Yu *et al. SpringerPlus* (2016) 5:809
DOI 10.1186/s40064-016-2445-5

SpringerPlus

RESEARCH Open Access

Complexity of line-*seru* conversion for different scheduling rules and two improved exact algorithms for the multi-objective optimization

Yang Yu[1], Sihan Wang[1], Jiafu Tang[2], Ikou Kaku[3] and Wei Sun[4*]

*Correspondence: iwuzetian@126.com
[4] Business School, Liaoning University, Shenyang 110316, People's Republic of China
Full list of author information is available at the end of the article

Abstract
Productivity can be greatly improved by converting the traditional assembly line to a *seru* system, especially in the business environment with short product life cycles, uncertain product types and fluctuating production volumes. Line-*seru* conversion includes two decision processes, i.e., *seru* formation and *seru* load. For simplicity, however, previous studies focus on the *seru* formation with a given scheduling rule in *seru* load. We select ten scheduling rules usually used in *seru* load to investigate the influence of different scheduling rules on the performance of line-*seru* conversion. Moreover, we clarify the complexities of line-*seru* conversion for ten different scheduling rules from the theoretical perspective. In addition, multi-objective decisions are often used in line-*seru* conversion. To obtain Pareto-optimal solutions of multi-objective line-*seru* conversion, we develop two improved exact algorithms based on reducing time complexity and space complexity respectively. Compared with the enumeration based on non-dominated sorting to solve multi-objective problem, the two improved exact algorithms saves computation time greatly. Several numerical simulation experiments are performed to show the performance improvement brought by the two proposed exact algorithms.

Keywords: Manufacturing, Scheduling rule, Time complexity, Space complexity, Exact algorithm

Background

The *seru* production, conceived at Sony, is an innovation of assembly system used widely in the Japanese electronics industry and recognized a new production patten. *Seru* is a manufacturing organization (an assembly unit) that consists of simple equipment and one or several worker(s) that are dedicated to one or several product(s). In *seru*, worker(s) must be multi-skilled operators, i.e., workers can operate the most or all processes of production.

To compete in a turbulent market, in 1992, several mini-assembly units were created in one of Sony's video-camera factories for an 8-mm CCD-TR55 video-camera, after dismantling a long assembly conveyor line. As did the original conveyor line, each mini-assembly unit produced the entire product. In 1994, Tatsuyoshi Kon, a former Sony staff, called this mini-assembly organization as *seru*, a Japanese word for cellular organism. *Seru* is similar to assembly cells, a widely adopted assembly system in western industries. Equipment, however, is less important for *seru*. As a human-centered assembly system, *seru* is an old-fashioned workshop where craftsperson, including jack-of-all-trades workers, assembles an entire product from-start-to-finish by her- or himself. This mini-assembly organization is regarded as an ideal combination of lean and agile production paradigms. By adopting *seru* production, Canon and Sony reduced 720,000 and 710,000 m^2 of floor space, respectively (Stecke et al. 2012). Cost can also be reduced largely by using *seru* systems. After adopting *seru* systems, Canon's costs were reduced significantly, by 55 billion yen in 2003, and by a total of 230 billion yen from 1998 to 2003. As a result, Canon emerged as a leading electronics maker. Its average productivity is higher than that of Toyota (Yin et al. 2008). Other benefits from *seru* systems include the reductions of throughput time, setup time, required labor hours, WIP inventories, and finished-product inventories. Some amazing cases related to the reductions in throughput time and required labor hours, the throughput time was reduced by 53 % at Sony Kohda and 35,976 required workers, equal to 25 % of Canon's previous total workforce, have been saved.

There are three types of *seru*: divisional *seru*, rotating *seru*, and *yatai*. A divisional *seru* is a short line staffed with several partially cross-trained workers. Tasks within a divisional *seru* are divided into different sections. Each section is operated by one or more workers. Workers staffed within rotating *seru* or *yatais* are completely cross-trained. A rotating *seru* is often organized in a U-shaped layout with several workers. Each worker assembles an entire product from-start-to-finish without disruption. A *yatai* is the *seru* with a single worker who does all operational and managerial tasks. An NEC (Nippon Electric Company in Japan) completely cross-trained worker can assemble a word processor of 120 components in 18 min (Shinohara 1995; Stecke et al. 2012). In this research, the *serus* are rotating *serus* or *yatai*. A detailed introduction of *seru* system and its managerial mechanism can be found in Yin et al. (2008), Liu et al. (2010) and Stecke et al. (2012).

Due to the merit of *seru* production, many companies converted assembly line into *seru* system to increase the productivity. The line-*seru* (or line-cell) conversion was used widely in the Japanese electronics industry (Isa and Tsuru 1999; Miyake 2006; Sakazume 2005, 2012; Shinobu 2009; Yoshimoto 2003). Its essence is to convert traditional conveyor assembly line to a *seru* system in which one (or multiple) worker performs the most of all tasks the *seru*. The total productivity of manufacturers may be increased dramatically by line-*seru* conversion (Johnson 2005; Kaku et al. 2009; Stecke et al. 2012; Yin et al. 2008). Liu et al. (2014) proposed an implementation framework and process for converting the assembly line into a *seru* system.

The first issue of line-*seru* conversion is to establish the mathematical model. Such technical and decision making problems had been defined as line-*seru* conversion problems (Kaku et al. 2009). Kaku et al. (2009) considered three types of systems including a pure *seru* system (as shown in Fig. 1, where two *serus* are constructed, i.e., workers 2 and 5 in *seru* 1 and workers 1, 3 and 4 in *seru* 2), a pure assembly line and a hybrid system with *serus* and line. The pure *seru* system is very simple and a special case of all other *seru* assembly systems. The results obtained for pure *seru* system models not only

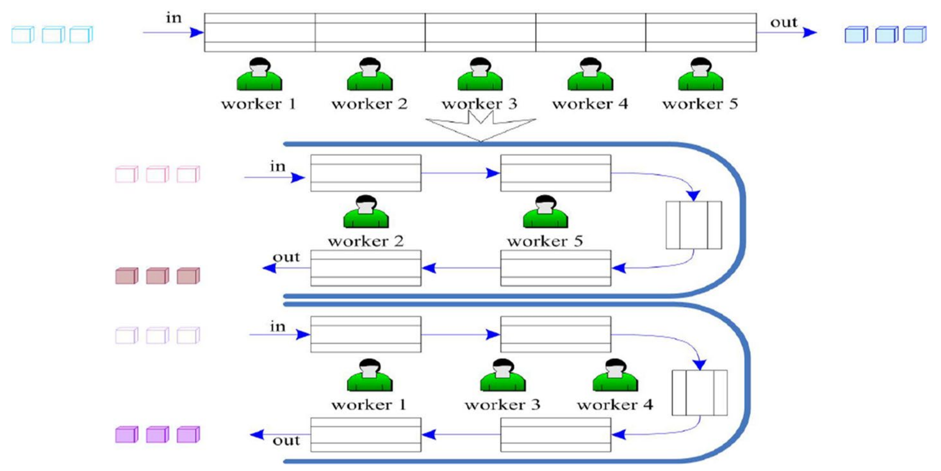

Fig. 1 An example of converting assembly line to a pure *seru* system (Yu et al. 2012)

provide insights into the pure *seru* system environment, they also provide a basis for heuristics that are applicable to more complicated assembly *seru* system environments. Therefore, many literatures focused on converting the assembly line to a pure *seru* system, such as Yu et al. (2012, 2013, 2014). Also, the research considers the assembly line is converted to a pure *seru* system.

Another key of line-*seru* conversion is to evaluate the performance improvement created by the conversion. Kaku et al. (2009) used total throughput time (TTPT) and total labor hours (TLH) to evaluate the performance improvement created by line-*seru* conversion. Kaku et al. (2009) and Yu et al. (2012) investigated the operational influence factors to TTPT and TLH. They summarized several managerial insights that could be used to improve the performances of TTPT and TLH through line-*seru* conversion. Yu et al. (2013) evaluated the performance improvement from the perspective of manpower reduction. They established the line-*seru* conversion model towards reducing worker(s) and proposed an exact algorithm.

Also, the decision problem in line-*seru* conversion is widespread concerned. In fact, line-*seru* conversion includes two decision problems, i.e., *seru* formation and *seru* load (Yu et al. 2012, 2013, 2014). Most of previous researches focused on *seru* formation. Yu et al. (2012) investigated how to format *serus* to improve the performances of TTPT and TLH. Yu et al. (2013) clarified the complexity of *seru* formation towards reducing workers. Yu et al. (2014) revealed the mathematical characteristics of *seru* formation such as solution space, complexity and non-convex properties. Regarding *seru* load, most researches used given scheduling rule to assign product batches to *serus*, because *seru* load is NP-hard. Yu et al. (2012, 2013, 2014) used the FCFS (First Come First Severed) rule to dispatch product batches into *serus*. Therefore, the *seru* load should be investigated.

This paper, originally motivated by line-*seru* conversion applications of Sony and Canon, has two purposes. First, we demonstrate the influence of ten different scheduling rules usually used in *seru* load on the performance of line-*seru* conversion. Subsequently, we clarify exactly the complexities of *seru* load and line-*seru* conversion for the ten scheduling rules. Second, to obtain Pareto-optimal solutions for the large-scale instances of line-*seru* conversion, we propose two improved exact algorithms by decreasing time complexity and space complexity respectively.

The remainder of this study is organized as follows. The bi-objective model of converting the assembly line into a pure *seru* system with minimizing TTPT and TLH is given in the second section. The third section illustrates the influence of ten scheduling rules on the TTPT and TLH performances of *seru* system. The forth section clarifies exactly the complexities of *seru* load and line-*seru* conversion for ten scheduling rules. In the fifth section, two exact algorithms are developed based on reducing time complexity and space complexity. Several examples to illustrate the performance of the two proposed algorithms are given in the sixth section. In the last section conclusions and future research are given. All theorem proofs can be found in the "Appendix".

Multi-objective model of converting the assembly line into a pure *seru* system

Assumption

The following assumptions are considered in this study.

1. The types and batches of products to be processed are known in advance. There are N product types that are divided into M product batches. Each batch contains a single product type.
2. The assembly tasks within a *seru* are manual so need only simple and cheap equipment and the cost of duplicating equipment is ignored (Stecke et al. 2012; Yu et al. 2012).
3. A product batch is assembled entirely within a *seru*.
4. All product types have the same assembly tasks. If a task is not used in a product, then we assume the task time for the product was zero.
5. In the assembly line, each task (or station) is in the charge of a single worker. That means that a worker only performs a single assembly task in the assembly line. Therefore, the total number of tasks in the line equals W.
6. The assembly tasks within each *seru* are the same as the ones within the line. A *seru* worker needs to perform all assembly tasks, assembles an entire product from-start-to-finish, and there is no disruption or delay between adjacent tasks.

Indices

i	Index of workers ($i = 1,2,\ldots,w$). w is the total number of workers in an assembly line.
j	Index of *serus* ($j = 1,2,\ldots,J$). J is the total number of *serus* in a *seru* system.
n	Index of product types ($n = 1, 2,\ldots, N$). N is the total number of product types.
m	Index of product batches ($m = 1, 2,\ldots, M$). M is the total number of product batches.
k	Index of the sequence of product batches in a *seru* ($k = 1, 2,\ldots, M$).
q	Index of sub-sets of all the feasible *seru* systems ($q = 1, 2,\ldots, Q$). Q is the total number of sub-sets.

Parameters

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

B_m	Size of product batch m.
T_n	Cycle time of product type n in the assembly line.
SL_n	setup time of product type n in the assembly line.
SCP_n	Setup time of product type n in a *seru*.
T_{mj}	Average task time of *seru* j assembling 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 within a *seru* will be longer than her or his task time within the original assembly line.
ε_i	Worker *i's* coefficient of influencing level of doing multiple assembly tasks.
β_{ni}	Skill level of worker i for each task of product type n.

Decision variables

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

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

Variables

C_i	Coefficient of variation of worker *i's* increased task time after line-*seru* conversion, i.e., from a specialist to a completely cross-trained worker. if the number of worker *i's* tasks within a *seru* is over her or his upper bound η_i, i.e., $w > \eta_i$, then the worker will cost more average task time than her or his task time within the original assembly line. c_i is given in Eq. (1).
TC_m	Assembly task time of product batch m per station in a *seru*. In a *seru*, the task time of product type n is calculated by the average task time of workers in the *seru*. TC_m is represented as Eq. (2).
FCB_m	Begin time of product batch m in a *seru*. There is no waiting time between two product batches so that FCB_m is the aggregation of flow time and setup time of the product batches processed prior to product batch m in the same *seru*. FCB_m is represented as Eq. (3).
SC_m	Setup time of product batch m in a *seru*. Setup time is considered when two different types of products are processed consecutively; otherwise the setup time is zero. For example, in Eq. (4), two adjacent assembled products in a *seru* are expressed as m and m'. If the product type of m is different with that of m', i.e., $V_{mn} = 1, V_{m'n} = 0$, and then the setup time of batch m is $SCP_n V_{mn}$. However, if the product types of m and m' are identical, i.e., $V_{mn} = V_{m'n} = 1$, and then the setup time of batch m is 0.
FC_m	Flow time of product batch m in a *seru*. FC_m is represented as Eq. (5).

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

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

$$FCB_m = \sum_{s=1}^{m-1}\sum_{j=1}^{J}\sum_{k=1}^{m}(FC_s + SC_s)Z_{mjk}Z_{sj(k-1)} \tag{3}$$

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

$$FC_m = \frac{B_mTC_mW}{\sum_{i=1}^{W}\sum_{j=1}^{J}\sum_{k=1}^{M} X_{ij}Z_{mjk}} \tag{5}$$

TTPT and TLH of the *seru* system

The total throughput time (TTPT) and the total labor hours (TLH) of the *seru* system are expressed as follows.

$$\text{TTPT of } seru \text{ system} = \operatorname*{Max}_m(FCB_m + FC_m + SC_m) \tag{6}$$

$$\text{TLH of } seru \text{ system} = \sum_{m=1}^{M}\sum_{i=1}^{W}\left(\sum_{j=1}^{J}\sum_{k=1}^{M} FC_mX_{ij}Z_{mjk}\right) \tag{7}$$

TTPT of *seru* system is the completion time of the last completed product batch. TLH of *seru* system is the cumulative working time of all workers in the *seru* system. Given product batches, the *seru* systems should have shorter TTPT and TLH than the line.

Formulation of bi-objective line-*seru* conversion with minimizing TTPT and TLH

The mathematical model is formulated as Eqs. (8)–(13).

Objective functions:

$$\text{Min TTPT of } seru \text{ system} \tag{8}$$

$$\text{Min TLH of } seru \text{ system} \tag{9}$$

Subject to:

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

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

$$\sum_{j=1}^{J}\sum_{k=1}^{M} Z_{mjk} = 1, \quad \forall m \tag{12}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{M} Z_{mjk} = 0, \quad \left(\forall j \middle| \sum_{i=1}^{W} X_{ij} = 0\right) \tag{13}$$

where Eq. (8) minimizes the total throughput time (TTPT). Equation (9) minimizes the total labor hours (TLH). Equation (10) is the number constraint that the number of workers within a *seru* must be in the interval of [1, *W*]. Equation (11) is the worker assignment rule, i.e., each worker should be assigned to one and only one *seru*. Equation (12) is the product batch assignment rule, i.e., each batch should be assigned to one and only one *seru*. Equation (13) is the rule of assigning constraint, i.e., a product must be assigned to a *seru* in which at least one worker is assigned. In other words, for a *seru* without any worker, i.e., $\forall j| \sum_{i=1}^{W} X_{ij} = 0$, any batch cannot be assigned into the *seru*, i.e., $\sum_{m=1}^{M} \sum_{k=1}^{M} Z_{mjk} = 0$.

Influence of the scheduling rules on line-*seru* conversion

Line-*seru* conversion includes *seru* formation and *seru* load (Yu et al. 2012). *Seru* load determines which product batches are dispatched to the *serus* formed in *seru* formation. *Seru* formation determines how many *serus* to be constructed and how to assign the workers into *serus*. A detailed introduction of *seru* formation can be found in Yu et al. (2012, 2013, 2014).

Property 1 *Given a seru formation, without a given scheduling rule seru load is NP-hard.*

Explanation Without a given scheduling rule, in *seru* load, each product batch can be assigned into any *seru* in the given *seru* formation. Therefore, *seru* load is an assignment and NP-hard problem.

Thus, for simplicity, earlier researches (Kaku et al. 2009; Yu et al. 2012, 2013, 2014) fixed the scheduling rules on *seru* load such as FCFS and SPT.

However, the different scheduling rules produce different performances or complexity of a system (Chuen and Robert 1993; Grabot and Geneste 1994; Holthaus and Rajendran 1997; Amirghasemi and Zamani 2014; Nurre and Sharkey 2014; Xu et al. 2015; Zeng et al. 2015). The comparative analysis of scheduling rules in some specific industrial environments can be found in Rajendran and Holthaus (1999), Kizil et al. (2006), Chiang and Fu (2007), Akturk (2011), and Li et al. (2015).

To investigate the influence of scheduling rules on the performance of line-*seru* conversion, we used a total of ten scheduling rules of *seru* load. The ten scheduling rules are selected from *Seru* production applications of Sony and Canon, because the ten rules are usually used in *Seru* production. The ten scheduling rules used in the paper are defined in detailed as follows.

To illustrate clearly that TTPT and TLH of line-*seru* conversion are influenced by the different scheduling rules, we used 5 product batches and 2 *serus*, where worker 1 in *seru* 1 and worker 2 in *seru* 2. The data of task time of 5 batches on 2 *seru* is shown in Table 1. The data of the earliest due date (*EDD*) of 5 batches is shown in Table 2.

First come first served (FCFS) This rule is often used as a bench-mark. FCFS of *seru* load is described as following: an arriving product batch is assigned to the empty *seru* with the smallest *seru* number. If all *serus* are occupied, the product batch is assigned to the *seru* with the earliest completion time. Result with FCFS on *seru* load shows in Fig. 2, where TTPT is 10 and TLH is 19.

Last come first served (LCFS) LCFS of *seru* load is described as following: the last arriving product batch is assigned to the empty *seru* with the smallest *seru* number. If all *serus* are occupied, the product batch is assigned to the *seru* with the earliest completion time. Result with LCFS on *seru* load shows in Fig. 3, where TTPT is 7 and TLH is 14.

Shortest processing time (SPT) This rule is perhaps the most commonly used rule for job shop scheduling. SPT of *seru* load is described as following: an arriving product batch is assigned to the *seru* with the shortest processing time for it. The shortest processing time (SPT) of batch m is $\min_{j=1}^{J}(Tmj)$, e.g., SPTs of batches 1 and 2 are 1 and 4 respectively. Result with SPT on *seru* load shows in Fig. 4, where TTPT is 10 and TLH is 13.

Table 1 Task time of five batches on 2 *serus*

***Serus*\batches**	**1**	**2**	**3**	**4**	**5**
*Seru*1 {1}	2	4	3	4	3
*Seru*2 {2}	1	6	4	2	4

*Seru*1 {1} and *Seru*2 {2} mean worker 1 in *seru* 1 and worker 2 in *seru* 2. The *seru* sequence is {1}-{2}

Table 2 Earliest due date of 5 batches

Batches	**1**	**2**	**3**	**4**	**5**
EDD	6	9	10	5	8

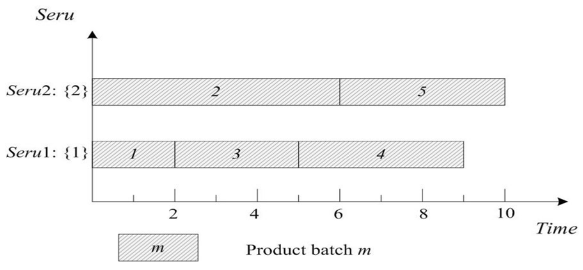

Fig. 2 Result of FCFS rule on *seru* load with the *seru* sequence of {1}–{2}

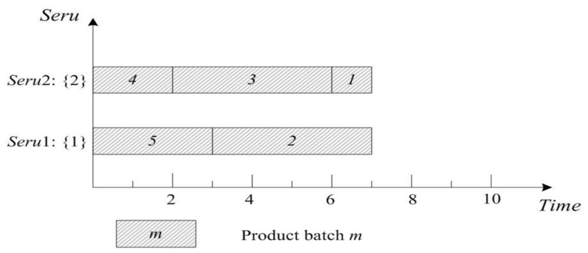

Fig. 3 Result of LCFS rule on *seru* load with the *seru* sequence of {1}–{2}

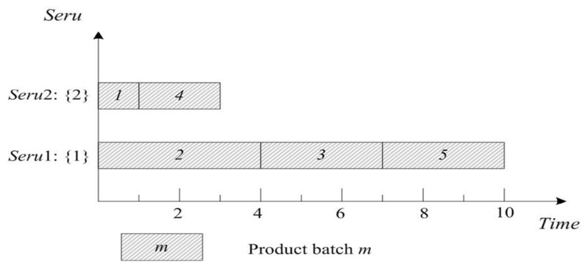

Fig. 4 Result of SPT rule on *seru* load with the *seru* sequence of {1}–{2}

Earliest completion time (ECT) As shown in Fig. 4, SPT rule on *seru* load may cause the imbalance among *serus*. For example, the difference between the two *serus* in Fig. 4 is 7 = 10-3. Therefore, Yu et al. (2012) proposed that the balance among *serus* should be considered in *seru* load. ECT of *seru* load is described as following: an arriving product batch is assigned to the *seru* with the earliest completion time of finishing the batch. Result with ECT on *seru* load shows in Fig. 5, where TTPT is 7, TLH is 14, and the difference between the two *serus* is 0.

Earliest due-date first (EDD) This rule is often used in industries for its simplicity of implementation. EDD of *seru* load is described as following: the product batch with the earliest due-date is selected and assigned to the *seru* with the shortest processing time for the batch. Result with EDD on *seru* load shows in Fig. 6, where TTPT is 10 and TLH is 13.

Modified earliest due-date first (MEDD) MEDD of *seru* load is described as following: the product batch with the earliest due-date is selected and assigned to the *seru*

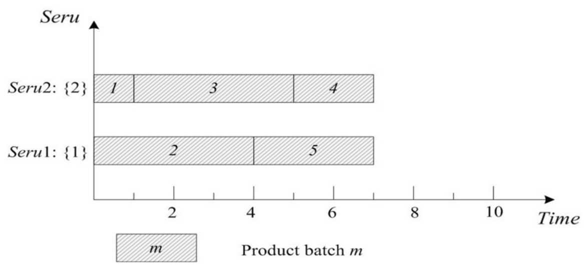

Fig. 5 Result of ECT rule on *seru* load with the *seru* sequence of {1}–{2}

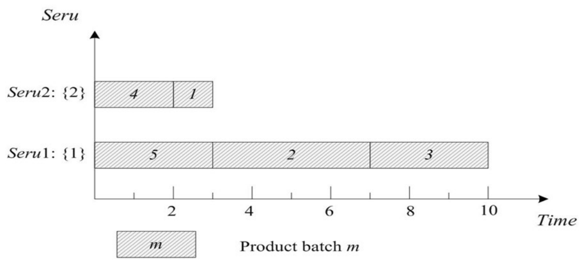

Fig. 6 Result of EDD rule on *seru* load with the *seru* sequence of {1}–{2}

with the earliest completion time of finishing the batch. Result with MEDD on *seru* load shows in Fig. 7, where TTPT is 8 and TLH is 16.

Minimal Shortest Processing Time first (MSPT) MSPT of *seru* load is described as following: the product batch with the minimal shortest processing time is selected and assigned to the *seru* with the shortest processing time for it. MSPT is the minimal SPT of all batches, i.e., $\min_{m=1}^{M} \min_{j=1}^{J} (Tmj)$. For example, MSPT of Table 1 is 1. Result with MSPT on *seru* load shows in Fig. 8, where TTPT is 10 and TLH is 13.

Modified minimal shortest processing time first (MMSPT) MMSPT of *seru* load is described as following: the product batch with the minimal shortest processing time is selected and assigned to the *seru* with the earliest completion time of finishing the batch. Result with MMSPT on *seru* load shows in Fig. 9, where TTPT is 9 and TLH is 15.

Longest shortest processing time first (LSPT) LSPT of *seru* load is described as following: the product batch with the longest shortest processing time is selected and assigned to the *seru* with the shortest processing time for it. LSPT is the longest SPT

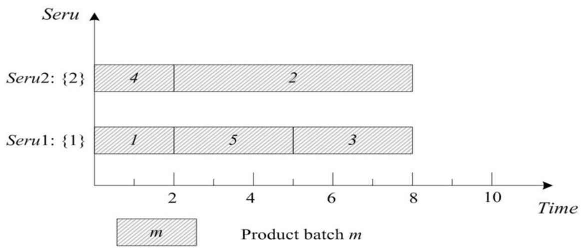

Fig. 7 Result of MEDD rule on *seru* load with the *seru* sequence of {1}–{2}

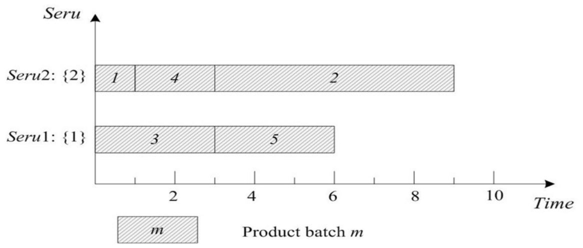

Fig. 8 Result of MSPT rule on *seru* load with the *seru* sequence of {1}–{2}

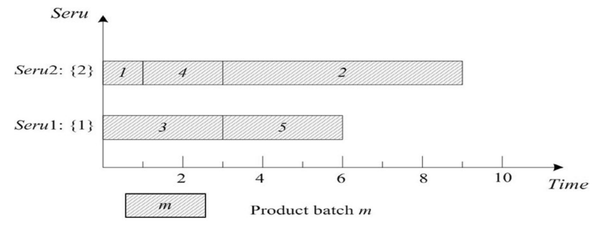

Fig. 9 Result of MMSPT rule on *seru* load with the *seru* sequence of {1}–{2}

of all batches, i.e., $\max_{m=1}^{M}\min_{j=1}^{J}(Tmj)$. For example, LSPT of Table 1 is 4. Result with LSPT on *seru* load shows in Fig. 10, where TTPT is 10 and TLH is 13.

Modified longest shortest processing time first (MLSPT) MLSPT of *seru* load is described as following: the product batch with the longest shortest processing time is selected and assigned to the *seru* with the earliest completion time of finishing the batch. Result with MLSPT on *seru* load shows in Fig. 11, where TTPT is 7 and TLH is 14.

Based on Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11, we can obtain Table 3. Table 3 shows that the different scheduling rules on *seru* load cause different TTPTs and TLHs of the converted *seru* system even though the *seru* formation is identical.

From Table 3, we can see that scheduling rules have a significant effect on the performance of converted *seru* system with the same *seru* formation. For example, the best and worst *TTPT* in the ten scheduling rules are 7 and 10 respectively, and the best and worst *TLH* in the ten scheduling rules are 13 and 19. Therefore, the investigation on scheduling rules used in *seru* load is important for *Seru* production. Consequently, we clarify the complexities of solution spaces of *seru* load and line-*seru* conversion for the ten scheduling rules from the theoretical perspective. The clarification of complexity of solution

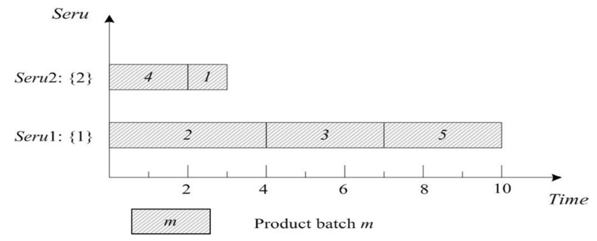

Fig. 10 Result of LSPT rule on *seru* load with the *seru* sequence of {1}–{2}

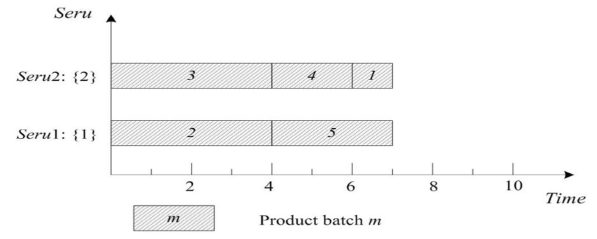

Fig. 11 Result of MLSPT rule on *seru* load with the *seru* sequence of {1}–{2}

Table 3 Result of 10 scheduling rules used in *seru* load with the same *seru* formation

Scheduling rules	FCFS	LCFS	SPT	ECT	EDD	MEDD	MSPT	MMSPT	LSPT	MLSPT
TTPT	10	7	10	7	10	8	9	9	10	7
TLH	19	14	13	14	13	16	15	15	13	14

space makes it possible to obtain the optimal solution or Pareto-optimal solutions of line-*seru* conversion.

Complexity of *seru* load and line-*seru* conversion for the different scheduling rules

The line-*seru* conversion is a two-stage decision process, i.e., *seru* formation and *seru* load (Yu et al. 2012, 2014). Therefore, the complexity of line-*seru* conversion should be clarified by combining the complexity of *seru* formation with the complexity of *seru* load.

Complexity of *seru* formation

Seru formation is the first step of line-*seru* conversion. Distinguished from the traditional manufacturing cell formation problems (Safaei and Tavakkoli-Moghaddam 2009; Wu et al. 2009), *seru* formation in line-*seru* conversion is to determine how many *serus* to be formed and how to assign workers into the *serus* (Yu et al. 2012). *Seru* formation is decided by decision variable X_{ij}.

Property 2 *Seru formation of line-seru conversion is an instance of the unordered set partition problem and NP-hard.*

Explanation *Seru* formation is to partition a conveyor line with W workers into pairwise disjoint nonempty *serus*, and so *seru* formation is an instance of the unordered set partition problem. Set partitioning is a well-known NP-hard problem (Garey and Johnson 1979). The detailed proof can be found in Yu et al. (2013).

Since *seru* formation is an instance of the unordered set partition, the number of all the feasible solutions of *seru* formation can be expressed as:

$$F(W) = \sum_{J=1}^{W} P(W, J) \tag{14}$$

where $P(W, J)$ is the count of partitioning W workers in assembly line into J *serus* and can be expressed as the Stirling numbers of the second kind (Rennie and Dobson 1969; Williamson 1985; Knopfmacher and Mays 2006; Klazar 2003).

Complexities of *seru* load for the different scheduling rules

Seru load is the second step of line-*seru* conversion and is decided by decision variable Z_{mjk}. It determines which product batches are dispatched to the *serus* formed in *seru* formation (Chen et al. 2013; Solimanpur and Elmi 2013).

According to Properties 1 and 2, line-*seru* conversion is a complex problem including two NP-hard problems (i.e., *seru* formation and *seru* load). For simplicity and without loss of generality, the scheduling rule in *seru* load is usually given. However, even given a scheduling rule in *seru* load, line-*seru* conversion is still an NP-hard problem because *seru* formation is NP-hard.

Different scheduling rules produce different performances or complexity of line-*seru* conversion. Up to now, the influences of scheduling rules to line-*seru* conversion are not investigated yet. Therefore, one objective of this study is to clarify the influence of different scheduling rules to complexity of line-*seru* conversion. Since the complexity of *seru* formation is independent of scheduling rule, we focus on clarifying the influence of different scheduling rules to complexity of *seru* load.

Gven a *seru* formation, the number of solutions (S) of *seru* load can be expressed by the number of *serus* (J).

Theorem 1 *Given a seru formation with J serus, without given a scheduling rule,* $S = J^M$.

Proof See Proof of Theorem 1 in "Appendix".

As described in Property 1 and Theorem 1, *seru* load is NP-hard and has J^M feasible solutions. For simplicity, therefore, earlier researches used the typical scheduling rules such as FCFS and SPT.

In addition, the number of solutions (S) of *seru* load varies with the scheduling rules. For example, for the line with two workers labeled 1 and 2, there are two solutions of *seru* formation, i.e., {{1,2}} and {{1},{2}}. For the latter solution of {{1},{2}}, there are two *seru* sequences, i.e., {1}–{2} and {2}–{1}.

For the 5 product batches in Table 1, there are two results of *seru* load with FCFS. The result of {1}–{2} is shown in Fig. 2 and the result of {2}–{1} is shown in Fig. 12. This means the *seru* sequence does influence the result of *seru* load with FCFS.

However, for the 5 product batches in Table 1, there is only one result of *seru* load with SPT. For the *seru* sequence of {1}–{2}, the result of *seru* load with SPT is shown in Fig. 4. For the *seru* sequence of {2}–{1}, the result of *seru* load with SPT is shown in Fig. 13. By comparing Figs. 4 and 13, we can easily observe that the two results are identical. This is because regardless of the *seru* sequence, using SPT an arriving batch is always assigned to the *seru* with the shortest processing time for it. This means the *seru* sequence does not influence the result of *seru* load with SPT.

Therefore, the ten scheduling rules are divided into two classes: (1) scheduling rules related to *seru* sequence (*SRRSS*); and (2) scheduling rules unrelated to *seru* sequence (*SRUSS*). A *SRRSS* rule is the one with which the *seru* load result is influenced by the

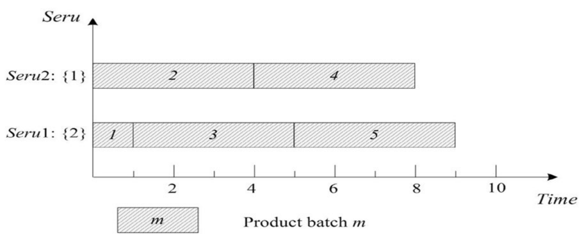

Fig. 12 Result with FCFS rule on *seru* load with the *seru* sequence of {2}–{1}

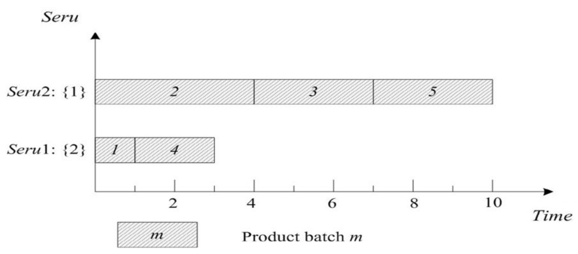

Fig. 13 Result with SPT rule on *seru* load with the *seru* sequence of {2}–{1}

seru sequence. However, a *SRUSS* rule means that the *seru* load result is independent of the *seru* sequence using the rule. In the ten scheduling rules, FCFS and LCFS belong to *SRRSS*, but the other eight scheduling rules belong to *SRUSS*. Thus, we clarify the complexity of *seru* load with the ten scheduling rules from the two classes.

Complexity (S) of *seru* load with *SRRSS* is clarified in Theorems 2–3. Complexity (S) of *seru* load with *SRUSS* is clarified in Theorem 4.

Theorem 2 *Given a seru formation with J serus, if seru load uses a SRRSS and* $M \geq J$, $S = J! = P_J^J$.

Proof See Proof of Theorem 2 in "Appendix".

Theorem 3 *Given a seru formation with J serus, if seru load uses a SRRSS and* $M < J$, $S = C_J^M P_M^M = P_J^M$.

Proof See Proof of Theorem 3 in "Appendix".

Theorem 4 *Given a seru formation with J serus, if seru load uses a SRUSS,* $S = 1$.

Proof See Proof of Theorem 4 in "Appendix".

Subsequently, the complexity ($T(W)$) of line-*seru* conversion with the different scheduling rules can be clarified by combining the complexity of *seru* formation ($F(W)$) with the complexity of *seru* load (S).

Complexities of line-*seru* conversion with the different scheduling rules

The complexities ($T(W)$) of line-*seru* conversion with the different scheduling rules are summarized in Table 4.

The clarification of complexity of solution space makes it possible to obtain the optimal solution or Pareto-optimal solutions of line-*seru* conversion.

Two improved exact approaches for multi-objective line-*seru* conversion

Multi-objective decisions are often used in line-*seru* conversion (Kaku et al. 2009; Yu et al. 2013, 2014). However, multi-objective optimization is more difficult to solve than single-objective optimization (Ebrahimipour et al. 2015). Enumeration algorithm based on non-dominated sorting (Deb et al. 2002) for multi-objective line-*seru* conversion is described as follows.

Table 4 Complexities ($T(W)$) of line-*seru* conversion with the different scheduling rules

Scheduling on *seru* load	$T(W)$	Explanation
Without given a scheduling	$\sum_{J=1}^{W} P(W,J) * (J^M)$	Combine Eq. (14) with Theorem 1
A *SRRSS* and $M \geq J$	$\sum_{J=1}^{W} P(W,J) * (P_J^J)$	Combine Eq. (14) with Theorem 2
A *SRRSS* and $M < J$	$\sum_{J=1}^{W} P(W,J) * (P_J^M)$	Combine Eq. (14) with Theorem 3
A *SRUSS*	$\sum_{J=1}^{W} P(W,J)$	Combine Eq. (14) with Theorem 4

Algorithm. Enumeration based on non-dominated sorting.

```
Input: $W$ (number of workers in the line), a given scheduling rule.
Output: $P\_OPTs$ (Pareto-optimal solutions).
(1) Produce the set ($S$) of all the feasible solutions with the given scheduling rule.
(2) for (each $S_i \in S$) do
        calculate TTPT of $S_i$
        calculate TLH of $S_i$
    end for
(3) Produce Pareto-optimal solutions ($P\_OPTs$) by non-dominated sorting the set $S$.
(4) Output $P\_OPTs$.
```

Step (1) is to produce all the feasible solutions (N) with the given scheduling rule. Both time complexity and space complexity are $O(N)$.

Step (2) is to calculate TTPT and TLH of each feasible solution. Time complexity is $O(N)$, but space complexity is $O(4N)$ for storing *seru* sequence, TTPT, TLH and the batches assigned in each *seru*.

Step (3) is to obtain exact Pareto-optimal solutions by non-dominated sorting of Deb et al. (2002). Time complexity of non-dominated sorting is $O(2N^2)$, where 2 is the number of objectives. Space complexity is $O(N)$.

Therefore, time complexity and space complexity of the enumeration algorithm are $O(2N^2)$ and $O(4N)$ respectively.

However, because of the higher time complexity (i.e., $O(2N^2)$), the enumeration cannot obtain the Pareto-optimal solutions for the instances with more than 6 workers using a *SRRSS*. We develop two improved exact algorithms for the large-scale instances by decreasing time complexity and space complexity respectively.

The improved exact algorithm by decreasing time complexity

When the solutions attending non-dominated sorting are reduced by R, the time complexity will be improved by $\left(1 - \frac{(N-R)^2}{(N)^2} * 100\ \%\right)$.

Therefore, we consider to cut off the solutions dominated by the certain Pareto-optimal solution(s) before running non-dominated sorting algorithm. The certain solutions are defined in Definitions 1 and 2.

Definition 1 mTTPT is the Pareto-optimal solution with the minimal TTPT.

Definition 2 mTLH is the Pareto-optimal solution with the minimal TLH.

By cutting off the solutions dominated by $mTTPT$ or $mTLH$ before non-dominated sorting, time complexity can be decreased greatly. The methods to find out the solutions dominated by $mTTPT$ and $mTLH$ are described in Theorems 5 and 6 respectively.

Theorem 5 *If a solution's TLH is more than mTTPT's, then the solution must be dominated by mTTPT.*

Proof See Proof of Theorem 5 in "Appendix".

Theorem 6 *If a solution's TTPT is more than mTLH's, then the solution must be dominated by mTLH.*

Proof See Proof of Theorem 6 in "Appendix".

The improved exact algorithm by decreasing time complexity is described as follows.

Algorithm. Improved exact algorithm by decreasing time complexity.

```
Input: $W$ (number of workers in the line), a given scheduling rule.
Output: $P\_OPTs$ (Pareto-optimal solutions).
(1) Produce the set ($S$) of all the feasible solutions with the given scheduling rule.
(2) for (each $S_i \in S$) do
        calculate TTPT of $S_i$
        calculate TLH of $S_i$
    end for
(3) $mTTPT$.TTPT←∞ (infinity)
    $mTLH$.TLH←∞ (infinity)
    for (each $S_i \in S$) do
        if ($S_i$.TTPT<$mTTPT$.TTPT) then
            $mTTPT$←$S_i$
        end if
        if ($S_i$.TTPT=$mTTPT$.TTPT) then
            if ($S_i$.TLH<$mTTPT$.TLH) then
                $mTTPT$←$S_i$
            end if
        end if
        if ($S_i$.TLH <$mTLH$.TLH) then
            $mTLH$←$S_i$
        end if
        if ($S_i$.TLH =$mTLH$.TLH) then
            if ($S_i$.TTPT<$mTLH$.TTPT) then
                $mTLH$←$S_i$
            end if
        end if
    end for
(4) for (each $S_i \in S$) do
        if (($S_i$.TLH≤$mTTPT$.TLH) || ($S_i$.TTPT≤$mTLH$.TTPT)) then
            $S$←$S$/{$S_i$}
        end if
    end for
(5) Produce Pareto-optimal solutions ($P\_OPTs$) by non-dominated sorting the set $S$.
(6) Output $P\_OPTs$.
```

In step (1), both time complexity and space complexity are $O(N)$.

In step (2), time complexity is $O(N)$ and space complexity is $O(4N)$.

Step (3) is to obtain *mTTPT* and *mTLH* by traversal all the feasible solutions. Both time complexity and space complexity are $O(N)$.

Step (4) is to obtain the solutions non-dominated (assume the number is K) by *mTTPT* or *mTLH* by traversing all the feasible solutions. Both time complexity and space complexity are $O(N)$.

Step (5) is to obtain the exact Pareto-optimal solutions by non-dominated sorting the K solutions obtained in step (4). The time complexity is $O(2K^2)$ and space complexity is $O(K)$.

The key of the improved exact algorithm by decreasing time complexity is leis in step (4), i.e., the operation of "$S \leftarrow S/\{S_i\}$". That cuts off the solutions dominated by *mTTPT* or *mTLH* before non-dominated sorting, i.e., step (5).

Obviously, for the improved exact algorithm by decreasing time complexity, the time complexity is the maximum between $O(2K^2)$ and $O(N)$. Space complexity is $O(4N)$ still. Therefore, we propose another improved exact algorithm by decreasing space complexity.

The improved exact algorithm by decreasing space complexity

If we partition all the feasible solutions (N) in several sub-sets to obtain the non-dominated solutions of each sub-set, then the space complexity will be decreased. Subsequently, the Pareto-optimal solutions can be obtained by sorting the non-dominated solutions in all sub-sets. The improved exact algorithm by decreasing space complexity is described as follows.

```
Algorithm. Improved exact algorithm by decreasing space complexity.
 Input: $W$ (number of workers in the line), a given scheduling rule.
 Output: $P\_OPTs$ (Pareto-optimal solutions).
 (1) Initialize.
     $finalAttendSortSolutions \leftarrow \varnothing$
 (2) Produce the set ($S$) of all the feasible solutions with the given scheduling rule and partition the solutions into $Q$ sub-sets.
 (3) for (each sub-set $q$) do
       for (each $S_i \in q$) do
           calculate TTPT of $S_i$
           calculate TLH of $S_i$
       end for
     end for
 (4) for (each sub-set $q$) do
       obtain non-dominated solutions ($nds_q$) of sub-set $q$ by non-dominated sorting all solutions in sub-set $q$.
       $finalAttendSortSolutions \leftarrow finalAttendSortSolutions \cup nds_q$
     end for
 (5) Produce Pareto-optimal solutions ($P\_OPTs$) by non-dominated sorting the set of $finalAttendSortSolutions$.
 (6) Output $P\_OPTs$.
```

Step (1) is to initialize.

Step (2) is to partition all the produced feasible solutions into Q sub-sets. In each sub-set, there are approximately $\frac{N}{Q}$ solutions.

Step (3) calculates the TTPT and TLH of each solution in each sub-set. The time complexity is $O\left(\frac{N}{Q}\right)$ and space complexity is $O\left(4\frac{N}{Q}\right)$.

Step (4) obtains non-dominated solutions (assume the number is S_q) of sub-set q using non-dominated sorting. The time complexity is $O\left(2\left(\frac{N}{Q}\right)^2\right)$ and space complexity is $O\left(\frac{N}{Q}\right)$.

Step (5) obtains the exact Pareto-optimal solutions by non-dominated sorting the $\sum_{q=1}^{Q} S_q$ non-dominated solutions, where $\sum_{q=1}^{Q} S_q$ non-dominated solutions refer to all sub-sets' non-dominated solutions obtained in step (4). Time complexity is $O\left(2\left(\sum_{q=1}^{Q} S_q\right)^2\right)$ and space complexity is $O\left(\sum_{q=1}^{Q} S_q\right)$.

The key of the improved exact algorithm by decreasing space complexity lies in steps (2) and (4). Step (2) partitions the whole solution space into several sub-spaces. Step (4) produces the solutions to attend the final non-dominated sorting by aggregating the non-dominated solutions of all sub-spaces.

For the improved exact algorithm by decreasing space complexity, space complexity is the maximum between $0\left(4\frac{N}{Q}\right)$ and $O\left(\sum_{q=1}^{Q} S_q\right)$, and time complexity is the maximum between $O\left(2\left(\frac{N}{Q}\right)^2\right)$ and $O\left(2\left(\sum_{q=1}^{Q} S_q\right)^2\right)$.

Computation experiments

Test instances

Tables 5, 6, 7, 8 and 9 show the parameters, data distribution and detail data of level of skill of workers, coefficient of influencing level of skill to multiple stations for workers and data of batches used in test, respectively. From the 5 Tables, it can be observed that the lot size of each batch is $N(50,5)$ and the ability of workers is also different with stations and $N(0.2,0.05)$. The detailed data of ε_i and batches are given in Tables 7 and 8 respectively.

Table 5 shows that the mean of skill level of each worker for product type n (β_{ni}) ranges from 1 to 1.2 and the standard deviations are fixed to 0.1. The detailed data of β_{ni} are given in Table 6.

For the instance with W workers, we use the following data set from Tables 5, 6, 7, 8, 9: the entire Table 5, the first W rows of Tables 7 and 8, and the entire Table 9.

Hardware and software specifications

The two improved exact algorithms were coded in C# and executed on an Intel Core(TM) i7-4790 CPU @ 3.6 GHz under Windows 7 using 8 GB of RAM.

Computation results of the improved exact algorithm by decreasing time complexity

The enumeration based on non-dominated sorting cannot solve the instances with more than 6 workers using a *SRRSS*, such as FCFS. We use the improved exact algorithm by decreasing time complexity to solve the instances with 5, 6 and 7 workers. The computation results are shown in Figs. 14, 15, 16 respectively.

Figure 14 shows 7 Pareto-optimal solutions, 541 feasible solutions, 493 solutions dominated by *mTTPT* or *mTLH*, and left 48 solutions to attend non-dominated sorting for the instance with 5 workers. Similarly, Fig. 15 shows 9 Pareto-optimal solutions, 4683 feasible solutions, 3831 solutions dominated by *mTTPT* or *mTLH*, and only left 852 solutions to attend non-dominated sorting for the instance with 6 workers. Figure 16

Table 5 Parameters in the experiments

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

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

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

Product type				
1	**2**	**3**	**4**	**5**
N(1,0.1)	*N*(1.05,0.1)	*N*(1.1,0.1)	*N*(1.15,0.1)	*N*(1.2,0.1)

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

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

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

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

shows the Pareto-optimal solutions of the instance with 7 workers, where 10 Pareto-optimal solution and 2437 solutions to attend non-dominated sorting.

Table 10 shows the performance of the improved exact algorithm by decreasing time complexity for different instances.

From Table 10, we can see that the improved exact algorithm by decreasing time complexity has a better performance than the enumeration based on non-dominated sorting because of cutting off approximately 89 % non Pareto-optimal solutions before running non-dominated sorting. Compared with the original non-dominated sorting algorithm, the step (5) saves approximately 98 % computational time. For example, for the instances with 5 and 6 workers, in step (5) the saved computational time are $\left(1 - \frac{0.002}{0.06}\right) * 100\ \% = 97\ \%$ and $\left(1 - \frac{0.03}{1.4}\right) * 100\ \% = 98\ \%$ respectively. That is because, by cutting off non-dominated solutions, the time complexities of non-dominated sorting are improved by $\left(\frac{48}{541}\right)^2 * 100\ \%$ for the instance with 5 workers and by $97\ \% = \left(1 - \left(\frac{852}{4{,}683}\right)^2\right) * 100\ \%$ for the instance with 6 workers.

Table 9 Data of batches

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

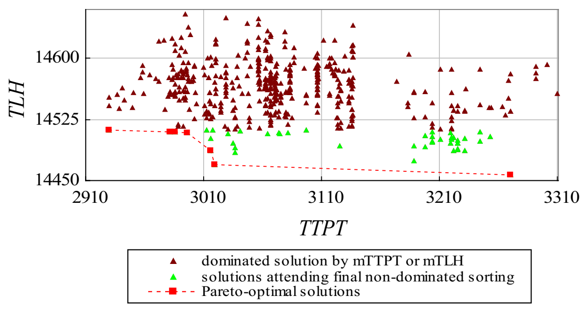

Fig. 14 The solutions dominated by *mTTPT* and *mTLH* for the instance with 5 workers

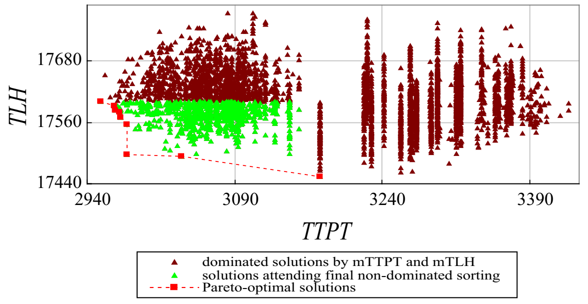

Fig. 15 The solutions dominated by *mTTPT* and *mTLH* for the instance with 6 workers

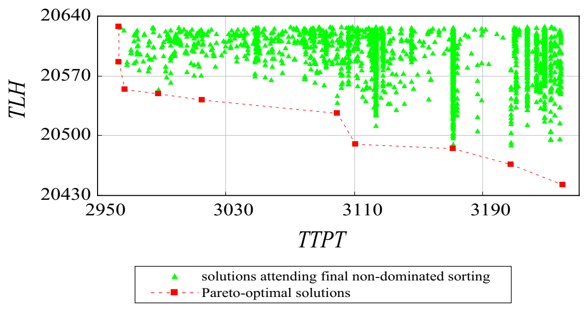

Fig. 16 The solutions to attend non-dominated sorting for the instance with 7 workers

The enumeration based on non-dominated sorting cannot solve the instances with more than 6 workers. The improved exact algorithm by decreasing time complexity solves the instance with 7 workers in 2.87 s. The time complexities of non-dominated sorting are improved by $99.7\ \% = \left(1 - \left(\frac{2{,}437}{47{,}293}\right)^2\right) * 100\ \%$.

Moreover, we can easily observe the computation time of enumeration based on non-dominated sorting increases exponentially with the number of all solutions, however, the total time of the improved exact algorithm by decreasing time complexity increases linearly with the number of all solutions.

The improved exact algorithm by decreasing time complexity cannot solve the instance with more than 7 workers because 9749 left solutions cannot be solved by non-dominated sorting.

Computation results of the improved exact algorithm by decreasing space complexity

We use the improved exact algorithm by decreasing space complexity to solve the instances with 8 and 9 workers using FCFS rule. The numbers of sub-sets (Q) of instances with 8 and 9 workers are set as 8 and 9 respectively. The computation results are shown in Figs. 17 and 18 respectively.

Figure 17 shows that there are final 599 solutions to attend non-dominated sorting and 8 Pareto-optimal solutions for the instance with 8 workers. Similarly, Fig. 18 shows there are 19 Pareto-optimal solutions generated by non-dominated sorting final 1142 solutions in all 7,087,261 feasible solutions for the instance with 9 workers.

Table 11 shows the performance of the improved exact algorithm by decreasing space complexity for different instances.

Table 10 Performance of the improved exact algorithm by decreasing time complexity

Number of workers (W)	5	6	7	8
Number of all solutions (N)	541	4683	47,293	545,835
Left solutions (K)	48	852	2437	9749
Solutions cut off	493	3831	44,856	536,086
Ratio of solutions cut off (%)	91	82	95	98.2
Time of enumeration based-NS (second)	0.06	1.4	–	–
Time of NS in step (5) (second)	0.002	0.03	0.4	–
Total time of the improved algorithm(second)	0.035	0.28	2.87	–

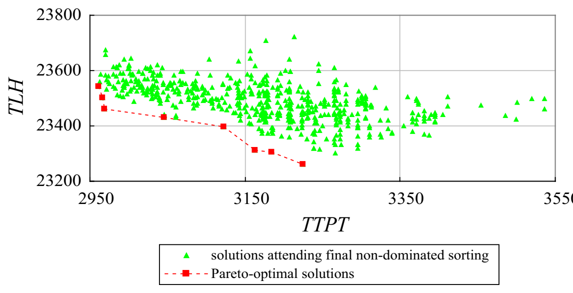

Fig. 17 The Pareto-optimal solutions of the instance with 8 workers

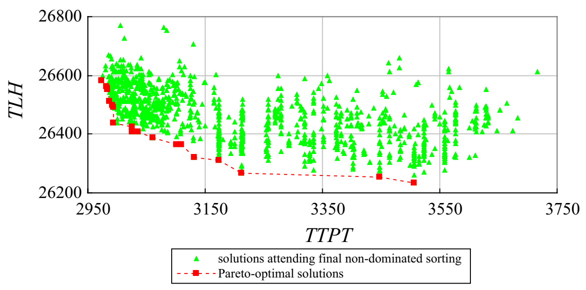

Fig. 18 The Pareto-optimal solutions of the instance with 9 workers

Table 11 Performance of the improved exact algorithm by decreasing space complexity

Number of workers (W)	5	6	7	8	9
Number of all solutions (N)	541	4683	47,293	545,835	7,087,261
Left solutions $\left(\sum_{q=1}^{Q} S_q\right)$	49	133	249	599	1142
Solutions cut off	492	4550	47,044	545,236	7,086,119
Ratio of solutions cut off (%)	91	97	99.5	99.9	99.98
Time of enumeration based-NS (second)	0.06	1.4	–	–	–
Total time of the improved algorithm (second)	0.03	0.27	2.24	25.9	342

From Table 11, we can see that the improved exact algorithm by decreasing space complexity has a better performance than the improved exact algorithm by decreasing time complexity. That is because the improved exact algorithm by decreasing space complexity cuts off more non Pareto-optimal solutions before running final non-dominated sorting (i.e., Tables 10, 11). Moreover, the total time of the improved exact algorithm by decreasing space complexity increases linearly with the number of all solutions (*N*).

However when producing all the feasible solution (*N*) is not possible, the improved exact algorithm by decreasing space complexity cannot obtain the Pareto-optimal solutions. For example of the instance with 10 workers using FCFS rule, there are 102,247,563 feasible solutions of line-*seru* conversion, and the computer cannot produce all the feasible solutions.

Conclusions

Our contributions in this paper are summarized as following. First, we investigate the significant influence of the 10 selected scheduling rules on the TTPT and TLH performances of *seru* system. Subsequently, we clarify the complexities of *seru* load and line-*seru* conversion for ten different scheduling rules in detail. Second, we develop two improved exact algorithms based on reducing time complexity and space complexity respectively, to obtain Pareto-optimal solutions of multi-objective line-*seru* conversion. Compared with the enumeration based on non-dominated sorting, the two proposed algorithms greatly decrease time complexity and space complexity respectively and improve the computation performance by approximately 98 %.

The line-*seru* conversion is a real problem in Japan electronics industry, therefore there are still a lot of works should be performed. For example, the influence of scheduling rules on the performance improvements in line-*seru* conversion should be further researched. Furthermore, other important production performances of *seru* system should be evaluated, such as balancing (Esmaeilbeigi et al. 2015) and WIP. In addition, the situations under which workers can't operate all tasks in a *seru* should be investigated, i.e., the fundamental principles of hybrid *seru* system with a short line and operation management of the *seru* system including divisional *serus*. Moreover, the further research should consider the number of assembly tasks varying with the product types. Also, the optimal methods to train the multi-skilled workers in *seru* production should be studied in future.

Authors' contributions

YY investigated the influence of scheduling rules on the performance of *seru* load and line-*seru* conversion and proposed the idea about the two exact algorithms for the bi-objective model. SH implemented the two exact algorithms and evaluated the algorithm performance by extensive experiments. JF focused on the influence of scheduling rules on the complexities of solution spaces of *seru* load and line-*seru* conversion. IK proposed the bi-objective model with minimizing TTPT and TLH for line-*seru* conversion. WS drafted the manuscript and proposed a lot of valuable suggestions. All authors read and approved the final manuscript.

Author details

[1] Institute of Systems Engineering, Northeastern University, Shenyang 110819, People's Republic of China. [2] College of Management Science and Engineering, Dongbei University of Finance and Economics, Shahekou, Dalian 116026, People's Republic of China. [3] Faculty of Environmental and Information Studies, Tokyo City University, Ushikubonishi 3-3-1, Yokohama 224-8551, Japan. [4] Business School, Liaoning University, Shenyang 110316, People's Republic of China.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (71420107028, 71571037) and Japan Society for the Promotion of Science (15K01209).

Competing interests

The authors declare that they have no competing interests.

Appendix: Proofs of Theorems

Proof of Theorem 1

Without given a scheduling rule, in *seru* load, each product batch (M) can be assigned into any *seru* (J) of the given *seru* formation.

Proof of Theorem 2

Given a *SRRSS*, if M (number of batches) $\geq J$ (number of *serus*), then the first J batches are assigned to the J *serus* according to the *seru* sequence and the *SRRSS*. That is to say a *seru* sequence of given *seru* formation produce an allocation result for the first J batches. Subsequently, the allocation result of the last $M - J$ batches can be obtained based on the result of the first J batches. This is because batch $J + 1$ will be assigned to the *seru* with the earliest completion time, batch $J + 2$ will be assigned to the *seru* with the earliest completion time, and so on. Therefore, using a *SRRSS* each *seru* sequence of the given *seru* formation produces a *seru* load result. For a *seru* formation with J *serus*, there are $J!$ *seru* sequence. Thus, if $M \geq J$, the number of solutions of *seru* load (S) equals $J! = P_J^J$.

Proof of Theorem 3

Given a *SRRSS*, if $M < J$, only the first M *serus* are used to assemble the M batches. There are C_J^M solutions of selecting arbitrary M *serus* from the J *serus*. Thus, if $M < J$, the number of solutions of *seru* load (S) equals $C_J^M P_M^M = P_J^M$.

Proof of Theorem 4

Consider $\{S_1,S_2,\ldots,S_J\}$ is the *seru* set and $\{B_1,B_2,\ldots,B_M\}$ is the batch set. If a *SRUSS* is used in *seru* load, then the *seru* sequence does not influence the scheduling results of *seru* load. Regardless of the *seru* sequence the first selected batch m (i.e., B_m) is always assigned to *seru* j (i.e., S_j) according to the *SRUSS*, the second selected batch is always assigned to the corresponding *seru*, and so on. Thus, given a *seru* formation the result of *seru* load with *SRUSS* is only.

Proof of Theorem 5

According to Definition 1, any solution's TTPT is not less than *mTTPT's*. If a solution's TLH is more than *mTTPT's*, then it must be dominated by *mTTPT*.

Proof of Theorem 6

According to Definition 2, any solution's TLH is not less than *mTLH's*. If a solution's TTPT is more than *mTLH's*, then it must be dominated by *mTLH*.

Received: 5 February 2016 Accepted: 27 May 2016
Published online: 21 June 2016

References

Akturk MS (2011) Joint cell loading and scheduling approach to cellular manufacturing systems. Int J Prod Res 49(21):6321–6341

Amirghasemi M, Zamani R (2014) A synergetic combination of small and large neighborhood schemes in developing an effective procedure for solving the job shop scheduling problem. SpringerPlus 3:193–207

Chen YY, Cheng CY, Wang LC (2013) A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems—a case study for solar cell industry. Int J Prod Econ 141(1):66–78

Chiang TC, Fu LC (2007) Using dispatching rules for job shop scheduling with due date-based objectives. Int J Prod Res 45(14):3245–3262

Chuen LC, Robert LB (1993) Complexity of single machine, multi-criteria scheduling problems. Eur J Oper Res 70(1):115–125

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Ebrahimipour V, Najjarbashi A, Sheikhalishahi M (2015) 2015, Multi-objective modeling for preventive maintenance scheduling in a multiple production line. J Intell Manuf 26(1):111–122

Esmaeilbeigi R, Naderi B, Charkhgard P (2015) The type E simple assembly line balancing problem: a mixed integer linear programming formulation. Comput Oper Res 64:168–177

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman and Company, New York

Grabot B, Geneste L (1994) Dispatching rules in scheduling: a fuzzy approach. Int J Prod Res 32(4):903–915

Holthaus O, Rajendran C (1997) Efficient dispatching rules for scheduling in a job shop. Int J Prod Econ 48(1):87–105

Isa K, Tsuru T (1999) Cell production and workplace innovation in Japan: toward a new model for Japanese manufacturing? Ind Relat 4(1):548–578

Johnson DJ (2005) Converting assembly lines to assembly cells at sheet metal products: insights on performance improvements. Int J Prod Res 43(7):1483–1509

Kaku I, Gong J, Tang J, Yin Y (2009) Modeling and numerical analysis of line-cell conversion problems. Int J Prod Res 47(8):2055–2078

Kizil M, Ozbayrak M, Papadopoulou TC (2006) Evaluation of dispatching rules for cellular manufacturing. Int J Adv Manuf Technol 28(9):985–992

Klazar M (2003) Bell numbers, their relatives, and algebraic differential equations. J Comb Theory Ser A 102(1):63–87

Knopfmacher A, Mays M (2006) Ordered and unordered factorizations of integers. Math J 10(1):72–89

Li C, Wu W, Feng Y, Rong G (2015) Scheduling FMS problems with heuristic search function and transition-timed Petri nets. J Intell Manuf 26(5):933–944

Liu CG, Lian J, Yin Y, Li W (2010) *Seru seisan*—an innovation of the production management mode in Japan. Asian J Technol Innov 18(2):89–113

Liu CG, Stecke KE, Lian J, Yin Y (2014) An implementation framework for *seru* production. Int Trans Oper Res 21(1):1–19

Miyake DI (2006) 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. Report paper of CIRJE-F-397

Nurre SG, Sharkey TC (2014) Integrated network design and scheduling problems with parallel identical machines: complexity results and dispatching rules. Networks 63(4):306–326
Rajendran C, Holthaus O (1999) A comparative study of dispatching rules in dynamic flowshops and jobshops. Eur J Oper Res 116(1):156–170
Rennie BC, Dobson AJ (1969) On stirling numbers of the second kind. J Comb Theory 7(2):116–121
Safaei N, Tavakkoli-Moghaddam R (2009) Integrated multi-period cell formation and subcontracting production planning in dynamic cellular manufacturing systems. Int J Prod Econ 120(2):301–314
Sakazume Y (2005) Is Japanese cell manufacturing a new system? A comparative study between Japanese cell manufacturing and cell manufacturing. J Jpn Ind Manag Assoc 12:89–94
Sakazume Y (2012) The organizing principles of assembly cells. Keio University Press, Tokyo **(in Japanese)**
Shinobu C (2009) The issues of *seru* production systems: from the viewpoints of autonomy and integration. St. Andrew's University. Econ Bus Rev 50(4):39–68 **(in Japanese)**
Shinohara T (1995) Shocking news of the removal of conveyor systems: single-worker *seru* production system. Nikkei Mech 24(July):20–38 **(in Japanese)**
Solimanpur M, Elmi A (2013) A tabu search approach for cell scheduling problem with makespan criterion. Int J Prod Econ 141(2):639–645
Stecke KE, Yin Y, Kaku I, Murase Y (2012) *Seru*: the organizational extension of JIT for a super-talent factory. Int J Strateg Decis Sci 3(1):105–118
Williamson SG (1985) Combinatorics for computer science. Computer Science Press, Rockville
Wu T, Chang C, Yeh J (2009) A hybrid heuristic algorithm adopting both Boltzmann function and mutation operator for manufacturing cell formation problems. Int J Prod Econ 120(2):669–688
Xu Z, Zou Y, Kong X (2015) Meta-heuristic algorithms for parallel identical machines scheduling problem with weighted late work criterion and common due date. SpringerPlus 4:782–794
Yin Y, Stecke KE, Kaku I (2008) The evolution of *seru* production systems throughout Canon. Oper Manag Educ Rev 2:27–40
Yoshimoto T (2003) Type of *seru* production: a case study of air-conditioner. Wide Bus Rev 4(2):65–75 **(in Japanese)**
Yu Y, Gong J, Tang J, Yin Y, Kaku I (2012) How to do assembly line-cell conversion? A discussion based on factor analysis of system performance improvements. Int J Prod Res 50(18):5259–5280
Yu Y, Tang J, Sun W, Yin Y, Kaku I (2013) Reducing worker(s) by converting assembly line into a pure cell system. Int J Prod Econ 145(2):799–806
Yu Y, Tang J, Gong J, Yin Y, Kaku I (2014) Mathematical analysis and solutions for multi-objective line-cell conversion problem. Eur J Oper Res 236(2):774–786
Zeng C, Tang J, Yan C (2015) Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles. J Intell Manuf 26(5):891–898