter-SSB* is $\frac{1}{J}$.

Proof. Obviously, $\sum_{j=1}^{J} makespan_j - maxMakespan \geqslant 0$, therefore,

$$\frac{\sum_{j=1}^{J} makespan_j}{J * maxMakespan} = \frac{1}{J}\frac{maxMakespan + \left(\sum_{j=1}^{J} makespan_j - maxMakespan\right)}{maxMakespan}$$
$$= \frac{1}{J}\left(1 + \frac{\sum_{j=1}^{J} makespan_j - maxMakespan}{maxMakespan}\right)$$
$$\geqslant \frac{1}{J}$$

When the *seru* system has the worst balance, i.e., $\sum_{j=1}^{J} makespan_j = maxMakespan$, *Inter-SSB* is 1. □

4. Definition and model of *seru* system balancing problem (*SSBP*)

4.1. Definition of SSBP

Definition 4. *Seru* system balancing problem (*SSBP*) is to seek a *seru* system with maximal *Intra-SSB* and *Inter-SSB* simultaneously.

According to the definition of *SSBP*, we establish the mathematical model of *SSBP*. Obviously, the model of *SSBP* is a bi-objective optimization model.

4.2. Mathematical model of SSBP

The bi-objective model of *SSBP* is shown as below:

Objective functions:

$$\max(Intra\text{-}SSB) = \max\left\{\frac{1}{J}\sum_{j=1}^{J}\frac{\sum_{m=1}^{M}\frac{\sum_{i=1}^{W}\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}}{\max_i(\sum_{k=1}^{M} T_{mi}X_{ij}Z_{mjk}) * \sum_{i=1}^{W} X_{ij}}}{\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk}}\right\} \tag{7}$$

$$\max(Inter\text{-}SSB) = \max\left\{\frac{\sum_{j=1}^{J} makespan_j}{J * maxMakespan}\right\} \tag{8}$$

Subject to:

$$\sum_{j=1}^{J}\sum_{i=1}^{W} X_{ij} = W \tag{9}$$

$$\sum_{i=1}^{W} X_{ij} \geqslant 1, \quad \forall\, j \tag{10}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk} = 0, \quad \left(\forall\, j | \sum_{i=1}^{W} X_{ij} = 0\right) \tag{11}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk} \leqslant \sum_{m'=1}^{M}\sum_{k'=1}^{M} Z_{(m-1)j'k'}, \quad m = 2, 3, \ldots, M \tag{12}$$

The objectives of *SSBP* are to maximize *Intra-SSB* and *Inter-SSB* simultaneously. Eq. (9) means that the *seru* system contains *W* workers. $\sum_{i=1}^{W} X_{ij}$is the number of the workers in *seru j*, and so $\sum_{j=1}^{J}\sum_{i=1}^{W} X_{ij}$ is the number of the workers in all *serus*, i.e., the total number of workers (*W*). Eq. (10) is *seru* formation rule to ensure each *seru* contain at least one worker, because a *seru* contains one or more worker(s). Eq. (11) is the assigning constraint to guarantee that a product is assigned to the *seru* containing at least one worker. Eq. (12) is the assignment rule by which product batches must be assigned sequentially.

4.3. Solution space of SSBP

SSBP is in effect a two-stage decision process. The first step is the *seru* formation and the second step is the *seru* load. *Seru* formation determines how many *serus* should be formed and how to assign workers to *serus*, decided by the decision variable of X_{ij}.

Seru formation of *SSBP* is to partition W workers into pair-wise disjoint nonempty *serus* as an instance of the unordered set partition problem. Set partitioning is a well-known NP-hard problem. The number of all the feasible solutions of *seru* formation with W workers can be expressed as $F(W) = \sum_{J=1}^{W} P(W, J)$, where $P(W, J)$ is the count of partitioning W workers into J *serus* and can be expressed as the Stirling numbers of the second kind.

The *seru* load determines which batches are dispatched to the appropriate constructed *serus*, decided by the decision variable of Z_{mjk}. Given a *seru* formation with J *serus* and M batches, the number of all the feasible solutions of *seru* load is J^M. Obviously, *seru* load is an instance of scheduling which is a well-known NP-hard problem.

Therefore, *SSBP* is a complex problem including two NP-hard problem (i.e., *seru* formation and *seru* load), and the number of all the feasible solutions in solution space of *SSBP* is $T(W) = \sum_{J=1}^{W} P(W, J)*J^M$.

For simplicity and without loss of generality, this study uses FCFS (First Come First Serve) rule in *seru* load. FCFS rule is applied in many companies. FCFS of *seru* load is described as follows: (1) an arriving product batch is assigned to the empty *seru* with the smallest *seru* number; and (2) if all *serus* are occupied, the product batch is assigned to the *seru* with the earliest completion time. *SSBP* with FCFS is still an NP-hard problem because *SSBP* includes *seru* formation that is NP-hard. Identical to line-*seru* conversion with FCFS (Yu et al., 2014), the number of feasible solutions in solution space of *SSBP* is $T(W) = \sum_{J=1}^{W} P(W, J)*J!$. Comparison **between** $T(W)$ and 2^W is given in Table 1.

Obviously, not only $T(W)$ increases exponentially with the number of workers (W), but also $T(W)$ is much larger than 2^W.

4.4. Several properties in solutions of SSBP

Property 1. The upper bound (i.e., 1) of *Intra-SSB* is in the solution containing *W seru*.

Explanation. The solution containing *W seru* means that each *seru* contains only one worker. According to the formulation of *Intra-SSB*, i.e., Eq. (8), $\sum_{i=1}^{W} X_{ij} = 1, \quad \forall j$. Thus, for the *seru* containing only one worker, *SB* equals 1. Since *SB* of each *seru* equals 1, *Intra-SSB* equals 1.

Property 1 is used to find out an important Pareto-optimal solution to cut some non-Pareto-optimal solutions in *SSBP*.

Property 2. The upper bound (i.e., 1) of *Inter-SSB* is in the solution containing only one *seru*.

Explanation. For the solution containing only one *seru*, $makespan_j$ ($j = 1$) equals *maxMakespan*. According to the formulation of *Inter-SSB*, i.e., Eq. (5), *Inter-SSB* equals 1.

Property 2 is used to find out another important Pareto-optimal solution to cut some non-Pareto-optimal solutions in *SSBP*.

Property 3. In SSBP, there is at least one solution whose *Intra-SSB* is no less than assembly line's balancing.

Explanation. For the solution containing only one *seru*, according to the formulation of *Intra-SSB*, i.e., Eq. (4), $\sum_{i=1}^{W} X_{ij} = W$ and $\sum_{i=1}^{W}\sum_{k=1}^{M} X_{ij} Z_{mjk} = M$. At this time, $Intra\text{-}SSB = SB = \frac{1}{M}\sum_{m=1}^{M} \frac{\sum_{i=1}^{W} T_{mi}}{\max_i(T_{mi}) * W}$. In addition, for the assembly line with W workers assembling M batch, the balancing also equals $\frac{1}{M}\sum_{m=1}^{M} \frac{\sum_{i=1}^{W} T_{mi}}{\max_i(T_{mi}) * W}$, according to the definition of SALBP-E.

5. Improved exact algorithm to solve *SSBP*

5.1. An improved exact algorithm based on ε-constraint

For a bi-objective problem the Pareto-optimal solutions (Ramezanian & Ezzatpanah, 2015) needs to be solved. The ε-constraint method is a classical exact algorithm in solving bi-objectives. The time complexity of ε-constraint method is $O(PN)$, where P is the number of Pareto-optimal solutions, and N is the number of all the feasible solutions. Obviously, time complexity will be greatly decreased if the number of solutions (N) engaged in ε-constraint is reduced. Thus, we consider to cut non-Pareto-optimal solutions before performing ε-constraint.

We define three special Pareto-optimal solutions, i.e., *Intra-SSB*-I, *Inter-SSB*-I, and *Closest-Ideal-Point*.

Definition 5. *Intra-SSB*-I is the Pareto-optimal solution whose *Intra-SSB* is 1.

Explanation. Theorem 1 proves that the maximal *Intra-SSB* is 1. *Intra-SSB*-I is the solution with the maximal *Inter-SSB* in all solutions with *Intra-SSB* equal to 1.

Definition 6. *Inter*-SSB-I is the Pareto-optimal solution whose *Inter-SSB* is 1.

Explanation. Theorem 2 proves that the maximal *Inter-SSB* is 1. *Inter-SSB*-I is the solution with the maximal *Intra-SSB* in all solutions with *Inter-SSB* equal to 1.

Definition 7. Closest-*Ideal-Point* is the solution closest to Ideal Point.

Explanation: Ideal Point in *SSBP* is (1, 1), i.e., *Intra-SSB* equal to 1 and *Inter-SSB* equal to 1.

Intra-SSB-I, *Inter-SSB*-I, and *Closest-Ideal-Point* are shown in Fig. 3.

Theorem 5. *Closest-Ideal-Point* is Pareto-optimal.

Proof. Since Closest-*Ideal-Point* is the solution closest to Ideal Point, there is no solution in the quarter circle with center point Ideal Point, and a radius of the distance from Ideal Point to *Closest-Ideal-Point*. Therefore, there is no solution in the rectangle formed by Ideal Point and *Closest-Ideal-Point* because the rectangle is in the quarter circle, as shown in Fig. 3. Thus, no solutions dominated *Closest-Ideal-Point*. So *Closest-Ideal-Point* is Pareto-optimal solution.

In Fig. 3, any solution in the rectangle formed by *Intra-SSB*-I and (0,0) is dominated by *Intra-SSB*-I; any solution in the rectangle formed by *Inter-SSB*-I and (0,0) is dominated by *Inter-SSB*-I; and any solution in the rectangle formed by *Closest-Ideal-Point* and (0, 0) is dominated by *Closest-Ideal-Point*. Thus, using *Intra-SSB*-I, *Inter-SSB*-I, and *Closest-Ideal-Point*, we cut most of non-Pareto-optimal solutions.

Based on Definitions 5–7 and Theorem 5, we develop an improved exact algorithm based on ε-constraint, described as follows. □

Table 1
Comparison between $T(W)$ and 2^W.

W	1	2	3	4	5	6	7	8	9	10
2^W	2	4	8	16	32	64	128	256	512	1024
$T(W)$	1	3	13	75	541	4683	47,293	545,835	7,087,261	102,247,563

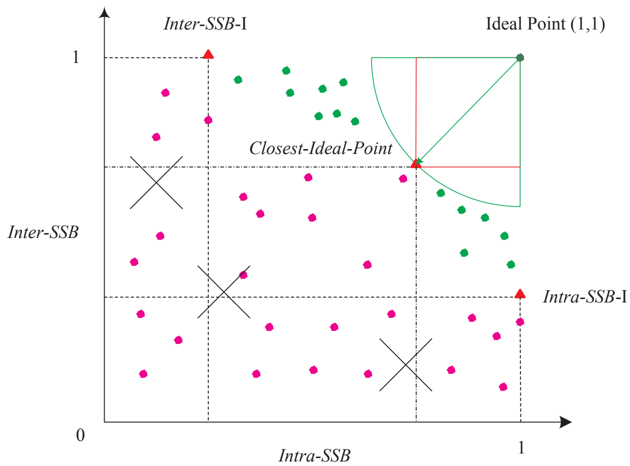

Fig. 3. *Intra-SSB*-I, *Inter-SSB*-I, *Closest-Ideal-Point*, and non-pareto-optimal solutions dominated by them.

Algorithm. Improved exact algorithm based on ε-constraint.

Input: S (Set of all the feasible solutions).
Output: *SPO* (Set of Pareto-optimal solutions of *SSBP*).
(1) Initialize
SF (Set of the solutions not dominated by *Intra-SSB*-I or *Inter-SSB*-I) $\leftarrow \emptyset$
SFF (Set of the final solutions not dominated by *Intra-SSB*-I, *Inter-SSB*-I, or *Closest-Ideal-Point*)
(2) Find out *Intra-SSB*-I and *Inter-SSB*-I in S.
(3) Produce *SF*.
(4) Find out *Closest-Ideal-Point* in *SF*.
(5) Produce *SFF* by *Closest-Ideal-Point* and *SF*.
(6) Obtain the Pareto-optimal solutions by performing ε-constraint in *SFF*.
(7) Output *SPO*.

5.2. Time complexity of the improved exact algorithm

Suppose that number of solutions in set S is N, number of the solutions in *SF* is $|SF|$, number of the solutions in *SFF* is $|SFF|$, then time complexity of each step is shown in Table 2.

Comprehensively, the total time complexity is $O(2N + 2|SF| + P|SFF|)$. Obviously, both $|SF|$ and $|SFF|$ are less than N. Therefore, the time complexity is less than $O(PN)$. When P is large or $|SFF|$ is small, our proposed exact algorithm is much better than the ε-constraint method with the time complexity of $O(PN)$.

6. Computational experiment

In computational experiment, we use examples that are generated by simulating *seru* applications we investigated in Canon (Yu et al., 2014). The data are described in Tables 3–6.

Table 2
Time complexity of each step in the proposed exact algorithm.

Steps	*Description*	*Time complexity*
(2)	Find out *Intra-SSB*-I and *Inter-SSB*-I in S	$O(N)$
(3)	Produce *SF*	$O(N)$
(4)	Find out *Closest-Ideal-Point* in *SF*	$O(\|SF\|)$
(5)	Produce *SFF* by *Closest-Ideal-Point* and *SF*	$O(\|SF\|)$
(6)	Obtain the Pareto-optimal solutions by performing ε-constraint in *SFF*	$O(P\|SFF\|)$

Table 3
The parameters used in the test.

Factor	*Value*
Product Types	5
SSP_n	1.0
η_i	9

6.1. Test instances

Table 3 shows the parameters used in the test. Table 4 shows the average task time of workers for product types, i.e., the detailed data of TP_{ni}. Table 5 shows coefficient of influencing level of doing multiple tasks for workers, i.e., the detailed data of ε_i. Table 6 shows the data of 30 batches with 5 product types and batch sizes.

For the instance with W workers, we use the following data set from Tables 3-6: the entire Table 3, the first W rows of Tables 4 and 5, and the entire Table 6.

6.2. Hardware and software specifications

The developed exact algorithm and the Enumeration were coded in C# and executed on an Intel Core (TM) i7 with 8 processors of 4 Gigahertz under Windows 7 using 4.0 Giga of RAM.

6.3. Performance analysis of the improved exact algorithm

Fig. 4 shows 17 Pareto-optimal solutions, 47,293 feasible solutions, 39,943 solutions dominated by *Intra-SSB*-I, 5829 solutions dominated by *Closest-Ideal-Point*, and 1521 final solutions to performing ε-constraint for the instance with 7 workers.

Table 7 shows the performances of the improved exact algorithm for different instances. From Table 7, we can see that the improved exact algorithm has a better performance than the ε-constraint method because of cutting off approximately 97% non-Pareto-optimal solutions before running ε-constraint.

The time complexity of the improved exact algorithm is $O(2N + 2|SF| + P|SFF|)$, as shown in Table 2. Therefore, the largest solved case is same as that of the single objective model. The largest case that can be solved by the improved exact algorithm is the instance with 10 workers, because there are 102,247,563 solutions in the instance.

6.4. Comparison of balancing and makespan of assembly line and that of *seru* system

Makespan is an important performance (Lin & Ying, 2016) for a production system. Therefore, we compare balancing and makespan of line and that of *seru* system. The instance with nine workers is taken as the compared case, and *Intra-SSB* and makespan of Pareto-optimal solutions of *SSBP* and are shown in Fig. 5.

In Fig. 5, the balancing of the corresponding assembly line is calculated according to SALBP-E (i.e., $\frac{1}{M}\sum_{m=1}^{M}\frac{\sum_{i=1}^{W}T_{mi}}{\max_i(T_{mi})*W}$, see Property 3), and makespan of the assembly line is calculated as: *makespan* of line $= \sum_{m=1}^{M}(SL_m + \sum_{i=1}^{W}TM_{mi} + (B_m - 1)\max_i(TM_{mi})$, where SL_m is the setup time of batch m in the assembly line (SL_m is set as 2.2), and $\max_i(TM_{mi})$ expresses the maximum task time of workers in the assembly line processing batch m.

Fig. 5 shows that except *Inter-SSB*-I all the Pareto-optimal solutions' *Inter-SSBs* are better than the corresponding assembly line's balancing, meaning that *seru* system can improve the balancing of workers than an assembly line. Moreover, the makespans of all the Pareto-optimal solutions are better than that of the corresponding assembly line, meaning that *seru* system can obtain a better makespan than the corresponding assembly line.

Table 4
The average task time of workers for product types (TP_{ni}).

Product	Worker									
	1	2	3	4	5	6	7	8	9	10
1	1.656	1.71	1.782	1.854	1.728	1.818	1.872	1.764	1.746	1.764
2	1.728	1.746	1.818	1.926	1.836	1.98	1.926	1.836	1.854	1.908
3	1.872	1.962	1.89	1.962	1.89	1.98	1.962	1.98	2.016	2.034
4	1.962	2.016	1.962	2.016	1.98	2.07	2.106	1.998	2.142	2.124
5	2.16	2.124	2.178	2.25	2.124	2.214	2.232	2.16	2.268	2.304

Table 5
The coefficient of influencing level of doing multiple tasks 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

Table 6
The data of batches.

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

7. Conclusions and futures

The study is originally motivated by *seru* applications of Sony and Canon. The objective of this study is to investigate the fundamental principles of *seru* system balancing and to explore how to improve the balancing of workers and makespan by implementing *Seru* production.

The contributions are summarized as follows. Firstly, for a *seru*, we define *SB* to describe workloads balancing of the workers in the *seru*; for a *seru* system containing more *serus*, we define and formulate *Intra-SSB* to evaluate workloads balancing of all workers in the *seru* system, and *Inter-SSB* to evaluate workloads balancing of *serus* in the *seru* system. Secondly, we theoretically develop the lower and upper bounds of *Intra-*

Table 7
Performances of the improved exact algorithm for different instances.

W	5	6	7	8	9
Number of all solutions	541	4683	47,293	545,835	7,087,261
Solutions cut by Intra-SSB-I	383	2391	39,943	486,309	6,375,916
Cut ratio using Intra-SSB-I	78%	51%	84%	89%	90%
Solutions cut by Intra-SSB-I and Closest-Ideal-Point	477	3837	45,772	531,227	6,854,164
Cut ratio using Intra-SSB-I and Closest-Ideal-Point	88%	82%	97%	97%	97%
Time of ε-constraint method (ss)	0.001	0.005	0.058	1.126	18.603
Time of our improved ε-constraint method (ss)	0.001	0.002	0.016	0.138	1.887
Number of Pareto-optimal solutions	12	13	17	31	39

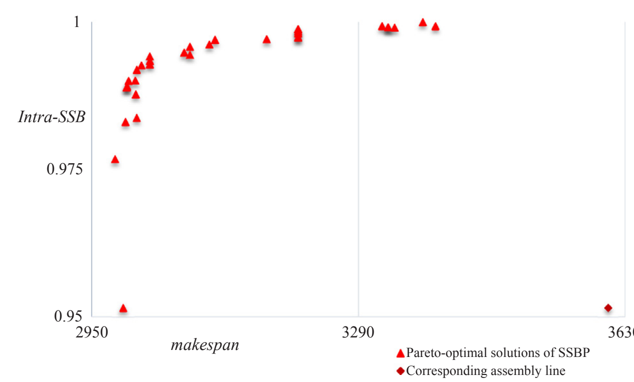

Fig. 5. *Intra-SSB* and makespan in the instance with 9 workers.

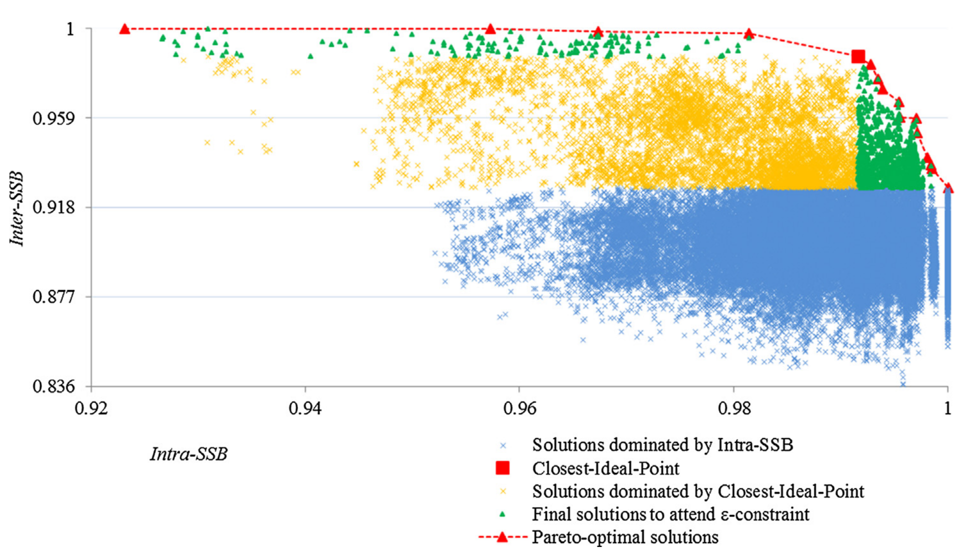

Fig. 4. Result obtained by the improved exact algorithm for the instance with 7 workers.

SSB and Inter-SSB respectively. Thirdly, we define and formulate *SSBP* as a bi-objective mathematical model with maximizing *Intra-SSB* and maximizing *Inter-SSB* simultaneously and clarify the property of solution space of *SSBP*. Fourthly, we develop an improved exact algorithm based on ε-constraint to obtain the Pareto-optimal solutions of *SSBP*.

There are still a lot of research problems. A thorough unanswered problem list can be found in Yin et al. (2017), such as partially multi-skilled workers (i.e., a worker cannot perform all tasks in a *seru*), different products needing different tasks, cost of duplication of equipment, human and psychology factors, carbon emission reduction by *seru* production, game mechanism for *seru* system, computational method for the profit brought by *seru* production and profit distribution, environmental considerations in different countries, and so on. In addition, the noise factors should be considered in the model of *seru* system balancing and others models of *Seru* production because of many noises in the manufacture world.

Acknowledgement

This research is funded by the National Natural Science Foundation of China (71420107028, 71571037, 71571156, and 71601089), the Liaoning Economic and Social Development Foundation (L16BGL020), and the Liaoning Social Science Planning Foundation (20181s1ktqn-027).

References

Aase, G. R., Schniederjans, M. J., & Olson, J. R. (2003). U-OPT: An analysis of exact U-shaped line balancing procedures. *International Journal of Production Research, 41*(17), 4185–4210.

Battaïa, O., & Dolgui, A. (2012). Reduction approaches for a generalized line balancing problem. *Computers & Operations Research, 39*(10), 2337–2345.

Baybars, İ. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science, 32*(8), 909–932.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715.

Bentaha, L. M., Battaïa, O., & Dolgui, A. (2014). A sample average approximation method for disassembly line balancing problem under uncertainty. *Computers & Operations Research, 51*, 111–122.

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research, 183*(2), 674–693.

Chica, M., Bautista, J., Cordón, Ó., & Damas, S. (2016). A multi-objective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand. *Omega, 58*, 55–68.

Esmaeilbeigi, R., Naderi, B., & Charkhgard, P. (2015). The type E simple assembly line balancing problem: A mixed integer linear programming formulation. *Computers & Operations Research, 64*, 168–177.

Fu, Y. P., Wang, H. F., Huang, M., & Wang, J. W. (2017). A decomposition based multiobjective genetic algorithm with adaptive multipopulation strategy for flowshop scheduling problem. *Natural Computing*. http://dx.doi.org/10.1007/s11047-016-9602-1.

Fu, Y., Ding, J. L., Wang, H. F., & Wang, J. W. (2017). Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Applied Soft Computing*. http://dx.doi.org/10.1016/j.asoc.2017.12.009.

Helgeson, W. B., & Birnie, D. P. (1961). Assembly line balancing using the ranked positional weighting technique. *Journal of Industrial Engineering, 12*, 394–398.

Kaku, I., Gong, J., Tang, J., & Yin, Y. (2009). Modeling and numerical analysis of line-cell conversion problems. *International Journal of Production Research, 47*(8), 2055–2078.

Klein, R., & Scholl, A. (1996). Maximizing the production rate in simple assembly line balancing—a branch and bound procedure. *European Journal of Operational Research, 91*(2), 367–385.

Li, L., Chang, Q., Ni, J., & Biller, S. (2009). Real time production improvement through bottleneck control. *International Journal of Production Research, 47*(21), 6145–6158.

Lin, S. W., & Ying, K. C. (2016). Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. *Computers & Industrial Engineering, 99*, 202–209.

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

Muriel, A., Somasundaram, A., & Zhang, Y. (2006). Impact of partial manufacturing flexibility on production variability. *Manufacturing & Service Operations Management, 8*(2), 192–205.

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBP Gen for assembly line balancing. *European Journal of Operational Research, 228*(1), 33–45.

Otto, A., & Otto, C. (2014). How to design effective priority rules: Example of simple assembly line balancing. *Computers & Industrial Engineering, 69*, 43–52.

Pinto, P. A., Dannenbring, D. G., & Khumawala, B. M. (1983). Assembly line balancing with processing alternatives: An application. *Management Science, 29*(7), 817–830.

Plenert, G. (1997). Line balancing techniques as used for just-in-time (JIT) product line optimization. *Production Planning & Control, 8*(7), 686–693.

Ramezanian, R., & Ezzatpanah, A. (2015). Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem. *Computers & Industrial Engineering, 87*(C), 74–80.

Salveson, M. E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering, 6*(3), 18–25.

Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*(3), 666–693.

Stecke, K. E., Yin, Y., Kaku, I., & Murase, Y. (2012). *Seru*: The organizational extension of JIT for a super-talent factory. *International Journal of Strategic Decision Sciences, 3*(1), 105–118.

Tekin, E., Hopp, W. J., & Oyen, M. P. V. (2002). Benefits of skill chaining in production lines with cross-trained workers: an extended abstract. *Manufacturing & Service Operations Management, 4*(1), 17–20.

Uğurdağ, H. F., Rachamadugu, R., & Papachristou, C. A. (1997). Designing paced assembly lines with fixed number of stations. *European Journal of Operational Research, 102*(3), 488–501.

Wang, H. F., Fu, Y. P., Huang, M., Huang, G. Q., & Wang, J. W. (2017). A NSGA-II based memetic algorithm for multiobjective parallel non-identical flowshop scheduling problem. *Computers & Industrial Engineering, 113*, 185–194.

Wang, J. W., Dou, R. L., Muddada, R. R., & Zhang, W. J. (2017). Management of a holistic supply chain network for proactive resilience: Theory and case study. *Computers & Industrial Engineering*. http://dx.doi.org/10.1016/j.cie.2017.12.021.

Wang, J. W., Muddada, R. R., Wang, H. F., Ding, J. L., Lin, Y., & Zhang, W. J. (2016). Towards a resilient holistic supply chain network system: concept, review and future direction. *IEEE Systems Journal, 10*(2), 410–421.

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.

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