



## Line-seru conversion towards reducing worker(s) without increasing makespan: models, exact and meta-heuristic solutions

Yang Yu, Wei Sun, Jiafu Tang, Ikou Kaku & Junwei Wang

To cite this article: Yang Yu, Wei Sun, Jiafu Tang, Ikou Kaku & Junwei Wang (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, DOI: 10.1080/00207543.2017.1284359

To link to this article: <https://doi.org/10.1080/00207543.2017.1284359>



Published online: 30 Jan 2017.



Submit your article to this journal [↗](#)



Article views: 120



View related articles [↗](#)




View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)

## Line-seru conversion towards reducing worker(s) without increasing makespan: models, exact and meta-heuristic solutions

Yang Yu<sup>a,b</sup>, Wei Sun<sup>c</sup>, Jiafu Tang<sup>d</sup>, Ikou Kaku<sup>e</sup> and Junwei Wang<sup>b,\*</sup> 

<sup>a</sup>Institute of Systems Engineering, Northeastern University, Shenyang, P.R. China; <sup>b</sup>Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong; <sup>c</sup>Liaoning University College of Business Administration, Liaoning University, Shenyang, P.R. China; <sup>d</sup>College of Management Science and Engineering, Dongbei University of Finance and Economics, Dalian, P.R. China; <sup>e</sup>Faculty of Environmental and Information Studies, Tokyo City University, Tokyo, Japan

(Received 24 July 2016; accepted 21 December 2016)

Compared with the traditional assembly line, *seru* production can reduce worker(s) and decrease makespan. However, when the two objectives are considered simultaneously, Pareto-optimal solutions may save manpower but increase makespan. Therefore, we formulate line-*seru* conversion towards reducing worker(s) without increasing makespan and develop exact and meta-heuristic algorithms for the different scale instances. Firstly, we analyse the distinct features of the model. Furthermore, according to the feature of the solution space, we propose two exact algorithms to solve the small to medium-scale instances. The first exact algorithm searches the solution space from more workers to fewer workers. The second exact algorithm searches the solution space from fewer workers to more workers. The two exact algorithms search a part of solution space to obtain the optimal solution of reducing worker(s) without increasing makespan. According to the variable length of the feasible solutions, we propose a variable-length encoding heuristic algorithm for the large-scale instances. Finally, we use the extensive experiments to evaluate the performance of the proposed algorithms and to investigate some managerial insights on when and how to reduce worker(s) without increasing makespan by line-*seru* conversion.

**Keywords:** *Seru* production; reducing worker(s); without increasing makespan; exact algorithm; variable-length encoding

### 1. Introduction

The *seru* production is widely used in the Japanese electronics industry such as Sony, Canon, Panasonic, NEC, Fujitsu, Sharp and Sanyo (Sakazume 2005; Miyake 2007) as a new production pattern. *Seru* is an assembly unit including several simple equipment and one or more multi-skilled operator(s). In *seru*, workers are multi-skilled (Kaku et al. 2009), because workers need to operate most or all processes of production. Distinguished from the traditional assembly cells, *seru* is a human-centred assembly system, where equipment is less important. A detailed introduction of *seru* system and its managerial mechanism can be found in Yin, Stecke, and Kaku (2008), Stecke et al. (2012), and Zhang et al. (2017).

As Stecke et al. (2012) claimed, with combined strengths from Toyota's lean philosophy and Sony's one-person production organisation, *seru* system is a more productive, efficient and flexible system (Pérez and Bedia 2016) than conveyor assembly line. Benefits from *seru* production include the reductions of makespan, cost, manpower, labour hours and WIP inventories.

Therefore, some companies converted traditional conveyor assembly line into a *seru* system (line-*seru* conversion) to enhance productivity (Kaku et al. 2009; Yu et al. 2017). Kaku et al. (2009) illustrated that line-*seru* conversion can be used to improve makespan and total labour hours (TLH). They used the term of 'line-cell conversion'. Yu et al. (2012) investigated the line-*seru* conversion with minimising makespan and TLH simultaneously, where an assembly conveyor line is converted to a pure *seru* system. Subsequently, Yu et al. (2013) extended to investigate the line-*seru* conversion model with minimising number of workers and makespan simultaneously. They stated the number of workers and makespan can be reduced simultaneously by line-*seru* conversion. However, when the two objectives are considered simultaneously Pareto-optimal solutions may save manpower but increase makespan. Therefore, this paper focuses on evaluating worker reduction without increasing makespan, according to the practical needs of manufactures.

This paper, originally motivated by line-*seru* applications in saving manpower without increasing makespan, has three contributions. Firstly, we formulate the model towards reducing worker(s) without increasing makespan. Secondly,

---

\*Corresponding author. Email: [jwwang@hku.hk](mailto:jwwang@hku.hk)

we analyse the distinct features of the model such as complexity of solution space, feature of solution space, feature of feasible solution space, NP-hard property and the variable length of the feasible solutions. Thirdly, according to the distinct features of solution space and feasible solution space, we propose two exact algorithms to solve the small to medium-scale instances. The first exact algorithm searches the solution space from more workers to fewer workers. The second exact algorithm searches the solution space from fewer workers to more workers. The two exact algorithms search a part of solution space to obtain the optimal solution. According to the variable length of feasible solutions, we propose a novel variable-length encoding heuristic algorithm for the large-scale instances.

## 2. Line-seru conversion towards reducing worker(s) without increasing makespan

### 2.1 Assumption

The considered assumptions in the research are same as those in Yu et al. (2013).

- (1) The types and batches of products are given in advance.
- (2) The cost of duplicating equipment is ignored (Stecke et al. 2012).
- (3) All product types have the same assembly tasks.
- (4) In the assembly line, a worker only performs a single task. Therefore, the total number of tasks equals that of workers (i.e.  $W$ ). In contrast, a worker in a *seru* needs to perform all the tasks in the *seru*.
- (5) The assembly tasks within each *seru* are the same as the ones within the assembly line.
- (6) A product batch is assembled entirely within a *seru*.

### 2.2 Indices

- $i$  Index of workers ( $i = 1, 2, \dots, W$ ).
- $j$  Index of *serus* ( $j = 1, 2, \dots, J$ ).
- $n$  Index of product types ( $n = 1, 2, \dots, N$ ).
- $m$  Index of product batches ( $m = 1, 2, \dots, M$ ).
- $k$  Index of the sequence of product batches assembled in a *seru* ( $k = 1, 2, \dots, M$ ). The maximum of the product batches assembled in a *seru* is  $M$  when all the batches are assembled in the *seru*. Moreover, the numbers of product batches in different *serus* are not fixed. Therefore, for simplicity, we use  $M$  as the upper bound of index  $k$ . Only  $Z_{mjk}$  uses the index of  $k$ . Assume the maximal product batches assembled in *seru*  $j$  is  $M_j$ , then for each  $k \in \{p | M_j \leq p \leq M\}$  in *seru*  $j$ ,  $Z_{mjk} = 0$

### 2.3 Parameters

- $V_{mn} = \begin{cases} 1, & \text{if product type of product batch } m \text{ is } n \\ 0, & \text{otherwise} \end{cases}$
- $B_m$  Size of product batch  $m$ .
- $T_n$  Cycle time of product type  $n$  in the assembly line.
- $\beta_{ni}$  Skill of worker  $i$  for each task of product type  $n$ .
- $SSP_n$  Setup time of product type  $n$  in a *seru*.
- $\eta_i$  Upper bound on the number of tasks in a *seru* for worker  $i$ . Workers in a *seru* are multi-skilled operators and can operate all tasks within a *seru* in this research. If the number of tasks within a *seru* (i.e.  $W$ )  $\leq \eta_i$ , then the task time of worker  $i$  in the *seru* equals the task time of the original assembly line, i.e.  $T_n \beta_{ni}$ .
- $\varepsilon_i$  Worker  $i$ 's coefficient of influencing level of doing multiple tasks. If the number of tasks within a *seru* (i.e.  $W$ )  $> \eta_i$ , then worker  $i$  will cost more task time in doing multiple tasks, i.e.  $T_n \beta_{ni} C_i$ , where  $C_i = 1 + \varepsilon_i(W - \eta_i)$ .

### 2.4 Decision variables

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

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

## 2.5 Variables

- $C_i$  Coefficient of variation of worker  $i$ 's increased task time.  $C_i$  relates to  $W$ ,  $\eta_i$  and  $\varepsilon_i$ , as given in Equation (1). If the number of tasks within a *seru* ( $W$ )  $> \eta_i$ , then worker  $i$  will cost more task time in doing multiple tasks, i.e.  $T_n \beta_{ni} C_i$ , where  $C_i = 1 + \varepsilon_i(W - \eta_i)$ ; otherwise,  $C_i = 1$ , i.e. the task time of worker  $i$  in the *seru* equals the task time of the original assembly line, i.e.  $T_n \beta_{ni}$ .
- $TC_m$  Assembly task time of product batch  $m$  per tasks in a *seru*.  $TC_m$  is calculated by the average task time of workers in the *seru*, as expressed in Equation (2). In Equation (2),  $\sum_{n=1}^N V_{mn} T_n \beta_{ni} C_i$  is the task time of worker  $i$  processing batch  $m$ ;  $\sum_{n=1}^N \sum_{i=1}^W \sum_{j=1}^J \sum_{k=1}^M V_{mn} T_n \beta_{ni} C_i X_{ij} Z_{mjk}$  is the total task time of all workers in the *seru* assembling batch  $m$ ; and  $\sum_{i=1}^W \sum_{j=1}^J \sum_{k=1}^M X_{ij} Z_{mjk}$  is the total number of workers in the *seru* assembling batch  $m$ .
- $FCB_m$  Begin time of product batch  $m$  in a *seru*.  $FCB_m$  is the aggregation of flow time and setup time of the previous product batches in the *seru* assembling batch  $m$ , as shown in Equation (3). In Equation (3), given batch  $m$ , then *seru*  $j$  and sequence  $k$  can be determined by  $Z_{mjk} = 1$ . Thus, all the previous batches assembled in *seru*  $j$  can be determined, i.e.  $\forall m' | \sum_{k'=1}^{k-1} Z_{m'jk'} = 1$ . Subsequently, the aggregation of flow time and setup time of all the previous product batches assembled in the *seru* is expressed as  $FCB_m = \sum_{m'=1}^{m-1} (FC_{m'} + SC_{m'})$ .
- $SC_m$  Setup time of product batch  $m$  in a *seru*.  $SC_m$  is considered only when two different types of products are processed consecutively, as shown in Equation (4).
- $FC_m$  Flow time of product batch  $m$  in a *seru*, as expressed in Equation (5).

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

$$TC_m = \frac{\sum_{n=1}^N \sum_{i=1}^W \sum_{j=1}^J \sum_{k=1}^M V_{mn} T_n \beta_{ni} C_i X_{ij} Z_{mjk}}{\sum_{i=1}^W \sum_{j=1}^J \sum_{k=1}^M X_{ij} Z_{mjk}} \quad (2)$$

$$FCB_m = \sum_{m'=1}^{m-1} (FC_{m'} + SC_{m'}), \quad \forall m' | \sum_{k'=1}^{k-1} Z_{m'jk'} = 1, j, k | Z_{mjk} = 1 \quad (3)$$

$$SC_m = \begin{cases} SSP_n V_{mn}, & V_{mn} = 1, V_{m'n} = 0 \\ 0, & V_{mn} = V_{m'n} = 1 \end{cases}, \quad m' | Z_{m'j(k-1)} = 1, j, k | Z_{mjk} = 1 \quad (4)$$

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

## 2.6 Evaluated performances

The number of workers and makespan are used to evaluate a *seru* system's performances in this research.

- (1) The number of workers ( $NW$ ) is the total number of workers left in a *seru* system.

$$NW = \sum_{j=1}^J \sum_{i=1}^W X_{ij} \quad (6)$$

(2) Makespan of a *seru* system is the due time of the last completed product batch and can be expressed as:

$$\text{makespan of a } seru \text{ system} = \max_{m=1}^M (FCB_m + SC_m + FC_m) \quad (7)$$

## 2.7 Constraints

The following constraints are considered in this research.

(1) Worker-reduction constraint

$$\sum_{j=1}^J X_{ij} \leq 1, \quad \forall i \quad (8)$$

$$\sum_{j=1}^J \sum_{i=1}^W X_{ij} < W \quad (9)$$

In line-*seru* conversion towards reducing worker(s), worker  $i$  is either left in the *seru* system or removed, as shown in Equation (8). Equation (9) means at least 1 worker is saved.

(2) *Seru*-formation constraint

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

Equation (10) ensures every formatted *seru* contains at least 1 worker and at most  $W - 1$  workers.

(3) *Seru*-load constraint

$$\sum_{j=1}^J \sum_{k=1}^M Z_{mjk} = 1, \quad \forall m \quad (11)$$

$$\sum_{m=1}^M \sum_{k=1}^M Z_{mjk} = 0, \quad \left( \forall j \mid \sum_{i=1}^W X_{ij} = 0 \right) \quad (12)$$

Equation (11) guarantees a product batch is only loaded to a *seru*. Equation (12) guarantees a product batch must be assigned to the *seru* with at least 1 worker.

(4) Makespan constraint

$$\text{makespan of } seru \text{ system} \leq \text{makespan of the line} \quad (13)$$

Equation (13) means that *seru* system's makespan is not more than the line's.

## 2.8 Formulations of line-*seru* conversion towards reducing worker(s)

We formulate the following models based on the above evaluated performances and constraints.

(1) Model of Min-NW: min(6), s.t. (8)–(12).

Model of Min-*NW* is to minimise the number of workers in the *seru* system (i.e. Equation (6)). Therefore, the objective is expressed as min(6) and the constraints include worker-reduction constraint, *seru*-formation constraint and *seru*-load constraint (i.e. Equations (8)–(12)).

If the makespan constraint is not considered, then model of Min-*NW* is meaningless because the minimal *NW* is zero. Consequently, we formulate the line-*seru* conversion towards reducing worker(s) without increasing makespan.

(2) Model of Min-*NW* with makespan constraint: min(6), s.t. (8)–(12), (13).

When makespan constraint (i.e. Equation (13)) is considered, model of Min-*NW* with makespan constraint is formulated as above.

(3) Bi-objective model of Min *NW* and makespan: min((6) and (7)), s.t. (8)–(12).

Bi-objective model of Min *NW* and makespan is to minimise *NW* and makespan simultaneously. Therefore, the bi-objective is expressed as min((6) and (7)).

## 2.9 Previous research on bi-objective model of Min *NW* and makespan

Bi-objective model of Min *NW* and makespan was formulated and solved by the exact algorithm in Yu et al. (2013). However, there are three limitations: (1) Pareto-optimal solutions of the bi-objective model may reduce worker(s) but increase makespan; (2) the complexity of the bi-objective model is larger, i.e.  $O(2N^2)$ , where 2 expresses two objectives and  $N$  is the number of all feasible solutions (Deb et al. 2002). Even though the existing exact bi-objective algorithms (e.g.  $\varepsilon$ -constraint method (Haimes et al. 1971) and Parallel partitioning method (PPM) (Lemesre, Dhaenens, and Talbi 2007)) are used, all the feasible solutions need to be enumerated; and (3) the exact algorithm solves only the small-scale instances. Therefore, some metaheuristic algorithms on multi-objective problem (Lin et al. 2015; Liu 2015; Tang et al. 2015; Wang et al. 2016a, 2016b; Fu et al., 2016) were developed to solve the related larger-scale problems.

Considering the above weaknesses, this paper converts the bi-objective model of Min *NW* and makespan into the single-objective line-*seru* conversion towards reducing worker(s) without increasing makespan, i.e. model of Min-*NW* with makespan constraint, because a company does not consider the situation under which workers are saved but makespan increases. Subsequently, according to the distinct features of the model, we propose two exact algorithms to solve the small to medium-scale instances without searching the whole solution space. Moreover, according to the distinct features of the model, we propose a heuristic algorithm to solve the large-scale instances.

## 3. Features of line-*seru* conversion towards reducing worker(s) without increasing makespan

Line-*seru* conversion towards reducing worker(s) includes three decision processes, i.e. worker reduction, *seru* formation and *seru* load: (1) worker reduction determines which workers are left in the *seru* system, as shown in worker-reduction constraint; (2) *seru* formation determines how many *serus* to be constructed and how to assign the left workers into *serus*, as shown in *seru*-formation constraint. Both worker reduction and *seru* formation are decided by decision variable  $X_{ij}$ ; and (3) *seru* load determines how to assign batches to *serus* and is decided by decision variable  $Z_{mjk}$ , as shown in *seru*-load constraint.

### 3.1 Solution space complexity of worker reduction

For the line with  $W$  workers, let  $r$  be the number of reduced workers and so there are  $C_W^r$  solutions. Obviously,  $r$  ranges from 1 to  $W - 1$ . Thus, the total number of solutions of worker reduction ( $R(W)$ ) is:

$$R(W) = \sum_{r=1}^{W-1} C_W^r \quad (14)$$

$R(1)$  to  $R(10)$  are 0, 2, 6, 14, 30, 62, 126, 254, 510 and 1022, respectively.

### 3.2 Solution space complexity of seru formation towards reducing worker(s)

For the line with  $W$  workers to reduce  $r$  worker(s), the number of solutions of *seru* formation ( $F(W, r)$ ) is:

$$F(W, r) = C_W^r \sum_{J=1}^{W-r} S(W - r, J) \quad (15)$$

where  $S(W-r, J)$  is the number of solutions of assigning  $W-r$  workers into  $J$  *serus* and equals the Stirling numbers of the second kind (Rennie and Dobson 1969). Since the number of *serus* in a *seru* system, i.e.  $J$  ranges from 1 to  $W-r$ ,  $\sum_{J=1}^{W-r} S(W-r, J)$  expresses the number of solutions of assigning  $W-r$  workers into a *seru* system with  $J$  *serus*.

By combining Equation (14) with Equation (15), for the line with  $W$  workers to reduce worker(s), the number of solutions of *seru* formation ( $F(W)$ ) is:

$$F(W) = \sum_{r=1}^{W-1} C_W^r \sum_{J=1}^{W-r} S(W-r, J) \quad (16)$$

### 3.3 Solution space complexity of seru load towards reducing worker(s)

Given a *seru* formation, the number of solutions ( $L$ ) of *seru* load towards reducing worker(s) can be expressed by the number of *serus* ( $J$ ) formatted in *seru* formation.

#### 3.3.1 Solution space complexity of seru load towards reducing worker(s) without given dispatching rule

**Property 1.** Given the *seru* formation with  $J$  *serus*, without given a dispatching rule,  $L = J^M$ .

**Explanation.** Without given a dispatching rule in *seru* load, any product batch ( $M$ ) can be assigned into any *seru* ( $J$ ).

Obviously, *seru* load is a scheduling problem and NP-hard (Liao, Lee, and Tsai 2015). For simplicity, therefore, earlier researches (such as Yu et al. 2012, 2013) used the FCFS (first come first served) rule. However, the different scheduling rules produce different solution space complexities of *seru* load towards reducing worker(s). Yu et al. (2016) investigated the complexities of *seru* load for ten different scheduling rules, divided into two categories: dispatching rules related to *seru* sequence (*RSS*) and dispatching rules unrelated to *seru* sequence (*USS*). FCFS belongs to *RSS*. SPT (shortest process time) belongs to *USS*.

Workers are reduced in the step of *seru* formation. Therefore, the *seru* load towards reducing worker(s) is same as that of *seru* load without reducing worker(s) which investigated by Yu et al. (2016). We can obtain solution space complexity of *seru* load towards reducing worker(s) with *RSS* and *USS* as follows.

#### 3.3.2 Solution space complexity of seru load towards reducing worker(s) with RSS

**Property 2.** Given the *seru* formation with  $J$  *serus*, if *seru* load with *RSS* and  $M < J$ ,  $L = C_J^M P_M^M$ .

**Property 3.** Given the *seru* formation with  $J$  *serus*, if *seru* load with *RSS* and  $M \geq J$ ,  $L = J! = P_J^J$ .

Properties 2 and 3 were clarified in *seru* load without reducing worker(s) of Yu et al. (2016).

#### 3.3.3 Solution space complexity of seru load towards reducing worker(s) with USS

**Property 4.** Given a *seru* formation, if *seru* load with *USS*,  $L = 1$ .

**Explanation.** *USS* means that the *seru* sequence does not influence the *seru* load result. Therefore, given *seru* formation and batches, the *seru* load result is only.

### 3.4 Solution space complexity of line-seru conversion towards reducing worker(s)

The total number of solutions  $T(W)$  can be expressed by combining the complexities of worker reduction, *seru* formation and *seru* load.

#### 3.4.1 Solution space complexity of line-seru conversion towards reducing worker(s) without a dispatching rule

**Theorem 1.** Given the *seru* formation with  $J$  *serus*, without given dispatching rule,  $T(W) = \sum_{r=1}^{W-1} C_W^r \sum_{J=1}^{W-r} S(W-r, J) * (J^M)$ .

**Proof.** Combine Equation (16) with Property 1.

### 3.4.2 Solution space complexity of line-seru conversion towards reducing worker(s) with RSS

**Theorem 2.** Given the seru formation with  $J$  serus, if seru load with RSS and  $M < J$ ,  $T(W) = \sum_{r=1}^{W-1} C_W^r \sum_{J=1}^{W-r} S(W-r, J) * (P_J^M)$ .

**Proof.** Combine Equation (16) with Property 2.

**Theorem 3.** Given the seru formation with  $J$  serus, if seru load with RSS and  $M \geq J$ ,  $T(W) = \sum_{r=1}^{W-1} C_W^r \sum_{J=1}^{W-r} S(W-r, J) * (P_J^J)$ .

**Proof.** Combine Equation (16) with Property 3.

Obviously,  $T(W)$  increases exponentially with the number of workers ( $W$ ) in the line. The complexity of line-seru conversion towards reducing worker(s) with FCFS and  $M \geq J$  was clarified by Yu et al. (2013).

### 3.4.3 Solution space complexity of line-seru conversion towards reducing worker(s) with USS

**Theorem 4.** Given the seru formation with  $J$  serus, if seru load with USS,  $T(W) = \sum_{r=1}^{W-1} C_W^r \sum_{J=1}^{W-r} S(W-r, J)$ .

**Proof.** Combine Equation (16) with Property 4.

## 3.5 Feasible solution space of the line-seru conversion towards reducing worker(s) without increasing makespan

The above section describes the solution space complexity of line-seru conversion towards reducing worker(s). However, for reducing worker(s) without increasing makespan, the solution must meet the makespan constraint, i.e. Equation (13). Therefore, the solution space of reducing worker(s) without increasing makespan must be not larger than that of line-seru conversion towards reducing worker(s).

Three instances of solution space of reducing worker(s) are shown in Figure 1, where 540, 4682 and 47,292 solutions satisfies the constraints of worker-reduction, seru-formation and seru-load for the instances with 5, 6 and 7 workers, respectively. All solutions are generated by enumeration with FCFS rule. The detailed data used in three instances are described in section 5.1.

Figure 1(b) and (c) show that some solutions satisfy makespan constraint. This means seru production can be used to reduce worker(s) without increasing makespan.

Figure 2 shows that 3592 solutions satisfy makespan constraint for the instance with 7 workers and 65,482 solutions satisfy makespan constraint for the instance with 8 workers. All the satisfying solutions reduce 1 worker.

From Figure 2, we can observe that the solutions satisfying makespan constraint are much fewer than the solutions of reducing worker(s). To investigate the feature of solution space complexity of line-seru conversion towards reducing worker(s) without increasing makespan, we define the index of  $P_{FS}$ , as shown in Equation (17).

$$P_{FS} = \frac{\text{Number of feasible solutions}}{\text{Number of solutions}} \quad (17)$$

where *feasible solution* represents the solution satisfying makespan constraint; *solution* represents the solution of reducing worker(s).

For the instances with 6, 7 and 8 workers,  $P_{FS}$  are 3.4, 7.6 and 12.0%, respectively.

## 3.6 Features of line-seru conversion towards reducing worker(s) without increasing makespan

Based on the above analysis, we can obtain the following features.

**Feature 1.** Line-seru conversion towards reducing worker(s) without increasing makespan is an NP-hard problem.

**Explanation.** Line-seru conversion towards reducing worker(s) without increasing makespan includes seru formation which is NP-hard.

**Feature 2.** The solutions of line-seru conversion towards reducing worker(s) without increasing makespan may be much fewer than the solutions of line-seru conversion towards reducing worker(s).

**Feature 3.** The number of the optimal solutions may be more than 1.

**Explanation.** In Figure 2(a), there are 3592 feasible solutions for the instance with 7 workers. All the solutions have 6 workers, and so any solution is optimal for the instance.

**Feature 4.** The feasible solutions of reducing worker(s) without increasing makespan often reduce fewer workers (i.e. more workers are left in the seru system).

According to Feature 4, we propose an exact algorithm by searching solution space from more workers to fewer workers. Thus, the solution space can be effectively reduced. The exact algorithm is developed in section 4.1.



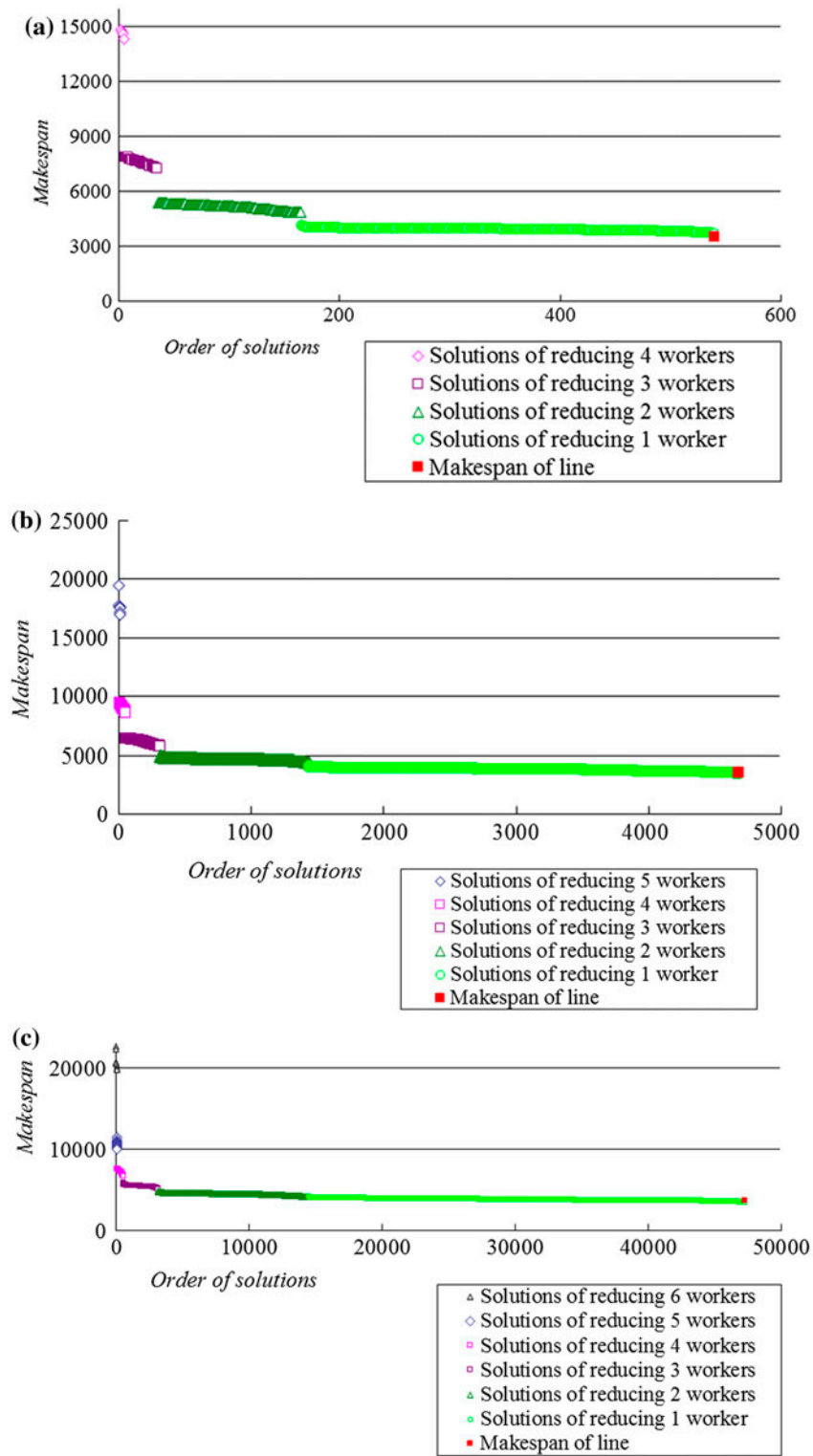


Figure 1. Solution space of reducing worker(s) for the instance with 5, 6 and 7 workers.

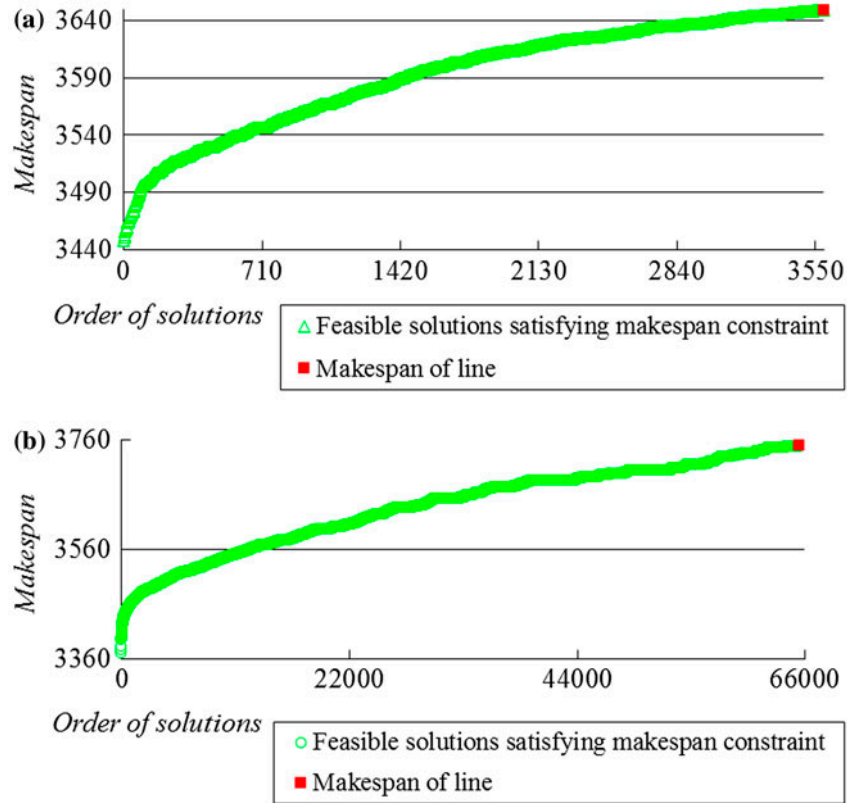


Figure 2. Solutions satisfying makespan constraint for the instance with 7 and 8 workers.

**Feature 5.** The first found feasible solution by searching the solution space from fewer workers to more workers must be optimal.

Feature 5 is proved in Theorem 6. According to Feature 5, we propose another exact algorithm to fast find one optimal solution. The exact algorithm is developed in section 4.2.

**Feature 6.** The number of workers in the solutions of reducing worker(s) is variable.

**Explanation.** As shown in Figure 1(c), the number of workers in the solutions of reducing worker(s) ranges from 1 to 6.

According to Feature 6, we propose a novel variable-length encoding heuristic algorithm for large-scale instances. The algorithm is developed in section 4.3.

#### 4. Exact and meta-heuristic solutions for line-seru conversion towards reducing worker(s) without increasing makespan

Line-seru conversion towards minimising the number of worker(s) and makespan was solved by the exact bi-objective algorithm of Yu et al. (2013). However, the Pareto-optimal solutions which reduce worker(s) but increase makespan are meaningless. Therefore, we propose the exact single-objective algorithms to solve line-seru conversion towards reducing worker(s) without increasing makespan.

##### 4.1 Exact algorithm by searching solution space from more workers to fewer workers

According to Feature 4, we propose an exact algorithm. The highlight is to search solution space from more workers to fewer workers, instead of searching the whole solution space.

For the instance with  $W$  workers to reduce worker(s) without increasing makespan, when obtaining a feasible solution in the solution space with  $b$  workers, the algorithm will go to the solution space with  $b - 1$  workers to continue to search the feasible solution. If the algorithm cannot find any feasible solution, then the feasible solution obtained in the solution space with  $b$  workers is optimal. The flow chart of the algorithm is depicted in Figure 3(a).

**Theorem 5.** *By searching solution space from more workers to fewer workers, the solution obtained by the algorithm must be optimal.*

**Proof.** Suppose the obtained solution reduces  $r$  workers. Therefore, we cannot find any feasible solution in the solution space of reducing  $(r + 1)$  workers. If the obtained solution is not optimal, then the optimal solution must be in the solution space of reducing more  $(r + 1)$  workers. Assume the optimal solution is in the solution space of reducing  $(r + t)$  ( $t > 1$ ) workers and the optimal solution is expressed as  $\{S_1, S_2, \dots, S_J\}$ . Obviously, the makespan of  $\{S_1, S_2, \dots, S_J\}$  is not more than that of the line. At this time, if we add the  $(t - 1)$  worker(s) into  $\{S_1, S_2, \dots, S_J\}$  to form a new solution and then the makespan of the new solution must be not more than that of  $\{S_1, S_2, \dots, S_J\}$  because if the added worker(s) do not assemble any product then the makespan does not increase. That is to say, if the optimal solution is in the solution space of reducing  $(r + t)$  ( $t > 1$ ) workers, then the solution space of reducing  $(r + 1)$  workers must contain the solution whose makespan is not more than that of the line. However, this is contradict with the previous assumption. Thus, the solution obtained by the algorithm must be optimal.

However, the worst computational time of the exact algorithm may be very high, because the solution space with more workers is large. For the instance with  $W$  workers to reduce  $r$  worker(s), the worst computational time of the exact algorithm is described as follows:

$$\sum_{l=1}^r \left\{ C_W^{W-l} \sum_{J=1}^{W-l} S(W-l, J) - f_{W-l} + 1 \right\} + C_W^{W-(r-1)} \sum_{J=1}^{W-(r-1)} S(W-(r-1), J) \quad (18)$$

where  $f_{W-l}$  is the number of the feasible solutions with  $W - l$  workers;  $C_W^{W-l} \sum_{J=1}^{W-l} S(W-l, J) - f_{W-l} + 1$  represents the worst computational time of searching the solution space with  $W - l$  workers.

Equation (18) indicates that the worst computational time will be very high for reducing more than 1 worker. Therefore, we propose another exact algorithm for reducing more workers.

#### 4.2 Exact algorithm by searching solution space from fewer workers to more workers

The highlight of the exact algorithm is to search solution space from fewer workers to more workers.

For the instance with  $W$  workers to reduce worker(s) without increasing makespan, the algorithm first searches the solution space with 1 worker. If a feasible solution is obtained, then the feasible solution is optimal and  $W - 1$  workers are reduced; otherwise, the algorithm continues to search the solution space with 2 workers. In the algorithm by searching solution space from fewer workers to more workers, the first obtained feasible solution is optimal. If the first obtained feasible solution has  $b$  workers, then  $W - b$  workers are reduced without increasing makespan. The flow chart of the algorithm is depicted in Figure 3(b).

**Theorem 6.** *By searching solution space from fewer workers to more workers, the solution obtained by the algorithm must be optimal.*

**Proof.** Since the solution space is searched from fewer workers to more workers, the number of workers in the obtained solution must be minimal in the all feasible solutions. Moreover, the solution's makespan must be less than the line's. Therefore, the solution must be one of the optimal solutions.

For the case with  $W$  workers to reduce  $r$  worker(s), the worst computational time of the exact algorithm by searching the solution space from fewer workers to more workers is described as:

$$\sum_{l=1}^{W-r-1} C_W^l \sum_{J=1}^l S(l, J) + \left\{ C_W^{W-r} \sum_{J=1}^{W-r} S(W-r, J) - f_{W-r} + 1 \right\} \quad (19)$$

where  $f_{W-r}$  is the number of feasible solutions in the solutions with  $W - r$  workers;  $C_W^{W-r} \sum_{J=1}^{W-r} S(W-r, J) - f_{W-r} + 1$  represents the worst computational time of searching the solution space with  $W - r$  workers.

#### 4.3 Variable-length encoding heuristic for large-scale instances

To solve the large-scale instances, we propose a novel variable-length encoding heuristic algorithm (VLEH) according to Feature 6.

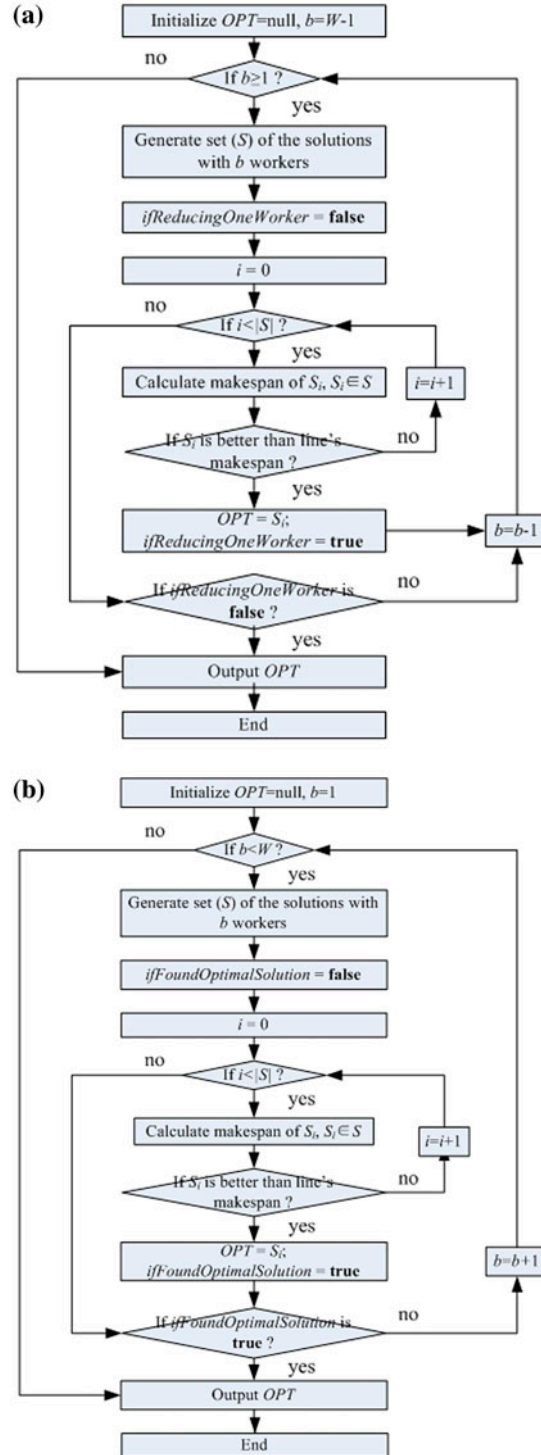


Figure 3. Flow charts of the three algorithms.

#### 4.3.1 Solution representation of line-seru conversion without reducing worker(s)

Yu et al. (2012) proposed a sequence encoding method to represent the *seru* formation without reducing worker(s), where the solution with  $W$  workers is represented by a vector with  $2W - 1$  elements. An element represents a worker ID if the value is not more than  $W$ ; otherwise, represents the separating character. Two fixed separating characters are in

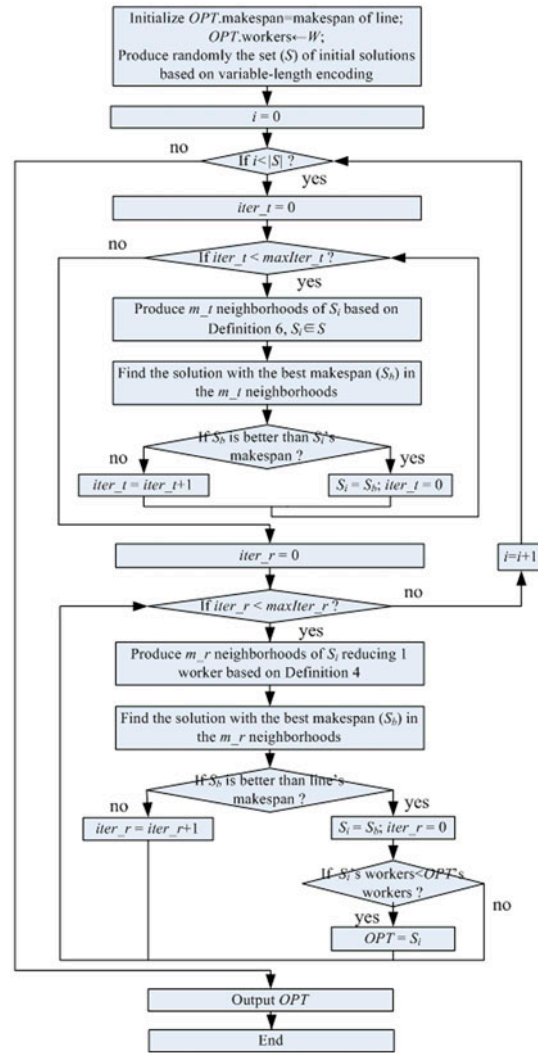


Figure 3. (Continued).

the start and end locations of the vector. If there is at least one worker between two separating characters, then a *seru* is formed. For example of the following two solutions with 6 workers.

Solution 1: 1 7 2 8 3 9 4 10 5 11 6

Solution 2: 1 3 8 7 6 4 5 9 10 11 2

In the two solutions, 7, 8, 9, 10 and 11 are separating characters. Therefore, in Solution 1, 6 *serus* are constructed where worker #1 is in *seru* 1, worker #2 is in *seru* 2 and so on. In Solution 2, 3 *serus* are set up where workers #1 and #3 are in *seru* 1, workers #6, #4 and #5 are in *seru* 2 and worker #2 is in *seru* 3.

Thus, the two solutions are expressed as:

Solution 1:  $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$ , and

Solution 2:  $\{\{1, 3\}, \{6, 4, 5\}, \{2\}\}$

#### 4.3.2 Variable-length encoding solution to represent reducing worker(s)

Based on the above method, for the line with  $W$  workers, we use a vector whose length is less than  $2W - 1$  to represent reducing worker(s). In the vector, the number ( $ne$ ) of elements representing worker is less than  $W$ . Thus, vector means that  $W-ne$  workers are reduced.

**Definition 1** (*Variable-length encoding solution for reducing worker(s)*). Variable-length encoding solution for reducing worker(s) is the solution in which one or more elements representing worker are removed.

For example, five variable-length encoding solutions for reducing worker(s) show as follows. They represent reducing 1, 2, 3, 4 and 5 worker(s), respectively.

Solution 1 of variable-length encoding: 7 2 8 3 9 4 10 5 11 6

Solution 2 of variable-length encoding: 1 7 8 9 4 10 5 11 6

Solution 3 of variable-length encoding: 1 7 2 8 9 10 5 11

Solution 4 of variable-length encoding: 1 7 8 9 10 11 6

Solution 5 of variable-length encoding: 7 8 9 4 10 11

Variable-length encoding is used to produce initial solutions. To further find the feasible solutions with fewer workers, we propose a method to reduce 1 worker for the produced variable-length encoding solutions.

**Definition 2** (*Reduce 1 worker*). The method to reduce 1 worker in a given solution is: any element representing worker is removed.

For example of the solution '7 2 8 3 9 4 10 5 11 6', i.e. ' $\{\{2\},\{3\},\{4\},\{5\},\{6\}\}$ ', two solutions of reducing 1 worker are shown as follows. They represent reducing worker #2 and worker #3, respectively.

Solution 1 for reducing 1 worker: 7 8 3 9 4 10 5 11 6

Solution 2 for reducing 1 worker: 7 2 8 9 4 10 5 11 6

#### 4.3.3 Method to search the better makespan

Since the model is to reduce worker(s) without increasing makespan, a method to search the better makespan based on a given solution needs to be developed.

**Definition 3** (*Neighbourhood for searching the better makespan*). Neighbourhood for searching the better makespan is to swapping two unique elements in the sequence of a given solution.

According to the solution representation, nearly half of elements in the sequence are the separating characters. Therefore, if two elements are the separating characters, then the neighbourhood generated by Definition 3 is same as the original solution. In addition, if the two elements are workers in a *seru*, then the neighbourhood generated is same as the original solution. We use the improved neighbourhood strategy.

**Definition 4** (*Improved neighbourhood for searching the better makespan*). Improved neighbourhood for searching the better makespan is that the worker in a *seru* is swapped with any element which is not in the *seru*.

For example of '3 8 7 6 4 5 9 10 11 2', where elements 8, 7, 9, 10 and 11 represent separating characters, and elements 3, 6, 4, 5 and 2 represent workers. The solution, i.e. ' $\{\{3\},\{645\},\{2\}\}$ ', means that worker #1 has been reduced. According to Definition 4, workers #3 and #2 can be swapped with any other element, but workers #6, #4 and #5 can be swapped with any other element except 4 and 5, 6 and 5 and 6 and 4, respectively.

#### 4.3.4 The procedure of variable-length encoding heuristic algorithm (VLEH)

VLEH first produces the set ( $S$ ) of initial solutions based on variable-length encoding. For each initial solutions ( $S_i$ ), VLEH obtains the neighbourhood with the best makespan as the new  $S_i$  by the local search of  $S_i$  based on Definition 4. Subsequently, for  $S_i$ , VLEH obtains the neighbourhood with the best makespan as  $S_b$  by the local search of  $S_i$  reducing 1 worker based on Definition 2. If  $S_b$  is better than  $S_i$ , then  $S_i$  is set as  $S_b$  and continue to search the next  $S_b$  which is better than  $S_i$  by the local search of  $S_i$  reducing 1 worker; otherwise, continue to search the  $S_b$  which is better than  $S_i$  until satisfying the abort condition. In the loop, if  $S_i$  is better the optimal solution ( $OPT$ ), then  $OPT$  is set as  $S_i$ . The flow chart of VLEH is depicted in Figure 3(c).

## 5. Experimental results

Line-*seru* conversion towards reducing worker(s) without increasing makespan includes two NP-hard problems (i.e. *seru* formation and *seru* load). For simplicity, we use the FCFS rule in *seru* load.

All tests were performed on a single processor from an Intel Core(TM) 2 processor at 2.66 GHz under Windows 7 using 3.0 GB of RAM. The two exact algorithms and the proposed VLEH were implemented in C# programming language.

In our experiments, the following parameters were used:  $|S| = 10$ ,  $m_t = 20$ ,  $maxIter_t = 5$ ,  $m_r = 5$  and  $maxIter_r = 3$ .  $|S|$  is the number of initial solutions;  $m_t$  is the number of neighbourhoods for searching the better makespan;  $maxIter_t$  is the maximum iterations of unchanging makespan;  $m_r$  is the number of neighbourhoods for reducing 1 worker;  $maxIter_r$  is the maximum iterations of unchanging number of workers.

### 5.1 Test instances

The used experimental data can be found in Tables 1–5 in Yu et al. (2013) except for  $\eta_i = 15$ . We add some data to test the larger-scale instances. Table 2 and Table 4 in Yu et al. (2013) are added by 5 workers (i.e. workers #11–#15), as shown in Tables 1 and 2, respectively.

For the instance with  $W$  workers, we use the following data-set from: the entire Tables 1 and 5 in Yu et al. (2013) and the first  $W$  rows of Tables 1 and 2.

### 5.2 Performances of the two exact algorithms and VLEH

We repeatedly run the VLEH five times to solve the following instances. The performance comparisons of VLEH to the two exact algorithms are shown in Table 3.

*WR*: The worker(s) reduced by line-*seru* conversion without increasing makespan; *Time* is running time, unit is second;  $EA_1$  is the exact algorithm by searching the solution space from more workers to fewer workers;  $EA_2$  is the exact algorithm by searching the solution space from fewer workers to more workers.

Table 1. The coefficient of influencing level of skill to multiple tasks for workers ( $\varepsilon_i$ ).

Worker	1	2	3	4	5	6	7	8
$\varepsilon_i$	0.18	0.19	0.2	0.21	0.2	0.2	0.2	0.22
Worker	9	10	11	12	13	14	15	
$\varepsilon_i$	0.19	0.19	0.18	0.23	0.24	0.22	0.16	

Table 2. The data of worker's skill ( $\beta_{ni}$ ).

Worker/product	1	2	3	4	5
1	1.02	1.05	1.1	1.05	1.13
2	1.09	1.15	1.16	1.24	1.29
3	0.96	0.98	1.06	1.16	1.22
4	0.94	0.99	1.1	1.09	1.
5	0.96	1.1	1.08	1.07	1.23
6	0.92	0.97	1.12	0.99	1.2
7	1.1	1.13	1.13	1.22	1.27
8	0.98	1.08	1.06	1.3	1.16
9	1.03	1.03	1.13	1.25	1.11
10	0.97	1.14	1.2	1.21	1.22
11	1.04	1.1	1.03	1.12	1.19
12	0.95	1.05	0.99	1.2	1.22
13	0.92	0.98	1.13	1.03	1.27
14	1.08	1.09	1.09	1.18	1.14
15	1.06	1	1.13	1.08	1.19



Table 3. Performance comparisons of the VLEH to the two exact algorithms.

Assemble line		Two exact algorithms				VLEH		
$W$	Makespan	WR	Makespan	Time of $EA_1$	Time of $EA_2$	WR	Makespan	Time (s)
6	3581	1	3469	6.5	8.1	1	3499	17.5
7	3649	1	3447	17.5	9.4	1	3473	19.0
8	3748	1	3368	89.6	21.3	1	3417	22.0
9	3809	2	3780	540	90.8	2	3790	26.9
10	3896	—	—	—	—	2	3748	28.0
11	3955	—	—	—	—	2	3647	30.9
12	4013	—	—	—	—	3	3975	37.7
13	4071	—	—	—	—	3	3868	41.4
14	4131	—	—	—	—	3	3830	44.0
15	4190	—	—	—	—	4	4122	47.5

Table 4. The first five optimal solutions of reduced workers without increasing makespan for 12 workers.

No.	Makespan	Solution
Reduced 1 worker without increasing makespan		
1	3269.468	{{1,261,134},{8,710,519}}
2	3287.66	{{1,261,134},{2,710,519}}
3	3295.144	{{24},{1019},{5,381,112},{6}}
4	3302.872	{{3,811,912,415},{762}}
5	3305.224	{{1,261,174,135},{8109}}
Reduced 2 workers without increasing makespan		
1	3577.753	{{11},{15,436},{781,012}}
2	3595.455	{{119},{15,436},{71,012}}
3	3597.484	{{119},{15,436},{7812}}
4	3597.484	{{119},{15,436},{81,012}}
5	3621.732	{{12},{311},{48},{156},{710}}
Reduced 3 workers without increasing makespan		
1	3968.37	{{4,816,115},{3912}}
2	3974.318	{{12,463},{112,918}}
3	3974.318	{{12,463},{1,129,101}}
4	3975.881	{{11},{61,354},{12,810}}
5	3979.512	{{12,463},{1,121,018}}

From the instances with no more than 9 workers in Table 3, we can see that the  $WR$ s of VLEH are same as those of the two exact algorithms. Moreover, the makespan gaps between VLEH and the two exact algorithms are not larger. For example, the maximum gap is  $1.4\% = (3417-3368)/3417$  in the instance with 8 workers. However, the running time of VLEH is so much better than that of two exact algorithms, especially for the instance with more workers. In addition, the more workers are reduced, the better  $EA_2$  is than  $EA_1$ , because  $EA_2$  does not need to search the solutions with more workers. Moreover, for the instances with more than 9 workers, the two exact algorithms cannot solve, but the propose VLEH can solve in a reasonable time.

### 5.3 Computational results of larger instances

We repeatedly run the VLEH five times to solve the instances with 12 workers to reduce 1, 2 and 3 worker(s), respectively. The results are shown in Table 4. For the instance with 12 workers, no solution reduces more than 3 workers without increasing makespan.

In Table 4, *No.* is the optimal solution number, *makespan* is the makespan of the optimal solution, and *Solution* is the representation of the optimal solution obtained by the proposed VLEH algorithm.



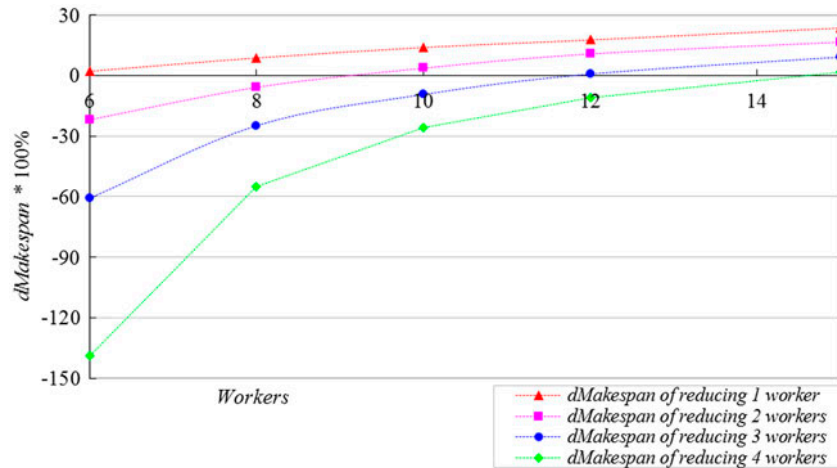


Figure 4. Makespan improvements by line-*seru* conversion towards reducing worker(s).

Table 5 The data of worker's skill for product combination.

Worker/Product combination	{25}	{1345}	{12,345}
1	2.18	4.3	5.35
2	2.44	4.78	5.93
3	2.2	4.4	5.38
4	1.99	4.13	5.12
5	2.33	4.34	5.44
6	2.17	4.23	5.2
7	2.4	4.72	5.85
8	2.24	4.5	5.58
9	2.14	4.52	5.55
10	2.36	4.6	5.74
11	2.29	4.38	5.48
12	2.27	4.36	5.41
13	2.25	4.35	5.33
14	2.23	4.49	5.58
15	2.19	4.46	5.46

#### 5.4 Discussion

We use *dMakespan* to evaluate how much makespan can be improved by the line-*seru* conversion. Obviously, if *dMakespan* is positive, then makespan of the *seru* system is better than that of the line.

$$dMakespan = \frac{\text{makespan of line} - \min \text{makespan of seru system}}{\text{makespan of line}} \quad (20)$$

Figure 4 shows the *dMakespan* (i.e. makespan improvements) by line-*seru* conversion towards reducing worker(s).

(1) The line-*seru* conversion can reduce worker(s) without increasing makespan.

In Figure 4, if *dMakespan* is larger than 0, then at least 1 worker is reduced without increasing makespan.

(2) The *dMakespan* increases with the number of workers.

From Figure 4, we can see that the *dMakespan* increases with the number of workers. For example, the *dMakespans* of reducing 1 worker for the instances with 6, 8, 10, 12 and 15 workers are 2.3, 8.8, 14.0, 18.7 and 23.5%, respectively. This means that the possibility of reducing worker(s) increases with the number of workers.

For reducing 2 workers, only the *dMakespans* of the instances with 10, 12 and 15 workers are larger than zero. This means that the instances with 6 and 8 workers cannot reduce 2 workers without increasing makespan. The *dMakespans* of reducing 2 workers for the instances with 6, 8, 10, 12 and 15 workers are  $-21.8\%$ ,  $-5.7\%$ ,  $3.8\%$ ,  $10.9\%$  and  $16.7\%$ , respectively. This means that the more workers are in the line, the more workers can be reduced without increasing makespan.

(3) The worker(s) with the worst skill level have a higher possibility to be reduced without increasing makespan.

In the first five optimal solutions of reduced 1 worker without increasing makespan, the reduced workers are #2, #8, #7, #10 and #2, respectively. In Table 2 (i.e. the data of worker's skill), we can easily observe that worker #8 has the worst skill for product 4 and worker #10 has the worst skill for product 3. In addition, based on Table 2, we obtain the worker's skill for product combination, as shown in Table 5. In table 5, worker #2 has the worst skill for product combination of {12,345} and worker #7 has the worst skill for product combination of {1345} except worker #2.

In the first five optimal solutions of reduced 2 workers without increasing makespan, the reduced two workers are {29}, {28}, {210}, {27} and {29}, respectively. Obviously, worker #2 is reduced in any solution. In addition, worker #9 has the worst skill for product combination of {1345} except workers #2, #7 and #10 (see Table 5). Therefore, the reduced two workers are probably the combinations of the two workers with the worse skill.

In the first five optimal solutions of reduced 3 workers without increasing makespan, the reduced three workers are {2710}, {5710}, {579}, {279} and {579}, respectively. From Table 5, we can easily observe that worker #5 has the worst skill for product combination of {25} except workers #2, #7 and #10. Also, the reduced three workers are probably the combinations of the three workers with the worse skill.

## 6. Conclusions and future research

Our main contributions in this paper can be summarised as following. First, we formulated a line-*seru* conversion model towards reducing worker(s) without increasing makespan. Second, we clarified the distinct features of the model such as complexity and features of solution space, features of feasible solution space, NP-hard and the variable length of feasible solutions. Third, based on the distinct features, we developed two exact algorithms to solve the small to medium-scale instances. The two exact algorithms search a part of solution space to obtain the optimal solution. According to the variable length of feasible solutions, we proposed a novel variable-length encoding heuristic algorithm for the large-scale instances. Fourth, we proposed some managerial insights on how to reduce worker(s) without increasing makespan by line-*seru* conversion.

The study on the line-*seru* conversion is relatively lacking. The following problem should be investigated in future, such as partially cross-trained workers (i.e. a worker cannot perform all assembly tasks), different products have different assembly tasks, the conversion from assembly line to the hybrid system with line and *serus*, cost of *karakuri* (i.e. duplication of equipment), human and psychology factors.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work was supported by the National Natural Science Foundation of China [grant number 71420107028], [grant number 71571037], [grant number 71601089]; the open project funded by State Key Laboratory of Synthetical Automation for Process Industries [PAL-N201505]; Liaoning Social Science Planning Fund [L16BGL020].

## ORCID

Junwei Wang  <http://orcid.org/0000-0001-8895-2214>

## References

- Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

- Fu, Y. P., H. F. Wang, M. Huang, and J. W. Wang. 2016. "A Decomposition Based Multiobjective Genetic Algorithm with Adaptive Multipopulation Strategy for Flowshop Scheduling Problem." *Natural Computing*.
- Haimes, Y. Y., L. S. Lasdon, D. A. Wismer, Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer. 1971. "On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization." *IEEE Transactions on Systems Man & Cybernetics* 1 (3): 296–297.
- Kaku, I., J. Gong, J. Tang, and Y. Yin. 2009. "Modeling and Numerical Analysis of Line-Cell Conversion Problems." *International Journal of Production Research* 47 (8): 2055–2078.
- Lemesre, J., C. Dhaenens, and E. G. Talbi. 2007. "Parallel Partitioning Method (PPM): A New Exact Method to Solve Bi-objective Problems." *Computers & Operations Research* 34 (8): 2450–2462.
- Liao, C. J., C. H. Lee, and H. T. Tsai. 2015. "Scheduling with Multi-attribute Set-up times on Unrelated Parallel Machines." *International Journal of Production Research* 54 (16): 1–15.
- Lin, S. W., K. C. Ying, W. J. Wu, and Y. Chiang. 2015. "Multi-objective Unrelated Parallel Machine Scheduling: A Tabu-Enhanced Iterated Pareto Greedy Algorithm." *International Journal of Production Research* 54 (4): 1–12.
- Liu, C. H. 2015. "Solving the Bi-objective Optimisation Problem with Periodic Delivery Operations using a Lexicographic Method." *International Journal of Production Research* 54 (18): 2275–2283.
- Miyake, D. I. 2007. "The Shift from Belt Conveyor Line to Work-Cell Based Assembly System to Cope with Increasing Demand Variation and Fluctuation in the Japanese Electronics Industries." *International Journal of Automotive Technology and Management* 6 (4): 419–439.
- Pérez, M. P., and A. M. S. Bedia. 2016. "A Review of Manufacturing Flexibility: Systematising the Concept." *International Journal of Production Research* 54 (10): 1–16.
- Rennie, B. C., and A. J. Dobson. 1969. "On Stirling Numbers of the Second Kind." *Journal of Combinatorial Theory* 7 (2): 116–121.
- Sakazume, Y. 2005. "Is Japanese Cell Manufacturing a New System? A Comparative Study between Japanese Cell Manufacturing and Cell Manufacturing." *Journal of Japan Industrial Management Association* 12: 89–94.
- Stecke, K. E., Y. Yin, I. Kaku, and Y. Murase. 2012. "Seru: The Organizational Extension of JIT for a Super-talent Factory." *International Journal of Strategic Decision Sciences* 3 (1): 105–118.
- Tang, L., L. Zheng, H. Cao, and N. Huang. 2015. "An Improved Multi-objective Genetic Algorithm for Heterogeneous Coverage Rfid Network Planning." *International Journal of Production Research* 54 (8): 1–14.
- Wang, H. F., Y. P. Fu, M. Huang, G. Q. Huang, and J. W. Wang. 2016a. "A Hybrid Evolutionary Algorithm with Adaptive Multi-population Strategy for Multi-objective Optimization Problems." *Soft Computing*. doi:10.1007/s00500-016-2414-5.
- Wang, H. F., Y. P. Fu, M. Huang, and J. W. Wang. 2016b. "Multiobjective Optimisation Design for Enterprise System Operation in the Case of Scheduling Problem with Deteriorating Jobs." *Enterprise Information Systems* 10 (3): 268–285.
- Yin, Y., K. E. Stecke, and I. Kaku. 2008. "The Evolution of Seru Production Systems throughout Canon." *Operations Management Education Review* 2: 27–40.
- Yu, Y., J. Gong, J. Tang, Y. Yin, and I. Kaku. 2012. "How to Do 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., J. Tang, W. Sun, Y. Yin, and I. Kaku. 2013. "Reducing Worker(S) by Converting Assembly Line into a Pure Cell System." *International Journal of Production Economics* 145 (2): 799–806.
- Yu, Y., S. Wang, J. Tang, I. Kaku, and W. Sun. 2016. Complexity of Line-seru Conversion for Different Scheduling Rules and Two Improved Exact Algorithms for the Multi-objective Optimization. *SpringerPlus* 5 (1): 1–26.
- Yu, Y., J. F. Tang, W. Sun, Y. Yin, and J. W. Wang. 2017. "Line-hybrid Seru System Conversion: Models, Complexities, Properties, Solutions and Insights." *Computers & Industrial Engineering* 103: 282–299.
- Zhang, X. L., C. G. Liu, W. J. Li, S. Evans, and Y. Yin. 2017. "Effects of Key Enabling Technologies for Seru Production on Sustainable Performance." *Omega* 66: 290–307.