Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A three-stage method with efficient calculation for lot streaming flow-shop scheduling*

Hai-yan WANG[†‡1], Fu ZHAO[2,3], Hui-min GAO[1], John W. SUTHERLAND[2]

[1]*College of Mechanical and Electrical Engineering, Jiaxing University, Jiaxing 314000, China*

[2]*Environmental and Ecological Engineering, Purdue University, West Lafayette, IN 47907, USA*

[3]*School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA*

[†]E-mail: wanghy@mail.zjxu.edu.cn

Received July 9, 2017; Revision accepted Mar. 17, 2018; Crosschecked July 3, 2019

Abstract: An important production planning problem is how to best schedule jobs (or lots) when each job consists of a large number of identical parts. This problem is often approached by breaking each job/lot into sublots (termed lot streaming). When the total number of transfer sublots in lot streaming is large, the computational effort to calculate job completion time can be significant. However, researchers have largely neglected this computation time issue. To provide a practical method for production scheduling for this situation, we propose a method to address the *n*-job, *m*-machine, and lot streaming flow-shop scheduling problem. We consider the variable sublot sizes, setup time, and the possibility that transfer sublot sizes may be bounded because of capacity constrained transportation activities. The proposed method has three stages: initial lot splitting, job sequencing optimization with efficient calculation of the makespan/total flow time criterion, and transfer adjustment. Computational experiments are conducted to confirm the effectiveness of the three-stage method. The experiments reveal that relative to results reported on lot streaming problems for five standard datasets, the proposed method saves substantial computation time and provides better solutions, especially for large-size problems.

Key words: Lot streaming; Flow-shop scheduling; Transfer sublots; Variable size; Bounded size; Differential evolution
https://doi.org/10.1631/FITEE.1700457 **CLC number:** TH166; TP278

1 Introduction

A flow shop is a type of manufacturing shop where each job is processed by the same group of machines, and the machine order does not change. The actions performed by a given machine may vary from job to job. The flow-shop scheduling problem (FSP) seeks to identify the best order in a manufacturing shop to process a set of jobs, where it is often desired to minimize the makespan or total flow time. Some shops deal with large orders from their customers (e.g., 2000 power strips of type *A* and 1000 of type *B* need to be produced). In this situation, when a job (or lot) consists of a batch of identical parts, it may be desirable to decompose a job into a number of transfer sublots. A sublot is a group of parts in a lot which are processed on one machine and then transferred to the next machine before the rest of the lot is completed. This was termed "lot streaming" and was first studied by Reiter (1966). With lot streaming, production may be accelerated and the work-in-process inventory may be decreased. A comprehensive review of lot streaming in flow shops (as well as other typical machine configurations such as job-shops and open-shops) can be found in Cheng et al. (2013).

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61403163) and the Zhejiang Provincial Natural Science Foundation of China (Nos. LQ14G010008 and LY15F030021)

ORCID: Hai-yan WANG, http://orcid.org/0000-0001-8289-5351

Motivated by practical applications, the lot streaming problem has received extensive attention, including the one-job (single-type job) case (Liu SC, 2003; Biskup and Feldmann, 2006; Liu JY, 2008; Sarin et al., 2008; Defersha and Chen, 2011; Mukherjee et al., 2017) and *n*-job (multi-type job) case (Kumar et al., 2000; Marimuthu et al., 2008, 2009; Tseng and Liao, 2008; Kim and Jeong, 2009; Martin, 2009; Defersha and Chen, 2010; Pan et al., 2011; Chakaravarthy et al., 2013, 2014; Ventura and Yoon, 2013; Davendra et al., 2014; Nejati et al., 2014, 2016; Han et al., 2016). Different meta-heuristic methods have been proposed to optimize the job processing sequence, such as genetic algorithm (GA) (Kumar et al., 2000; Marimuthu et al., 2008; Kim and Jeong, 2009; Martin, 2009; Defersha and Chen, 2010; Ventura and Yoon, 2013; Nejati et al., 2014, 2016), ant colony optimization (ACO) (Marimuthu et al., 2009), artificial bee colony (ABC) (Chakaravarthy et al., 2014), particle swarm optimization (PSO) (Tseng and Liao, 2008; Chakaravarthy et al., 2013), and differential evolution (DE) (Chakaravarthy et al., 2013). Most research has focused on the subset of problems with equal size sublots (ESSs). For example, Sarin et al. (2008) studied a single-lot multi-machine lot streaming FSP with ESSs to minimize a unified cost-based criterion. Mukherjee et al. (2017) considered the lot streaming problem in a single-lot two-machine flow shop with the learning effect of setup and processing time. Davendra et al. (2014) dealt with the *n*-job *m*-machine lot streaming FSP with ESSs and sequence-dependent setups, and sought to minimize the makespan. Marimuthu et al. (2008, 2009) and Chakaravarthy et al. (2013, 2014) studied the *n*-job *m*-machine problem with a makespan/total flow time criterion, where the attached setups and lot streaming with ESSs were involved. Tseng and Liao (2008), Pan et al. (2011), and Ventura and Yoon (2013) considered the total weighted earliness and tardiness penalties criteria in an *n*-job, *m*-machine lot streaming FSP with ESSs. Han et al. (2016) considered the uncertainty in machine processing time, and formulated a multi-objective lot streaming FSP model with interval processing time without an intermediate buffer.

ESSs strategy is common in practice since it is easy to implement and manage in terms of shop floor control. However, due to a lack of flexibility, the general ESSs strategy can employ smaller size sublots to achieve faster completion; almost always, as the sublot size decreases, the makespan decreases as well. This may lead to a large number of transfers to obtain the fastest completion, and may not be feasible because of transfer/handling costs between machines. A more flexible alternative to ESSs is the consistent size sublots (CSSs) strategy, where the size of a sublot remains the same from machine to machine, but may vary within a job. Fewer researchers have considered the *n*-job *m*-machine lot streaming FSP with CSSs. Kumar et al. (2000) addressed a no-wait lot streaming FSP with CSSs. Kim and Jeong (2009) solved the lot streaming problem in a no-wait flexible flow shop with detached setups and predetermined CSSs. Martin (2009) considered the interleaving of sublots from different jobs in the processing sequence in a lot streaming FSP. Nejati et al. (2014, 2016) dealt with a CSSs lot streaming problem with a work shift constraint in a hybrid flow shop and a two-stage assembly flow shop.

An even more flexible approach to lot streaming (relative to ESSs and CSSs) is variable size sublot (VSS), which allows the number of sublots and the sublot sizes to vary among machines (and jobs). In many situations, material handling systems may differ throughout the production system, which makes it necessary to consider VSSs. Fig. 1 illustrates lot streaming for a two-job three-machine flow shop under different sublot sizing options (job size, setup time, and processing time are specified in Table 1). A setup is required when there is a job change on a machine tool (time is required to change tooling, machine settings, etc.). In this example, the setup for a job on a machine can begin only when the first sublot of that job arrives at the machine. This may be termed an "attached setup." If a machine is allowed to begin setting up before the actual arrival of the first sublot of the job, it may be referred to as a "detached setup." In Fig. 1c, a schedule with unit size sublots (which is a special case of ESSs) gives the minimum makespan for all possible sublot sizing options. With VSSs, however, the same minimum makespan with fewer transfers between machines can be obtained (Fig. 1d).

Despite its relevance to production practice, few researches have examined the VSSs lot streaming for the *n*-job *m*-machine FSP. Liu (2003), Biskup and Feldmann (2006), and Defersha and Chen (2011)

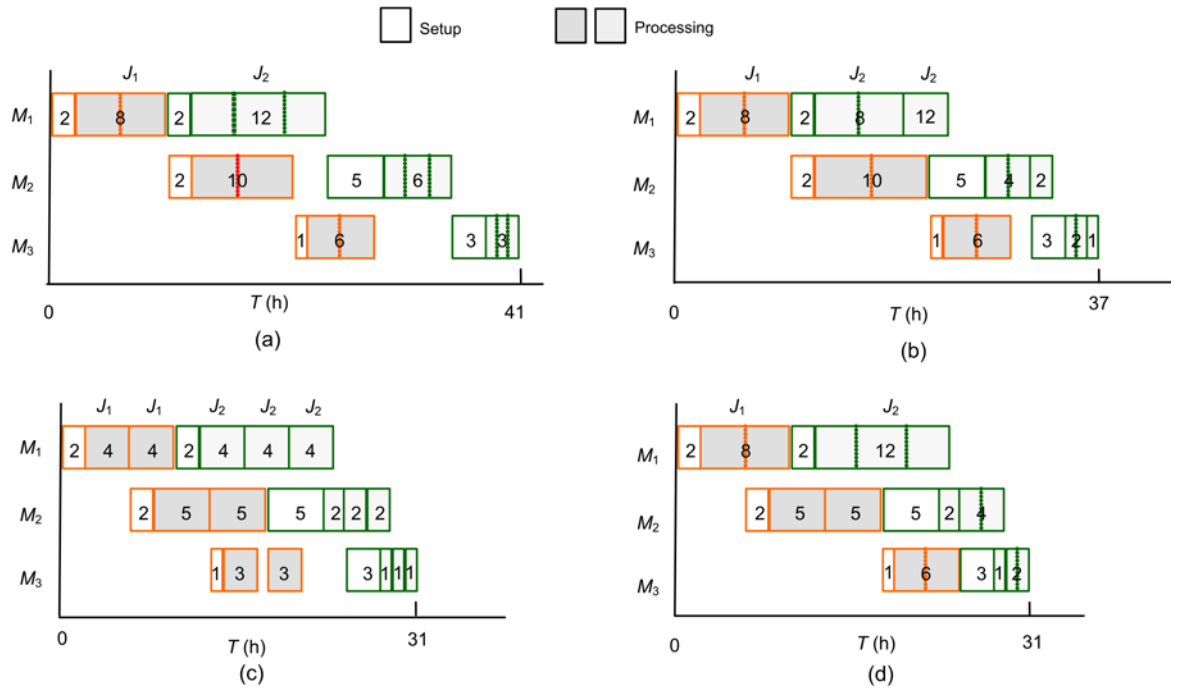

Fig. 1 An example of lot streaming with transfer sublots: (a) a schedule without lot splitting; (b) lot streaming with consistent size sublots; (c) lot streaming with unit size transfer sublots; (d) lot streaming with variable size sublots

The number in each bar refers to the time duration of the sublots

Table 1 Lot streaming problem in a two-job and three-machine flow shop

Job	Size	Setup time on each machine (h)			Unit processing time on each machine (h)		
		M_1	M_2	M_3	M_1	M_2	M_3
J_1	2	2	2	1	4	5	3
J_2	3	2	5	3	4	2	1

studied the m-machine lot streaming problem with VSSs, but they focused on the one-job case. Defersha and Chen (2010) developed a hybrid GA to solve the n-job m-machine lot streaming FSP with VSSs, where interleaving of sublots from different jobs in the processing sequence was allowed. Based on a predetermined total number of sublots (Defersha and Chen, 2010), the hybrid GA determines the number of sublots for each job, the size of each sublot on each machine, and the processing sequence of the sublots; thus, the makespan could be minimized. Despite these efforts, little attention has been devoted to the computational issues associated with production scheduling in the event of lot streaming. With the existing methods for a lot streaming FSP, the computation time heavily depends on the size of a job. That is, when a job consists of a large number of identical parts, which normally results in a large number of transfer sublots, the computational effort is likely to be significant.

In this study, we consider the n-job m-machine lot streaming FSP with VSSs, where sublots from different jobs are not allowed to interleave in the processing sequence (sometimes switching between different jobs requires a costly setup). Transfer sublot sizes may be bounded because of capacity constrained transportation (transfer or handling) activities between machines. The proposed method is separated into three stages: initial lot splitting, job sequencing optimization, and transfer adjustment. During the first stage, each job is split based on the low bound on sublot size, to obtain the fastest (or close to the fastest) completion for a processing sequence. In the second stage, a DE-based algorithm is applied to optimize the processing sequence, and the makespan/total flow time criterion is efficiently calculated to save computation time. Finally, based on the best sequence obtained through the second stage, some adjustments to sublots are undertaken to reduce the number of transfers between machines. Overall, the proposed three-stage method (TSM) seeks to minimize the makespan/total flow time, and reduce the

number of transfers without affecting the optimized makespan/total flow time. Computational experiments are conducted to demonstrate the TSM and the experiments results are compared with those of other studies.

2 Problem description and formulation

The n-job m-machine lot streaming FSP with VSSs will be described in this section. There are n jobs $J=\{J_1, J_2, ..., J_n\}$ to be processed on m machines $M=\{M_1, M_2, ..., M_m\}$, where each job must follow the same order on the machines. Each job J_i consists of a batch of identical parts, where the job size JS_i is the number of parts in the batch. Before a new job is initiated on a machine, a setup is required. To complete the set of jobs more quickly, once a group of parts has been processed on a machine, it may be transferred to the next machine as a sublot. Then processing on this sublot can begin when the next machine is available. Sublot size may be bounded and vary from machine to machine. The aim of the lot streaming FSP is to minimize the makespan/total flow time. The decision variables are the job processing sequence and the number and size of transfer sublots between each machine.

Let $\pi=\{\pi(1), \pi(2), ..., \pi(n)\}$ represent a processing sequence solution (to be optimized), where the k^{th} job in the sequence may be referred to as $\pi(k)\in J$ (k=1, 2, ..., n). A mathematical model for the n-job m-machine lot streaming FSP with VSSs is developed with the notations in Table 2. Assume that $i=\pi(k)$ (k=1, 2, ..., n), l=1, 2, ..., m, and s=1, 2, ..., $n_{i,l}$. The details are as follows:

1. Objective

$$\min C_{\max}=\mathrm{Cpt}_{i,l,s},\ i=\pi(n),\ l=m,\ s=n_{\pi(n),m}, \quad (1)$$

$$\min \mathrm{TFT}=\sum_{i=1}^{n}\mathrm{Cpt}_{i,l,s},\ l=m,\ s=n_{i,m}. \quad (2)$$

Eq. (1) specifies the objective to minimize the makespan, defined as the completion time of the last sublot on the final machine. Another criterion considered in this study is the total flow time, as shown in Eq. (2).

2. Lot splitting constraint and sublot size constraint:

$$\sum_{s=1}^{n_{i,l}}\mathrm{SZ}_{i,l,s}=\mathrm{JS}_i,\ \forall i,\ l, \quad (3)$$

$$\mathrm{SZ}_{i,l,s}\in[\mathrm{SZ}_{\min},\ \mathrm{SZ}_{\max}],\ \mathrm{SZ}_{i,l,s}\in\mathbb{R},\ \forall i,\ l,\ s. \quad (4)$$

Constraint (3) ensures that the sum of the sizes of all the transfer sublots for a job transferring to the next machine remains the same as the original job size. Constraint (4) requires sublot sizes be integers and bounded by a lower bound and an upper bound.

3. Sublot availability constraint and machine availability constraint

$$\mathrm{Sst}_{i,1}=0,\ i=\pi(1). \quad (5)$$

Assume that all jobs and machines are available at time zero. Thus, the first machine in the flow shop begins setting up for the first job in the processing sequence at time zero (Eq. (5)).

Table 2 Notations

Index and parameter	Decision variable
i: job index	π: job processing sequence
l: machine index	$C_{\max}$: makespan
s: sublot index	TFT: total flow time
n: total number of jobs	$n_{i,l}$: number of transfer sublots for job J_i when transferred to machine M_l
m: total number of machines	$\mathrm{SZ}_{i,l,s}$: size of the s^{th} sublot of job J_i when transferred to machine M_l
JS_i: size of job J_i	$\mathrm{Sst}_{i,l}$: start time of setup for job J_i on machine M_l
$\mathrm{pt}_{i,l}$: unit processing time of job J_i on machine M_l	$\mathrm{Cst}_{i,l}$: completion time of setup for job J_i on machine M_l
$\mathrm{st}_{i,l}$: setup time of job J_i on machine M_l	$\mathrm{Spt}_{i,l,s}$: start time for processing the s^{th} sublot of job J_i on machine M_l
$\mathrm{SZ}_{\min}$: a lower bound on sublot size	$\mathrm{Cpt}_{i,l,s}$: completion time for processing the s^{th} sublot of job J_i on machine M_l
$\mathrm{SZ}_{\max}$: an upper bound on sublot size	

$$\mathrm{Sst}_{i,\,l}=\mathrm{Cpt}_{i,\,l-1,\,s^*}-\mathrm{pt}_{i,\,l-1}\left(\sum_{s=1}^{s^*}\mathrm{SZ}_{i,\,l-1,\,s}-\mathrm{SZ}_{i,\,l,\,1}\right),\quad(6)$$

where $\forall l>1$, $i=\pi(1)$, and for $\forall i$, $\forall l>1$, and $s=1$, s^* is given by Eq. (7), which is shown at the bottom of this page.

Eqs. (6) and (7) address the sublot availability constraint for the first job in the sequence. That is, a machine $l>1$ may begin setting up for job $\pi(1)$ when the first sublot arrives, which is equivalent to the completion time of the last part in that sublot on the preceding machine ($l-1$). Eq. (6) provides the completion time, and Eq. (7) provides the index associated with the sublot on machine ($l-1$) that contains the last part.

$$\begin{aligned}&\mathrm{Sst}_{i,\,1}=\mathrm{Cpt}_{i',\,1,\,s},\\&i=\pi(k),\ i'=\pi(k-1),\ s=n_{i',\,1},\forall k>1.\end{aligned}\quad(8)$$

Eq. (8) is the machine availability constraint associated with the job processing sequence for the first machine, meaning that machine M_1 may begin setting up for a job after the machine finishes processing all the sublots from the previous job.

$$\begin{aligned}\mathrm{Sst}_{i,\,l}=\max\Bigg\{&\mathrm{Cpt}_{i',\,l,\,n_{i',\,l}},\\&\mathrm{Cpt}_{i,\,l-1,\,s^*}-\mathrm{pt}_{i,\,l-1}\left(\sum_{s'=1}^{s^*}\mathrm{SZ}_{i,\,l-1,\,s'}-\mathrm{SZ}_{i,\,l,\,1}\right)\Bigg\},\end{aligned}\quad(9)$$

where $\forall k>1$, $\forall l>1$, $i=\pi(k)$, $i'=\pi(k-1)$, and s^* is given by Eq. (7) for $s=1$.

Eqs. (7) and (9) consider both the sublot and machine availability constraints for the setup for job $\pi(k)$ on machine l with $l>1$ and $k>1$.

$$\mathrm{Cst}_{i,\,l}=\mathrm{Sst}_{i,\,l}+\mathrm{st}_{i,\,l},\ \forall i,\ l.\quad(10)$$

Each job requires a single setup on each machine before processing, and setups cannot be interrupted once started, as shown in Eq. (10).

$$\mathrm{Spt}_{i,\,l,\,1}=\mathrm{Cst}_{i,\,l},\ \forall i,\ l,\quad(11)$$

$$\mathrm{Spt}_{i,\,1,\,s}=\mathrm{Cpt}_{i,\,1,\,s-1},\forall i,\ \forall s>1,\quad(12)$$

$$\begin{aligned}\mathrm{Spt}_{i,\,l,\,s}=\max\Bigg\{&\mathrm{Cpt}_{i,\,l,\,s-1},\mathrm{Cpt}_{i,\,l-1,\,s^*}\\&-\mathrm{pt}_{i,\,l-1}\left(\sum_{s'=1}^{s^*}\mathrm{SZ}_{i,\,l-1,\,s'}-\sum_{s'=1}^{s}\mathrm{SZ}_{i,\,l,\,s'}\right)\Bigg\},\end{aligned}\quad(13)$$

where $\forall i$, $\forall l>1$, and $\forall s>1$, and s^* is given by Eq. (7) for $s>1$.

Eq. (11) shows that a machine can start processing the first sublot of a job right after the machine finishes the setup for that job. Eq. (12) means that machine M_1 can start processing a sublot s ($s>1$) of a job after the machine has finished processing the preceding sublot ($s-1$) of that job. Eqs. (7) and (13) consider both the sublot and machine availability constraints for processing a sublot s ($s>1$) of job i on a machine $l>1$.

$$\mathrm{Cpt}_{i,\,l,\,s}=\mathrm{Spt}_{i,\,l,\,s}+\mathrm{pt}_{i,\,l}\mathrm{SZ}_{i,\,l,\,s},\ \forall i,\ l,\ s.\quad(14)$$

All the parts in one sublot are consecutively processed on a machine. Therefore, the processing time of a sublot is defined as the sum of the processing time of all the parts in that sublot. Processing procedures cannot be interrupted once started, as shown in Eq. (14).

Note that in the above model each sublot is started as soon as possible in a sequence π, and the setup for a job on a machine can begin only when the first sublot of that job arrives at the machine (attached setup). If a detached setup is involved in the lot streaming FSP, then Eqs. (6) and (9) need to be

$$s^*=\begin{cases}n_{i,\,l-1}, & \sum_{s'=1}^{s}\mathrm{SZ}_{i,\,l,\,s'}=\mathrm{JS}_i,\\ 1, & \sum_{s'=1}^{s}\mathrm{SZ}_{i,\,l,\,s'}\le\mathrm{SZ}_{i,(l-1),\,1},\\ s^*\ \text{falls in}\ \sum_{s'=1}^{s^*-1}SZ_{i,\,l-1,\,s'}<\sum_{s'=1}^{s}SZ_{i,\,l,\,s'}\le\sum_{s'=1}^{s^*}SZ_{i,\,l-1,\,s'}, & \text{otherwise.}\end{cases}\quad(7)$$

changed to Eqs. (15) and (16), allowing a machine to set up for a job before the arrival of the first sublot of that job.

$$\mathrm{Sst}_{i,\,l}=\max\left\{0,\ \mathrm{Cpt}_{i,\,l-1,\,s^*}-\mathrm{pt}_{i,\,l-1}\cdot\left(\sum_{s'=1}^{s^*}\mathrm{SZ}_{i,\,l-1,\,s'}-\mathrm{SZ}_{i,\,l,\,1}\right)-\mathrm{st}_{i,\,l}\right\},\tag{15}$$

where $i=\pi(1)$, $\forall l>1$, and s^* is given by Eq. (7) for $s=1$.

$$\mathrm{Sst}_{i,\,l}=\max\left\{\mathrm{Cpt}_{i',\,l,\,n_{i',\,l}},\ \max\left\{0,\ \mathrm{Cpt}_{i,\,l-1,\,s^*}-\mathrm{pt}_{i,\,l-1}\left(\sum_{s'=1}^{s^*}SZ_{i,\,l-1,s'}-\mathrm{SZ}_{i,\,l,\,1}\right)-\mathrm{st}_{i,\,l}\right\}\right\},\tag{16}$$

where $\forall k>1$, $\forall l>1$, $i=\pi(k)$, and $i'=\pi(k-1)$, and s^* is given by Eq. (7) for $s=1$.

3 Preliminary analysis of lot streaming

Determining the makespan/total flow time in lot streaming is different from the case without lot streaming. Take the four cases in Fig. 1 as an example. Compared with Fig. 1a, more computations are needed to obtain the makespan/total flow time in Figs. 1b–1d, since the start and completion times for each sublot on each machine need to be calculated. The increased computational effort can be significant when the total number of sublots is large.

To reduce the computation time taken to calculate the makespan/total flow time in a lot streaming FSP, the analyses for ESSs lot streaming are carried out. Several important properties are identified, based on which efficient steps will later be developed to calculate the makespan/total flow time criterion in Section 4.2.

Consider the lot streaming of one job with four same sized transfer sublots in a five-machine flow shop (Fig. 2). The processing times for each sublot are 3, 2, 4, 1, and 2 h on machines M_1, M_2, M_3, M_4, and M_5, respectively. Assume that the setup time can be ignored, and all sublots and machines are available at time zero. There are three important properties associated with the one-job lot streaming problem:

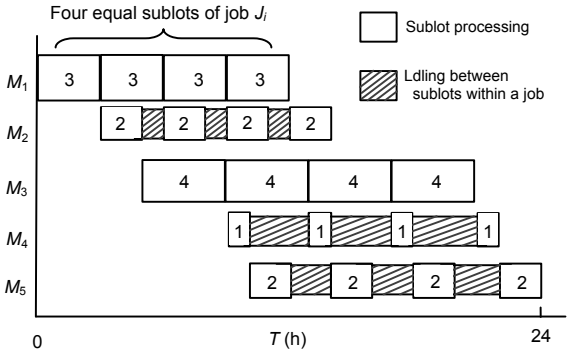

Fig. 2 Lot streaming for a one-job case with no setup
The number in each bar refers to the time duration of the sublot

Property 1 There is either no or equal idling between sublots within a job on the l^{th} machine M_l.

Property 2 The time that machine M_l starts processing the first sublot of job J_i, denoted by $\mathrm{Spt}_{i,\,l,\,1}$ (in the one-job case, $i=1$) can either be zero when $l=1$, or equal to the completion time of the first sublot on machine M_{l-1} when $l>1$.

Property 3 The completion time for the last sublot of job J_i on machine M_l, denoted by $\mathrm{Cpt}_{i,\,l,\,n_{i,\,l}}$, is either the start time of the first sublot $\mathrm{Spt}_{i,\,l,\,1}$ plus the sum of the processing times for all the sublots, if there is no idling between the sublots, or the completion time of the last sublot on the preceding machine $\mathrm{Cpt}_{i,\,l-1,\,n_{i,\,l}}$ plus the processing time for one sublot on machine M_l if there is idling. That is,

$$\mathrm{Cpt}_{i,\,l,\,n_{i,\,l}}=\max\{\mathrm{Spt}_{i,\,l,\,1}+\mathrm{pt}_{i,\,l}\mathrm{JS}_i,\ \mathrm{Cpt}_{i,\,l-1,\,n_{i,\,l}}+\mathrm{pt}_{i,\,l}\mathrm{SZ}_{i,\,l,\,n_{i,\,l}}\}.\tag{17}$$

From Properties 1–3, the start time for the first sublot of a job and the completion time for the last sublot of that job on each machine can be obtained. This means that there is no need to calculate the start and completion times of each sublot on every machine to obtain the makespan/total flow time.

If we consider the lot streaming of job J_i in a processing sequence in an n-job case, each machine becomes available for J_i after it finishes processing all the sublots from the preceding job in the sequence

(Fig. 3). The completion time of the jobs (the time at which a machine becomes available for processing job J_i) is denoted by AT_l. There might be inter-sublot idling during the periods associated with the light and dark grey bars.

The machine availability constraint determines when the processing of the four same sublots of job J_i may begin (Fig. 3). The amount of idling, which remains the same or is reduced relative to the situation in Fig. 2, depends on the constraint. For the case shown, the idle period is reduced on M_2, M_4, and M_5. Considering the difference between Figs. 2 and 3, the initial idling period between sublots on a machine is first affected, and the last idling period is affected only when there is no idling between all the preceding sublots for a job. The same effect can be caused by machine setup before the first sublot of a job. As long as there is an idling period before processing the last sublot of a job, the last sublot of the job will begin at the time when the sublot completes processing on the preceding machine. Therefore, Property 3 still applies to the n-job case. This means for an n-job lot streaming FSP with setup time and ESSs, there is an efficient way to obtain the makespan/total flow time by focusing on only the start and completion times for the first and the last sublots for each job on every machine.

4 Three-stage method (TSM)

For a lot streaming FSP with an unbounded sublot size, when each part (unit) in each job is regarded as a separate transfer sublot between any successive machines, the minimum makespan/total flow time can be achieved for a processing sequence. For a lot streaming FSP with a bounded sublot size, if the size of a job is divisible by the lower bound SZ_{min}, splitting a job into a set of identical sublots, whose size is equal to SZ_{min}, would lead to the fastest completion. Based on this idea, we could first undertake lot splitting based on the lower bound on sublot size, and then find the optimal processing sequence of jobs based on the initial splitting to minimize the makespan/total flow time. In addition, considering that the size of sublots can be variable, based on the obtained optimal sequence, the possibility of merging sublots to be transferred together under the upper bound constraint may be explored, with the goal of reducing the number of transfers between machines without affecting the optimized makespan/total flow time. In summary, the problem-solving process can be broken into three stages: initial lot splitting, job sequencing optimization with efficient calculation of the makespan/total flow time criterion, and transfer adjustment. The framework of the TSM is illustrated in Fig. 4.

4.1 Initial lot splitting

As discussed above, when the size of job J_i is divisible by the lower bound SZ_{min}, the job can be split into ESSs to obtain the fastest completion. In this case, the initial splitting is $n_{i,l}=JS_i/SZ_{min}$ and $SZ_{i,\,l,\,s}=SZ_{min}$, where $l=1, 2, ..., m$ and $s=1, 2, ..., n_{i,\,l}$.

If the size of J_i is not divisible by SZ_{min}, then the job may be split into the following CSSs:

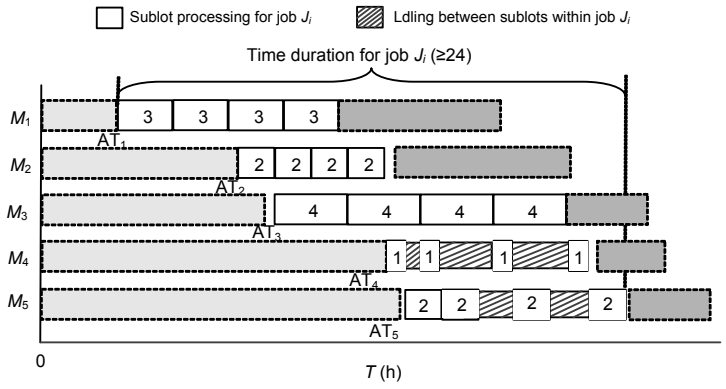

Fig. 3 Lot streaming of job J_i in a multi-job case

The number in each bar refers to the time duration of the sublots. The light grey bars are associated with the processing of the sublots from preceding jobs. The dark grey bars are associated with the processing of jobs after J_i

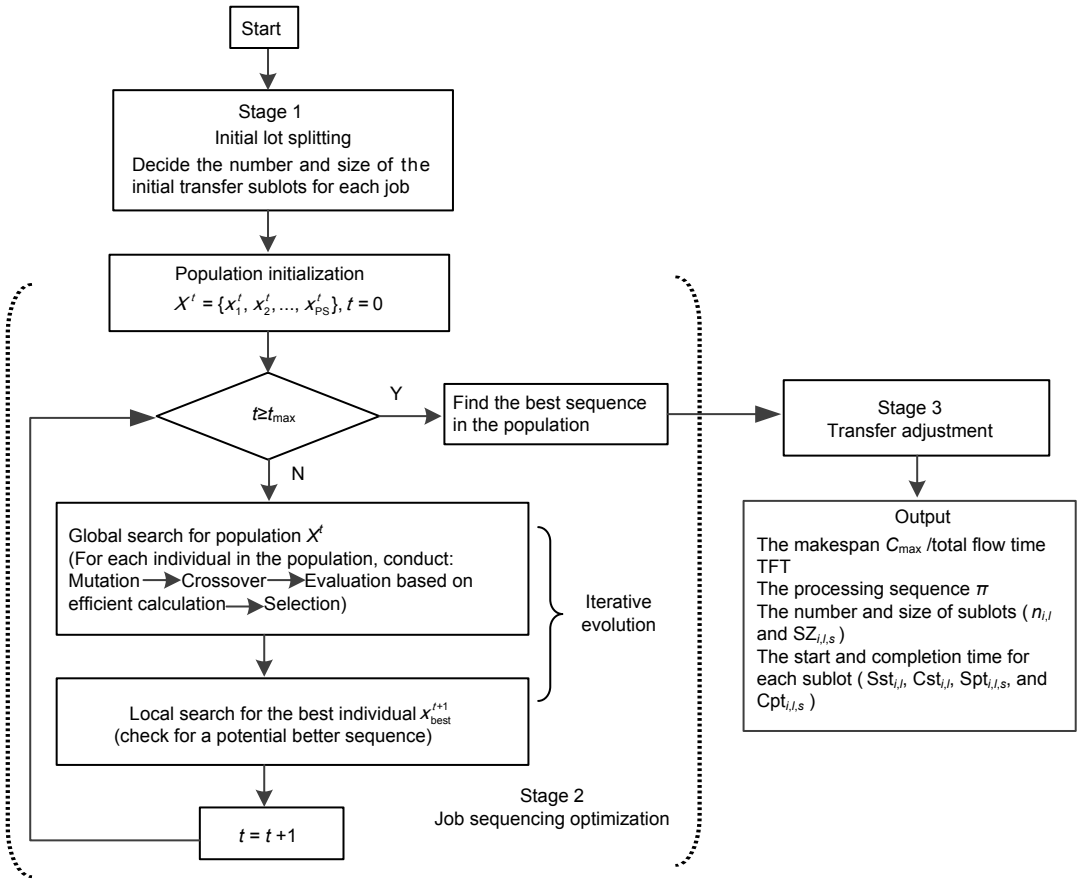

Fig. 4 Framework of the three-stage method

$$n_{i,l} = \left\lfloor \frac{\mathrm{JS}_i}{\mathrm{SZ}_{\min}} \right\rfloor, \forall l, \tag{18}$$

$$\mathrm{SZ}_{i,l,s} = \left\lceil \frac{\mathrm{JS}_i - n_{i,l} \cdot \mathrm{SZ}_{\min}}{n_{i,l}} \right\rceil + \mathrm{SZ}_{\min}, \forall l,\ s = 1, 2, \ldots, s_i', \tag{19}$$

$$\mathrm{SZ}_{i,l,s} = \left\lceil \frac{\mathrm{JS}_i - n_{i,l} \cdot \mathrm{SZ}_{\min}}{n_{i,l}} \right\rceil + \mathrm{SZ}_{\min}, \quad \forall l,\ s = s_i' + 1, \ldots, n_{i,l}, \tag{20}$$

$$s_i' = \mathrm{JS}_i - n_{i,l}\mathrm{SZ}_{\min} - \left\lfloor \frac{\mathrm{JS}_i - n_{i,l} \cdot \mathrm{SZ}_{\min}}{n_{i,l}} \right\rfloor n_{i,l}, \tag{21}$$

where $\lfloor b \rfloor$ means the largest integer less than or equal to b and $\lceil b \rceil$ is the smallest integer greater than or equal to b.

Note that both Eqs. (22) and (23) have to be satisfied that all the sublot sizes are within [$\mathrm{SZ}_{\min}$, $\mathrm{SZ}_{\max}$]. With the initial splitting, when all the job sizes are divisible by $\mathrm{SZ}_{\min}$ (e.g., when $\mathrm{SZ}_{\min}$=1), a minimum makespan/total flow time can be achieved for a processing sequence:

$$\mathrm{JS}_i \geq \mathrm{SZ}_{\min}, \ \forall i, \tag{22}$$

$$\left\lceil \frac{\mathrm{JS}_i - \left\lfloor \frac{\mathrm{JS}_i}{\mathrm{SZ}_{\min}} \right\rfloor \mathrm{SZ}_{\min}}{\left\lfloor \frac{\mathrm{JS}_i}{\mathrm{SZ}_{\min}} \right\rfloor} \right\rceil + \mathrm{SZ}_{\min} \leq \mathrm{SZ}_{\max}, \ \forall i. \tag{23}$$

4.2 Job sequencing optimization with efficient calculation

Based on the initial lot splitting, the second stage aims to find the optimal processing sequence of jobs. A DE-based algorithm is used to minimize the makespan/total flow time. DE is a stochastic and efficient population-based heuristic proposed by Storn and Price (1997), which has been successfully

$$\mathrm{Cpt}_{i,\,l,\,n_{i,\,l}}=\begin{cases}\mathrm{Spt}_{i,\,l,\,1}+\mathrm{pt}_{i,\,l}\mathrm{JS}_i, & l=1,\\ \max\{\mathrm{Spt}_{i,l,1}+\mathrm{pt}_{i,\,l}\mathrm{JS}_i,\ \mathrm{Cpt}_{i,\,l-1,\,n_{i,\,l}}+\mathrm{pt}_{i,\,l}\mathrm{SZ}_{i,\,l,\,n_{i,\,l}}\}, & l>1,\end{cases} \tag{24}$$

$$\mathrm{Cpt}_{i,\,l,\,s_i'}=\begin{cases}\mathrm{Spt}_{i,\,l,\,1}+\mathrm{pt}_{i,\,l}s_i'\mathrm{SZ}_{i,\,l,\,1}, & l=1,\\ \max\{\mathrm{Spt}_{i,\,l,\,1}+\mathrm{pt}_{i,\,l}s_i'\mathrm{SZ}_{i,\,l,\,1},\ \mathrm{Cpt}_{i,\,l-1,\,s_i'}+\mathrm{pt}_{i,\,l}\mathrm{SZ}_{i,\,l,\,s_i'}\}, & l>1,\end{cases} \tag{25}$$

$$\mathrm{Spt}_{i,\,l,s'+1}=\begin{cases}\mathrm{Cpt}_{i,\,l,\,s_i'}, & l=1,\\ \max\{\mathrm{Cpt}_{i,\,l,\,s'},\ \mathrm{Spt}_{i,\,l-1,\,s'+1}+\mathrm{pt}_{i,\,l-1}\mathrm{SZ}_{i,\,l-1,\,s'+1}\}, & l>1,\end{cases} \tag{26}$$

$$\mathrm{Cpt}_{i,\,l,\,n_{i,\,l}}=\begin{cases}\mathrm{Spt}_{i,\,l,\,s'+1}+\mathrm{pt}_{i,\,l}(n_{i,\,l}-s')\mathrm{SZ}_{i,\,l,\,s'+1}, & l=1,\\ \max\{\mathrm{Spt}_{i,\,l,\,s'+1}+\mathrm{pt}_{i,\,l}(n_{i,l}-s')\mathrm{SZ}_{i,\,l,\,s'+1},\ \mathrm{Cpt}_{i,\,l-1,\,n_{i,\,l}}+\mathrm{pt}_{i,\,l}\mathrm{SZ}_{i,\,l,n_{i,\,l}}\}, & l>1.\end{cases} \tag{27}$$

used to solve production scheduling problems (Onwubolu and Davendra, 2006; Wang et al., 2010; Chakaravarthy et al., 2013; Tasgetiren et al., 2013). The flowchart of the proposed DE-based optimization is illustrated in Fig. 4. Details will be provided in this subsection.

4.2.1 Individual representation and population initialization

In DE optimization, the evolution is an iterative process, starting from an initial population of randomly generated candidate solutions (individuals). Through the iterative evolution, the initial population is evolved towards better solutions from one generation to the next. The algorithm terminates when the maximum number of generations $t_{\max}$ can be completed.

Let $X^t=\{x_1^t, x_2^t, ..., x_{\mathrm{PS}}^t\}$ be the populations of the t^{th} generation, where PS is the population size and $x_h^t=\{x_{h,\,1}^t, x_{h,\,2}^t, ..., x_{h,\,n}^t\}$ is the h^{th} individual in the populations (t=0, 1, …, $t_{\max}$). Here, $x_{h,\,i}^t\in[0,1]$ is the processing priority index of job J_i. During population initialization, we randomly generate PS individuals to construct an initial population X^0, where each individual x_h^0 (h=1, 2, ..., PS) consists of n uniformly distributed random real numbers within [0, 1].

A processing sequence π can be obtained for an individual x_h^t by ranking the processing priority index $x_{h,\,i}^t$, i=0, 1, ..., n; for example, if x_h^t={0.31, 0.40, 0.75, 0.28}, then π={3, 2, 1, 4}.

4.2.2 Efficient calculation of the makespan/total flow time criterion

According to the preliminary analysis in Section 3, there is an efficient way to obtain the makespan/total flow time, where we need to calculate only the start and completion times for the first and the last sublot for each job on every machine. Note that the properties apply only to lot streaming with ESSs. However, after the initial lot splitting in Section 4.1, each job is split into ESSs or CSSs, depending on whether a job's size JS_i is divisible by $\mathrm{SZ}_{\min}$ or not.

1. If JS_i is divisible by $\mathrm{SZ}_{\min}$, job J_i is split into ESSs. Then to calculate the makespan/total flow time, we need to compute only the start time of the first sublot and the completion time of the last sublot for job J_i on each machine according to Property 3.

2. If JS_i is not divisible by $\mathrm{SZ}_{\min}$, job J_i is split into CSSs. According to Eqs. (18)–(21), there are only two cases for the sublot sizes; thus, we can take the lot splitting process as two parts of ESSs: the first s_i' equal sublots with the size specified in Eq. (19), and the rest $(n_{i,\,l}-s_i')$ equal sublots with the size specified in Eq. (20). Thereby, we can apply Property 3 to the two parts, which means that we need to calculate the completion time for the $(s_i')^{\mathrm{th}}$ sublot and the start time for the $(s_i'+1)^{\mathrm{th}}$ sublot.

Based on the above-mentioned analysis, the following efficient calculation steps are developed to calculate the makespan/total flow time criterion for an n-job m-machine lot streaming FSP with setup time.

Step 1: Obtain the processing sequence π={π(1), π(2), ..., π(n)} from an individual that is being evaluated. Set the time when a machine becomes available as AT_l=0 for each l=1, 2, ..., m. Set k=1 and TFT=0.

Step 2: Set i=π(k) and l=1.

Step 3: If l=1, then $\mathrm{Sst}_{i,\,l}$=AT_l, meaning that M_1 can set up for job J_i as soon as the machine is available; otherwise, considering the sublot availability

constraint of $\mathrm{Sst}_{i,\,l}=\max\{\mathrm{AT}_l,\ \mathrm{Cpt}_{i,\,l-1,\,1}-\mathrm{st}_{i,\,l}\}$ for the attached setup, or $\mathrm{Sst}_{i,\,l}=\max\{\mathrm{AT}_l,\ \mathrm{Cpt}_{i,\,l-1,\,1}-\mathrm{st}_{i,\,l}\}$ for the detached setup.

Step 4: The start time for processing the first sublot of J_i on M_l is $\mathrm{Spt}_{i,\,l,\,1}=\mathrm{Cst}_{i,\,l}=\mathrm{Sst}_{i,\,l}+\mathrm{st}_{i,l}$.

Step 5: If the size of J_i is divisible by $\mathrm{SZ}_{\min}$, the completion time for the last sublot of J_i on M_l can be determined according to Property 3, as shown in Eq. (24); otherwise, according to the above-mentioned discussion, we need to calculate the completion time for the $(s')^{\text{th}}$ sublot and the start time for the $(s_i'+1)^{\text{th}}$ sublot of J_i on M_l using Eqs. (25) and (26), where s' is specified in Eq. (21). Then the completion time for the last sublot of J_i on M_l can be calculated by Eq. (27).

Step 6: Update $\mathrm{AT}_l=\mathrm{Cpt}_{i,\,l,\,n_{i,\,l}}$. Execute $l=l+1$. If $l\leq m$, return to step 3.

Step 7: Execute $\mathrm{TFT}=\mathrm{TFT}+\mathrm{AT}_m$ and $k=k+1$. If $k\leq n$, return to step 2; otherwise, obtain the makespan $\mathrm{C}_{\max}=\mathrm{AT}_m$.

The time complexity for the above calculation steps is $O(mn)$. This means that no matter how large the total number of sublots is, the calculation time nearly remains the same for the given n and m. It is more efficient than the traditional way to calculate the makespan/total flow time criterion.

4.2.3 Global search

During each round of iterative evolution, three operators, mutation, crossover, and evaluation and selection are conducted for each individual x_h^t the population X^t (h=1, 2, ..., PS) to produce the population of the next generation X^{t+1}.

1. Mutation

For individual x_h^t, a mutated individual $v_h^{t+1}=\{v_{h,\,1}^{t+1}, v_{h,\,2}^{t+1}, \ldots, v_{h,\,n}^{t+1}\}$ is generated by

$$v_h^{t+1}=x_{d_1}^t+F(x_{d_2}^t-x_{d_3}^t)v, \tag{28}$$

where d_1, d_2, and d_3 are three different integers randomly generated within [1, PS], and are different from h. $F\in[0, 2]$ is a real constant scaling factor.

If $v_{h,\,i}^{t+1}$ (i=1, 2, ..., n) obtained through mutation exceeds [0, 1], then perform

$$v_{h,\,i}^{t+1}=\begin{cases}-v_{h,\,i}^{t+1}+\left\lfloor v_{h,\,i}^{t+1}\right\rfloor, & v_{h,\,i}^{t+1}<0,\\ 2-v_{h,\,i}^{t+1}+\left\lfloor v_{h,\,i}^{t+1}-1\right\rfloor, & v_{h,\,i}^{t+1}>1.\end{cases} \tag{29}$$

2. Crossover generation

For each h, randomly generate an integer within [1, n] as j. Perform a crossover operation between x_h^t and x_h^{t+1} by

$$u_{h,\,i}^{t+1}=\begin{cases}v_{h,\,i}^{t+1}, & r_i\leq \mathrm{CR}\text{ or }i=j,\\ x_{h,\,i}^t, & \text{otherwise.}\end{cases} \tag{30}$$

A trial individual $u_h^{t+1}=\{u_{h,\,1}^{t+1}, u_{h,\,2}^{t+1}, \ldots, u_{h,\,n}^{t+1}\}$ is obtained. $\mathrm{CR}\in[0, 1]$ is the crossover probability and r_i (i=1, 2, ..., n) is a uniformly distributed random value within [0, 1].

3. Evaluation and selection

Use the efficient calculation steps in Section 4.2.2 to evaluate and choose one individual among x_h^t, x_h^{t+1}, and u_h^{t+1} as x_h^{t+1} in the next generation using a greedy selection criterion. That is, whichever the individual has the smallest makespan/total flow time is selected.

4.2.4 Local search

To obtain a better performance, the best individual in X^{t+1}, denoted as $x_{\mathrm{best}}^{t+1}=\{x_{\mathrm{best},\,1}^{t+1}, x_{\mathrm{best},\,2}^{t+1}, \ldots, x_{\mathrm{best},\,n}^{t+1}\}$, is selected to perform the interchange-based local search procedure according to the following steps:

Step 1: Randomly generate an integer within [1, n] and denote it as i_1. Set k=1 and a temporary individual $x'_{\mathrm{temp}}=x_{\mathrm{best}}^{t+1}$ ($x'_{\mathrm{temp}}=\{x'_{\mathrm{temp},1}, x'_{\mathrm{temp},\,2}, \ldots, x'_{\mathrm{temp},\,n}\}$).

Step 2: Randomly generate an integer within [1, n] which is different from i_1. Denote the integer as i_2. Interchange the values between $x'_{\mathrm{temp},\,i_1}$ and $x'_{\mathrm{temp},\,i_2}$ on the temporary individual x'_{temp}, which means jobs i_1 and i_2 are interchanged in the processing sequence.

Step 3: Evaluate the newly obtained x'_{temp} using the calculation steps in Section 4.2.2. If x'_{temp} has a smaller makespan/total flow time than x_{best}^{t+1}, set $x_{\mathrm{best}}^{t+1}=x'_{\mathrm{temp}}$.

Step 4: Set k=k+1. If $k \le n$, return to step 2; otherwise, the local search for the best individual is finished.

4.3 Transfer adjustment

In this subsection, based on the best sequence obtained in Section 4.2, we make some adjustments to the initial transfer sublots to reduce the number of transfers between machines without affecting the obtained optimized makespan/total flow time. That is, some sublots can be merged and transferred together under the upper bound size constraint. The initial transfer sublots become VSSs after the adjustment.

First, use the traditional calculation method to obtain the start and completion times for each sublot on each machine under the best sequence $\pi=\{\pi(1), \pi(2), ..., \pi(n)\}$. To maintain the makespan/total flow time during transfer adjustment, we start by adjusting the last job $\pi(n)$ in the sequence, then job $\pi(n-1)$, and so forth. After transfer adjustments for each job (from the last to the first) in the best sequence, the final transfer sublots and the new start and completion times can be obtained.

An example of lot streaming may be used to illustrate the transfer adjustment process of a job in a processing sequence (Fig. 5a). Job $\pi(k)$ in Fig. 5a is split into four equal transfer sublots with size SZ_{min} after initial splitting. Assume that $SZ_{max}=2\times SZ_{min}$. First, we delay the processing of the four sublots on the last machine M_3 as much as possible. If the makespan criterion is considered (Fig. 5b), the first two sublots can be merged when they are transferred from M_2 to M_3, since they both finish processing on M_2 before the delayed start time of the first sublot on M_3. Similarly, the remaining two sublots can be merged on M_3. The initial transfer sublots can be adjusted into two sublots, each of which with a size of $2\times SZ_{min}$ when transferred to machine M_3. The size of the emerged sublot will not exceed the upper bound SZ_{max}.

Second, we delay processing sublots of job $\pi(k)$ on M_2 as much as possible without affecting the new start times of all the sublots on M_3. The second and the third sublots can be merged when they are transferred from M_1 to M_2 (Fig. 5c). The final transfer sublots for job $\pi(k)$ after adjustment are shown in Fig. 5d. In this example, after adjustment on each machine, the total number of transfers for job $\pi(k)$ from M_1 to M_2 and M_2 to M_3 is reduced from 8 to 5, while the makespan criterion remains the same as before. The reduction would be significant when the job is large in size.

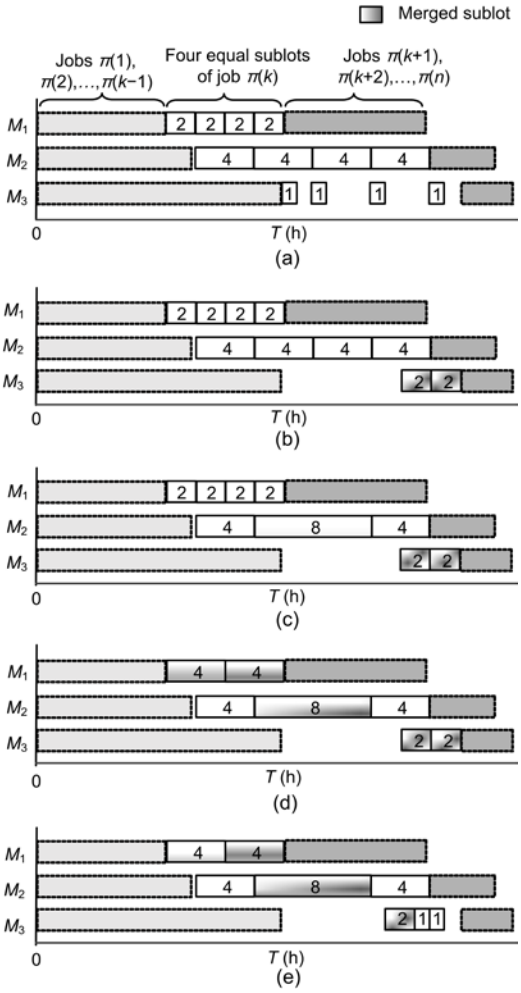

Fig. 5 Transfer adjustment for job $\pi(k)$ in a processing sequence: (a) lot streaming of job $\pi(k)$ in a processing sequence with other jobs; (b) delaying processing sublots of job $\pi(k)$ on M_3 without affecting the makespan; (c) delaying processing sublots of job $\pi(k)$ on M_2 without affecting the makespan; (d) the final transfer sublots of job $\pi(k)$ for the makespan criterion; (e) transfer adjustment of job $\pi(k)$ for the total flow time criterion

The number in each bar refers to the time duration of the sublot

When the total flow time criterion is considered in this example, the completion time of job $\pi(k)$ on the last machine should not be affected when delaying processing sublots during the adjustment. Fig. 5e shows the final transfer sublots of job $\pi(k)$ after the transfer adjustment for the total flow time criterion.

4.4 Time complexity of the method

In the TSM, most of the total computation time is attributed to stage 2, where iterative evolution is involved. Therefore, we focus on the analysis of time complexity in stage 2. The computation time of the DE-based optimization is spent mostly in iterative evolution, which includes global and local searches. Mutation, crossover, and selection are carried out in the global search. The time complexity for every mutation and crossover is $O(n)$. During the selection procedure, evaluation is required to obtain the makespan/total flow time criterion for the mutated and trial individuals, and the time complexity is $O(mn)$ for each individual. Considering the population size PS, the total time complexity of the global search is PS($2O(n)$+$2O(mn)$).

During the local search, the best individual is selected to perform the interchange procedure, which occurs n times. An evaluation is required each time and two jobs in the processing sequence are interchanged. Therefore, the local search has a time complexity of $O(n^2m)$.

Both the global and local searches are performed $t_{\max}$ times; so, the total time complexity of the proposed algorithm is

$$\begin{aligned} &O(t_{\max},\ \mathrm{PS},\ n,\ m)\\ &\approx t_{\max}\mathrm{PS}(2O(n)+2O(mn))+t_{\max}O(n\cdot n\cdot m)\\ &\approx t_{\max}\mathrm{PS}O(mn)+t_{\max}O(n^2m). \end{aligned} \tag{31}$$

We can see that the maximum number of iterations, the population size, and the number of jobs and machines determine the computational burden of the algorithm. Note that the time complexity has little relation with sizes of production jobs (lots) according to the efficient calculation developed in Section 4.2.2.

5 Experiments and performance analysis

Five different datasets from the appendices of Marimuthu et al. (2008) and Chakaravarthy et al. (2013) were used to evaluate the performance of the TSM on a lot streaming FSP. Each dataset in Marimuthu et al. (2008) was a problem with n=50 and m=7, and m was extended by Chakaravarthy et al. (2013) to 10. Smaller-sized problems can be generated from the same dataset. For any $n'\leq 50$ and $m'\leq 10$, the data up to the n'^{th} column and $(2m'+1)^{\text{th}}$ row in a dataset provide information about job size, setup time, and processing time for a problem with n' jobs and m' machines. The datasets were initially generated for lot streaming with an unbounded sublot size and attached setups.

5.1 Experiments on lot streaming with an unbounded sublot size

In the problem formulated in Section 2, when $SZ_{\min}$=1 and $SZ_{\max}$ is unlimited, the problem becomes lot streaming with an unbounded sublot size. In this case, Baker (1995) provided an optimal makespan for the two-machine problem. Therefore, we first tested the TSM in the two-machine problem to see whether it could obtain the same makespan as Baker's algorithm (BA), and how fast and stable TSM can achieve the optimal makespan. Then we conducted experiments on the multi-machine problem for the makespan/total flow time criterion, and compared the results with those from the GA and the hybrid genetic algorithm (HEA) proposed by Marimuthu et al. (2008), and the differential evolution algorithm (DEA) and PSO proposed by Chakaravarthy et al. (2013).

The TSM was coded with Visual C++.NET 2010 and run on a personal computer with an Intel i5-3320M CPU, 2.60-GHz processor, and 4.00-GB RAM. Parameters involved in the algorithm in stage 2 were set as PS=n, $t_{\max}$=100n, CR=0.1, and F=0.7. The maximum number of iterations and the population size were set as the values used by Marimuthu et al. (2008). The results obtained over 30 runs are presented in Tables 3–7. The makespan/total flow time marked in bold refers to the minimum makespan/total flow time.

From Table 3, we can see that TSM (the three-stage method) can always obtain the optimal makespan with fast convergence in each run for the two-machine problem. For many cases in Tables 4–7, TSM outperforms GA, DEA, PSO, and HEA, especially when the size of the problem becomes large.

The CPU time consumed by TSM, GA, and HEA presented in Tables 4 and 5 are not comparable, since the computers with different performances may have been used by Marimuthu et al. (2008); it is unclear which calculation method was used by Marimuthu et al. (2008).

However, from the computation time consumed by GA and HEA for the 30-job problem and the increasing rate of computation time with the increase of the number of machines, we infer that Marimuthu et al. (2008) may have used the traditional calculation, which means that in addition to the number of machines, the size of jobs (or the number of sublots) also affects the computation time. With the efficient calculation steps developed in Section 4.2.2, however, the CPU time of TSM slowly increases as the number of machines increases. Since we set the parameters PS=n and t_{max}=100n, the computation time of TSM becomes proportional to mn^3, according to the time complexity analysis in Section 4.4. Therefore, compared with the number of machines, the number of jobs has a greater impact on computation time.

When optimizing for the total flow time criterion, it takes longer for TSM to reach convergence, and the standard deviations are larger compared with the results of TSM when optimizing the makespan

Table 3 Performance of TSM on the two-machine problem with the makespan criterion

n	Dataset	BA	TSM				n	Dataset	BA	TSM			
			BM	SD	AG (s)	AT (s)				BM	SD	AG (s)	AT (s)
15	1	**429**	**429**	0	0.0013	0.96	35	1	**914**	**914**	0	0.0030	6.26
	2	**363**	**363**	0	0.0007	0.99		2	**772**	**772**	0	0.0143	6.30
	3	**249**	**249**	0	0.0007	0.99		3	**709**	**709**	0	0.0024	6.35
	4	**291**	**291**	0	0.0008	0.98		4	**720**	**720**	0	0.0028	6.31
	5	**305**	**305**	0	0.0011	0.98		5	**892**	**892**	0	0.0052	6.30
25	1	**648**	**648**	0	0.0016	2.37	50	1	**1258**	**1258**	0	0.0042	15.56
	2	**581**	**581**	0	0.0045	2.37		2	**1153**	**1153**	0	0.0180	15.54
	3	**436**	**436**	0	0.0012	2.39		3	**1013**	**1013**	0	0.0037	15.51
	4	**489**	**489**	0	0.0015	2.36		4	**1035**	**1035**	0	0.0056	15.51
	5	**671**	**671**	0	0.0013	2.36		5	**1250**	**1250**	0	0.0146	15.52

BA: Baker's algorithm; BM: best makespan over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs

Table 4 A comparison with DEA, PSO, GA, and HEA for the makespan criterion in the 30-job problem

m	Dataset	DEA	PSO	GA		HEA		TSM			
				BM	AT (s)	BM	AT (s)	BM	SD	AG (s)	AT (s)
3	1	**796**	**796**	803	22.25	**796**	36.37	**796**	0	0.0199	3.29
	2	**691**	**691**	**691**		**691**		**691**	0	0.0022	3.31
	3	**625**	**625**	**625**		**625**		**625**	0	0.0041	3.32
	4	**623**	**623**	**623**		**623**		**623**	0	0.0048	3.30
	5	**770**	**770**	**770**		**770**		**770**	0	0.0209	3.28
5	1	851	854	843	35.21	835	52.34	**827**	0.77	0.8499	3.31
	2	748	749	**747**		**747**		**747**	0	0.0574	3.36
	3	638	634	**627**		**627**		**627**	0	0.0387	3.42
	4	**677**	**677**	**677**		**677**		**677**	0	0.0064	3.38
	5	790	796	788		782		**774**	0	0.2142	3.39
7	1	888	896	878	50.92	870	65.20	**855**	0	0.4290	3.55
	2	766	778	761		758		**751**	0.43	0.9918	3.44
	3	702	709	701		696		**686**	0	0.2224	3.59
	4	690	697	**685**		**685**		**685**	0	0.0423	3.50
	5	821	829	810		796		**786**	0.81	0.9302	3.59

The results of DEA and PSO are from Chakaravarthy et al. (2013); the results of GA and HEA are from Marimuthu et al. (2008); the makespan/total flow time marked in bold refers to the minimum makespan/total flow time; the underlined numbers are improved makespans. BA: Baker's algorithm; BM: best makespan over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs

criterion (Tables 4–7). This is probably because the total flow time (the sum of the job completion times) is larger than the makespan (the time of the last job completion) for the same problem, and there tends to be more possible processing sequences that lead to the same flow time in the solution space.

If we allow transfer sublots to be a variable size instead of a unit size, TSM can provide a solution with fewer transfers between machines. Take the 15×2 problem with the makespan criterion in the No. 1 dataset as an example. From the Gantt chart illustrated in Fig. 6, the total number of transfers from M_1 to M_2 in the problem is reduced from 85 to 21.

In Fig. 6, the position and length of a bar reflect the start time, duration, and completion time of sublot processing. For the first sublot of a job on a machine, the setup activity is included. The bar reflects both the job setup and the sublot processing time.

Table 5 A comparison with DEA, PSO, GA, and HEA for the total flow time criterion in the 30-job problem

m	Dataset	DEA	PSO	GA		HEA		TSM			
				BF	AT (s)	BF	AT (s)	BF	SD	AG (s)	AT (s)
3	1	11 205	11 135	9829	23.51	9722	29.83	**9573**	14.908	2.5971	3.53
	2	8794	8994	7361		7353		**7229**	4.616	2.1886	3.43
	3	8743	8644	7589		7544		**7448**	7.830	2.5135	3.50
	4	8466	8555	7616		7512		**7365**	5.579	2.3435	3.45
	5	10 192	10 087	8716		8681		**8467**	9.724	1.7883	3.42
5	1	12 847	12 871	11 165	37.46	11 165	49.49	**10 949**	20.388	2.8396	3.80
	2	10 532	10 527	8948		8880		**8564**	11.409	2.4266	3.65
	3	9598	9759	8578		8513		**8304**	19.402	2.8864	3.80
	4	9874	9836	8690		8616		**8364**	16.260	2.4867	3.73
	5	10 919	11 175	9564		9379		**9179**	8.098	2.5279	3.65
7	1	13 696	13 767	12 039	51.85	11 949	67.56	**11 727**	26.235	2.9601	4.06
	2	11 391	11 253	9894		9745		**9414**	12.952	2.8480	3.90
	3	10 838	10 757	9517		9472		**9279**	21.890	2.9945	4.05
	4	10 748	10 682	9541		9377		**9026**	15.132	2.7263	3.96
	5	11 737	11 706	10 260		10 164		**9889**	18.273	2.7974	3.94

The underlined numbers are improved total flow times. BF: best total flow time over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs. The results of DEA and PSO are from Chakaravarthy et al. (2013); the results of GA and HEA are from Marimuthu et al. (2008); the makespan/total flow time marked in bold refers to the minimum makespan/total flow time

Table 6 A comparison with DEA, and PSO for the makespan criterion in the 50-job problem

m	Dataset	DEA	PSO	TSM				*m*	Dataset	DEA	PSO	TSM			
				BM	SD	AG (s)	AT (s)					BM	SD	AG (s)	AT (s)
3	1	**1345**	**1345**	**1345**	0	0.0078	16.22	7	1	1374	1384	**1354**	0	1.9212	19.33
	2	1155	1155	**1154**	0	0.1556	16.10		2	1357	1361	**1332**	0	1.4852	19.48
	3	**1057**	**1057**	**1057**	0	0.0244	16.14		3	1169	1170	**1133**	0.24	7.5705	19.36
	4	**1148**	**1148**	**1148**	0	0.0135	16.12		4	1225	1228	**1189**	0	1.3111	19.26
	5	1254	1258	**1251**	0	0.5900	15.99		5	1340	1358	**1265**	0.84	11.261	19.43
5	1	1349	1352	**1347**	0	0.1541	18.81	10	1	1458	1475	**1370**	4.68	14.914	22.80
	2	1334	1336	**1329**	0	0.3069	18.78		2	1422	1422	**1338**	1.85	13.196	22.83
	3	1076	1075	**1059**	0	0.2677	18.76		3	1269	1280	**1195**	4.31	14.338	22.61
	4	1190	1197	**1186**	0	0.0749	18.73		4	1307	1306	**1230**	2.75	14.568	22.65
	5	1305	1316	**1258**	0	6.2930	18.81		5	1414	1427	**1316**	3.72	14.591	22.81

The results of DEA and PSO are from Chakaravarthy et al. (2013); the makespan/total flow time marked in bold refers to the minimum makespan/total flow time; the underlined numbers are improved makespans. BM: best makespan over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs

Table 7 A comparison between DEA and PSO for the total flow time criterion in the 50-job problem

m	Dataset	DEA	PSO	TSM				*m*	Dataset	DEA	PSO	TSM			
				BF	SD	AG (s)	AT (s)					BF	SD	AG (s)	AT (s)
3	1	30 751	30 178	**25 040**	36.108	12.725	17.67	7	1	34 514	34 346	**28 790**	66.106	14.023	20.49
	2	26 879	27 579	**21 553**	50.460	12.843	17.71		2	33 663	33 809	**26 550**	64.169	14.884	20.26
	3	24 848	24 429	**19 217**	28.189	12.657	17.70		3	29 351	29 924	**23 241**	64.769	15.437	20.27
	4	25 906	25 852	**20 934**	30.154	12.291	17.57		4	31 733	31 252	**25 028**	88.003	15.518	19.95
	5	29 185	28 471	**22 878**	27.495	12.902	17.51		5	32 606	33 376	**26 303**	71.508	15.205	20.12
5	1	32 775	33 390	**27 170**	73.194	14.167	19.26	10	1	37 760	37 452	**30 798**	73.554	17.691	23.67
	2	32 439	32 158	**24 771**	61.412	14.596	19.07		2	35 461	35 391	**28 361**	94.647	17.183	23.26
	3	26 377	26 650	**21 267**	68.665	14.234	19.29		3	32 269	32 263	**25 794**	73.184	17.086	23.18
	4	29300	29 505	**23320**	55.557	13.934	19.00		4	33 976	33 267	**27 406**	83.66	17.476	23.09
	5	31 732	31 350	**24 653**	58.744	14.046	19.09		5	35 589	35 553	**28 268**	66.106	14.023	20.49

The results of DEA and PSO are from Chakaravarthy et al. (2013). The makespan/total flow time marked in bold refers to the minimum makespan/total flow time. BF: the best total flow time over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs

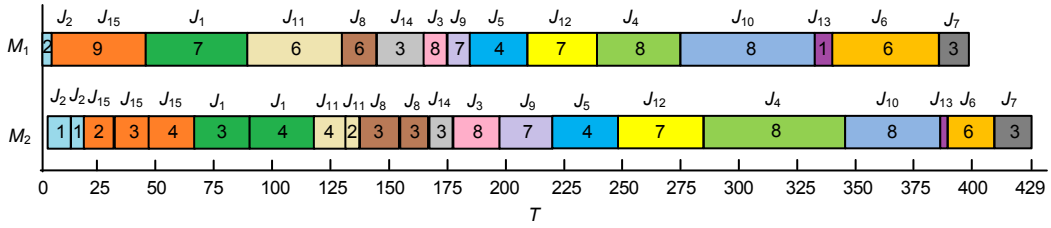

Fig. 6 Gantt chart for the 15×2 problem in the No. 1 dataset with VSS and the makespan criterion (C_{max}=429)
Each bar represents a sublot, and the number in each bar refers to the size of the sublot

5.2 Experiments on lot streaming with a bounded sublot size

To evaluate the performance of TSM on a lot streaming FSP with a bounded sublot size, we set JS_i=10, SZ_{max}=10, and SZ_{min}=3 for each job J_i in the No. 1 dataset, and applied TSM under the same parameter setting as that in Section 5.1. The results over 30 runs are shown in Tables 8 and 9.

Note that for the 10×10, 30×3, 30×7, 40×3, 40×5, 50×3, 50×5, and 50×7 problems in the No. 1 dataset, TSM could obtain the minimum makespan in each run (i.e., SD=0), which can be seen from Tables 4, 6, and 8. We conclude that besides the size of the problem. The problem itself could also affect the performance of the algorithm (e.g., when there is more than one possible processing sequence that leads to the minimum makespan in the solution space, it would be easier to find the optimal solution).

The Gantt charts for the 5×5 problem with the makespan/total flow time criterion in the No. 1 dataset when JS_i=10, SZ_{min}=3, and SZ_{max}=10 are illustrated in Figs. 7 and 8. For the problem with the makespan criterion, after the size adjustment in the third stage, the total number of transfers from M_1 to M_2, M_2 to M_3, M_3 to M_4, and M_4 to M_5 is reduced from 60 to 42 (Fig. 7), compared with the initial splitting obtained through the first stage, where each job is split into three sublots on each machine with sizes 4, 3, and 3, respectively. For the problem with the total flow time criterion, the total number of transfers is reduced from 60 to 46 (Fig. 8).

To show the efficiency of the calculation steps developed in Section 4.2.2 with the increase of job size JS_i (i=1, 2, ..., n), we conducted experiments on the 50×10 problem with makespan criterion in the No. 1 dataset under different values of JS_i (from 10 to 100 with an increment of 10) for each job. The variation

in the average CPU time (the vertical axis, in seconds) consumed by TSM for one run across different values of JS_i (the horizontal axis) under three different cases of SZ_{min}=1, 3, and 5 is illustrated in Fig. 9.

The efficient calculation shows the benefit of saving computation time over traditional calculation, and the benefit is the most remarkable in the SZ_{min}=1 case when using unit size transfer sublots. The

Table 8 Performance of the TSM on a lot streaming FSP with a bounded sublot size for the makespan criterion in the No. 1 dataset

$n\times m$	BM	SD	AG (s)	AT (s)	$n\times m$	BM	SD	AG (s)	AT (s)
5×5	345	1.28	0.0005	0.03	40×5	1854	0	1.2036	11.16
10×10	621	0	0.0202	0.16	40×7	1874	0.81	5.6471	12.20
20×10	1108	3.26	0.7190	1.06	40×10	1969	5.37	10.374	16.74
30×3	1349	0	0.2713	1.42	50×3	2290	0	0.0904	19.66
30×5	1393	2.66	0.8399	1.42	50×5	2296	0	0.8687	21.65
30×7	1511	0	0.1357	2.22	50×7	2320	0	6.4719	23.50
30×10	1542	3.55	1.9753	2.47	50×10	2393	6.19	12.5860	31.12
40×3	1843	0	0.1565	10.21					

BM: best makespan over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs

Table 9 Performance of the TSM on a lot streaming FSP with a bounded sublot size for the total flow time criterion in the No. 1 dataset

$n\times m$	BF	SD	AG (s)	AT (s)	$n\times m$	BF	SD	AG (s)	AT (s)
5×5	1076	0	0.0006	0.08	40×5	37 762	112.530	5.9827	12.03
10×10	4054	14.306	0.0500	0.20	40×7	40 494	116.482	7.1991	12.35
20×10	13 305	45.238	0.8095	1.47	40×10	43 742	164.640	9.3931	17.24
30×3	18 847	26.013	1.5563	1.98	50×3	50 157	91.777	13.5043	20.35
30×5	21 931	71.740	2.0219	2.57	50×5	56 290	225.074	15.1491	22.15
30×7	24 152	54.105	2.1536	2.93	50×7	60 289	237.065	15.6489	24.32
30×10	26 299	95.194	2.4919	3.31	50×10	64 921	224.219	21.8573	33.32
40×3	32 982	73.208	5.5635	10.56					

BF: best total flow time over 30 runs; SD: standard deviation of makespan obtained over 30 runs; AG: average CPU time consumed to reach convergence over 30 runs; AT: average CPU time consumed for one run over 30 runs

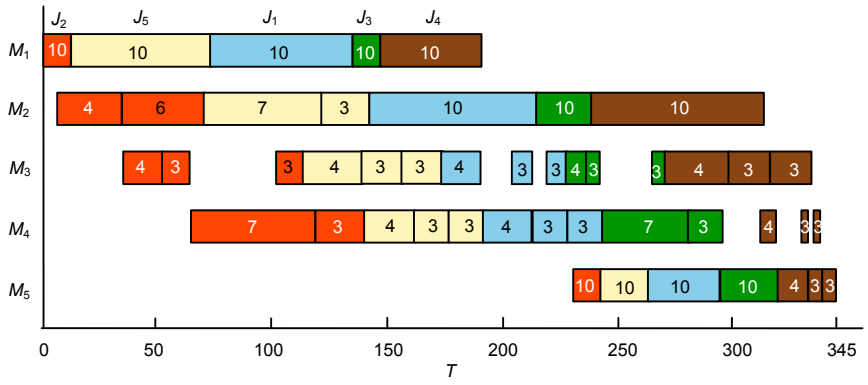

Fig. 7 Gantt chart for the 5×5 problem in the No. 1 dataset with the bounded VSS and the makespan criterion (C_{max}=345)

The number in each bar refers to the size of the sublot, and the number of bars from job J_i with the same color on machine M_l represents the number of sublots of J_i when they are transferred to M_l (i=1, 2, ..., 5 and l=1, 2, ..., 5)

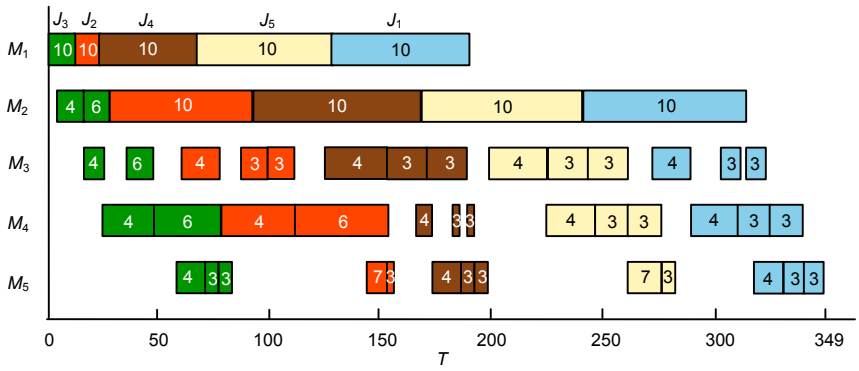

Fig. 8 Gantt chart for the 5×5 problem in the No. 1 dataset with the bounded VSS and the total flow time criterion (TFT=1076)

The number in each bar refers to the size of the sublot, and the number of bars from job J_i with the same color on machine M_l represents the number of sublots of J_i when they are transferred to M_l (i=1, 2, ..., 5 and l=1, 2, ..., 5)

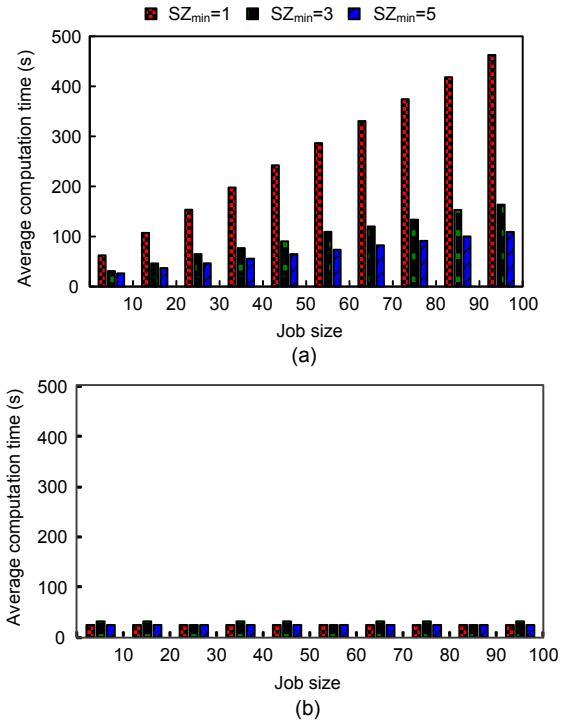

Fig. 9 Variation in the average computation time across different values of JS_i: (a) traditional calculation; (b) efficient calculation

References to color refer to the online version of this figure

computation time of traditional calculation always increases as job size increases in all three cases. With the efficient calculation, the value of job size has no influence on computation time in the SZ_{min}=1 and SZ_{min}=5 cases, and also in the SZ_{min}=3 case when JS_i=30, 60, and 90 (when JS_i is divisible by SZ_{min}). When JS_i is not divisible by SZ_{min}, the computation-time is a little longer, but does not increase as job size increases.

Note that the transfer adjustment in the third stage aims at reducing the number of transfers between machines without affecting the optimized makespan/total flow time criterion obtained from the second stage. For the best results obtained above over 30 runs for the 50×10 problem with the makespan criterion in the No. 1 dataset under SZ_{min}=1 and SZ_{min}=5 cases, the variation in the total number of transfers (the vertical axis) across different values of JS_i (the horizontal axis) is illustrated in Fig. 10. The results show that the transfer adjustment reduces the number of transfers between machines, and the reduction can be significant when the number of jobs is large. The reduction of transfers in the SZ_{min}=5 case is less significant than that in the SZ_{min}=1 case, since the initial total number of transfer sublots is small in the SZ_{min}=5 case.

The experimental results confirm the effectiveness of TSM from the following aspects: (1) Job sequencing optimization under the initial lot splitting can provide the desirable optimization solution; (2) The efficient calculation for the makespan/total flow time criterion can save a great deal of computation time in lot streaming, especially when the number of jobs (production lots) is large; (3) Transfer adjustment reduces the number of transfers between machines without affecting the optimized criterion, by adjusting the initial transfer sublots into variable size sublots (VSSs). In addition, we find that although a local

search procedure in an evolutionary algorithm can be necessary for improving the optimization ability for a large-scale and complex problem, it leads to an inevitable increase in computation time. We look forward to developing a more effective local search with greater optimization power and less computation time in our future work.

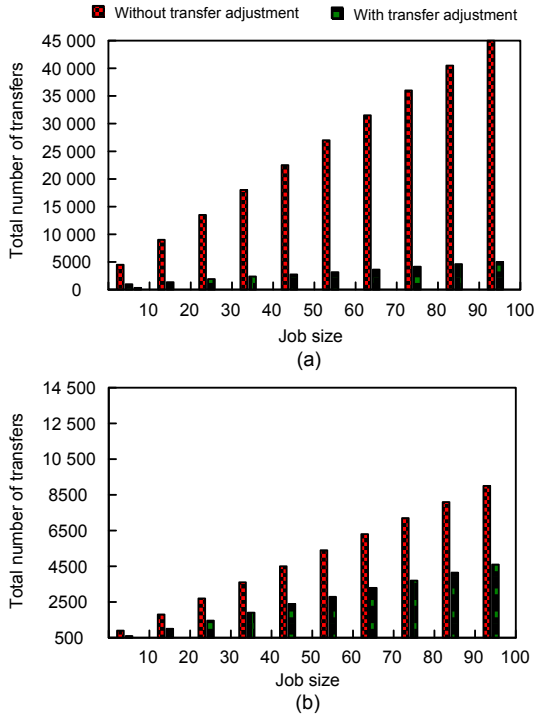

Fig. 10 Variation in the total number of transfers across different values of JS_i: (a) SZ_{min}=1; (b) SZ_{min}=5
References to color refer to the online version of this figure

6 Conclusions

In this paper, we have investigated a lot streaming FSP with bounded variable size sublots, aiming at minimizing the makespan/total flow time. The problem has been formulated and a TSM has been proposed to solve the problem efficiently. During the first stage, each job has been split according to the lower bound on sublot size, which leads to the fastest (or close to the fastest) completion for a processing sequence. Then we have developed a DE-based optimization with efficient calculation for the makespan/total flow time criterion to find the optimal processing sequence of jobs, based on the initial lot splitting in stage 1. During the third stage, to decrease the number of transfers between machines, some adjustments to sublots have been made based on the best sequence obtained through stage 2 to adjust the initial transfer sublots into VSSs under the upper bound size constraint.

Experimental results of lot streaming with unbounded and bounded sublot size cases have verified the effectiveness of the proposed TSM. Compared with the results reported before, experimental results on two- and multi-machine problems in five datasets have revealed the good performance of the proposed method in providing a better solution with less computational effort. We have provided a new method for the flow shop scheduling problem with lot streaming. The efficient calculation and transfer adjustment can be incorporated into other algorithms to save computation time and reduce the number of transfers. In future work, we will consider multi-objective and dynamic scheduling in the problem to apply real-time knowledge to achieve greater production system flexibility and enhance throughput.

Compliance with ethics guidelines

Hai-yan WANG, Fu ZHAO, Hui-min GAO, and John W. SUTHERLAND declare that they have no conflict of interest.

References

Baker KR, 1995. Lot streaming in the two-machine flow shop with setup times. *Ann Oper Res*, 57(1):1-11. https://doi.org/10.1007/BF02099687

Biskup D, Feldmann M, 2006. Lot streaming with variable sublots: an integer programming formulation. *J Oper Res Soc*, 57(3):296-303. https://doi.org/10.1057/palgrave.jors.2602016

Chakaravarthy GV, Marimuthu S, Sait AN, 2013. Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling *m*-machine flow shops with lot streaming. *J Intell Manuf*, 24(1):175-191. https://doi.org/10.1007/s10845-011-0552-2

Chakaravarthy GV, Marimuthu S, Ponnambalam SG, et al., 2014. Improved sheep flock heredity algorithm and artificial bee colony algorithm for scheduling *m*-machine flow shops lot streaming with equal size sub-lot problems. *Int J Prod Res*, 52(5):1509-1527. https://doi.org/10.1080/00207543.2013.848304

Cheng M, Mukherjee NJ, Sarin SC, 2013. A review of lot streaming. *Int J Prod Res*, 51(23-24):7023-7046. https://doi.org/10.1080/00207543.2013.774506

Davendra D, Senkerik R, Zelinka I, et al., 2014. Utilising the chaos-induced discrete self organising migrating algorithm to solve the lot-streaming flowshop scheduling

problem with setup time. *Soft Comput*, 18(4):669-681. https://doi.org/10.1007/s00500-014-1219-7

Defersha FM, Chen MY, 2010. A hybrid genetic algorithm for flowshop lot streaming with setups and variable sublots. *Int J Prod Res*, 48(6):1705-1726. https://doi.org/10.1080/00207540802660544

Defersha FM, Chen MY, 2011. A genetic algorithm for one-job m-machine flowshop lot streaming with variable sublots. *Int J Oper Res*, 10(4):458-468. https://doi.org/10.1504/IJOR.2011.039713

Han YY, Gong DW, Jin YC, et al., 2016. Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time. *Appl Soft Comput*, 42:229-245. https://doi.org/10.1016/j.asoc.2016.01.033

Kim K, Jeong IJ, 2009. Flow shop scheduling with no-wait flexible lot streaming using an adaptive genetic algorithm. *Int J Adv Manuf Technol*, 44(11-12):1181-1190. https://doi.org/10.1007/s00170-007-1236-0

Kumar S, Bagchi TP, Sriskandarajah C, 2000. Lot streaming and scheduling heuristics for *m*-machine no-wait flowshops. *Comput Ind Eng*, 38(1):149-172. https://doi.org/10.1016/S0360-8352(00)00035-8

Liu JY, 2008. Single-job lot streaming in *m*-1 two-stage hybrid flowshops. *Eur J Oper Res*, 187(3):1171-1183. https://doi.org/10.1016/j.ejor.2006.06.066

Liu SC, 2003. A heuristic method for discrete lot streaming with variable sublots in a flow shop. *Int J Adv Manuf Technol*, 22(9-10):662-668. https://doi.org/10.1007/s00170-002-1516-7

Marimuthu S, Ponnambalam SG, Jawahar N, 2008. Evolutionary algorithms for scheduling *m*-machine flow shop with lot streaming. *Robot Comput Integr Manuf*, 24(1):125-139. https://doi.org/10.1016/j.rcim.2006.06.007

Marimuthu S, Ponnambalam SG, Jawahar N, 2009. Threshold accepting and ant-colony optimization algorithms for scheduling *m*-machine flow shops with lot streaming. *J Mater Process Technol*, 209(2):1026-1041. https://doi.org/10.1016/j.jmatprotec.2008.03.013

Martin CH, 2009. A hybrid genetic algorithm/mathematical programming approach to the multi-family flowshop scheduling problem with lot streaming. *Omega*, 37(1):126-137. https://doi.org/10.1016/j.omega.2006.11.002

Mukherjee NJ, Sarin SC, Singh S, 2017. Lot streaming in the presence of learning in sublot-attached setup times and processing times. *Int J Prod Res*, 55(6):1623-1639. https://doi.org/10.1080/00207543.2016.1200760

Nejati M, Mahdavi I, Hassanzadeh R, et al., 2014. Multi-job lot streaming to minimize the weighted completion time in a hybrid flow shop scheduling problem with work shift constraint. *Int J Adv Manuf Technol*, 70(1-4):501-514. https://doi.org/10.1007/s00170-013-5265-6

Nejati M, Mahdavi I, Hassanzadeh R, et al., 2016. Lot streaming in a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint. *J Ind Prod Eng*, 33(7):459-471. https://doi.org/10.1080/21681015.2015.1126653

Onwubolu G, Davendra D, 2006. Scheduling flow shops using differential evolution algorithm. *Eur J Oper Res*, 171(2):674-692. https://doi.org/10.1016/j.ejor.2004.08.043

Pan QK, Suganthan PN, Liang JJ, et al., 2011. A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem. *Expert Syst Appl*, 38(4):3252-3259. https://doi.org/10.1016/j.eswa.2010.08.111

Reiter S, 1966. A system for managing job-shop production. *J Bus*, 39(3):371-393. https://doi.org/10.1086/294867

Sarin SC, Kalir AA, Chen M, 2008. A single-lot, unified cost-based flow shop lot-streaming problem. *Int J Prod Econ*, 113(1):413-424. https://doi.org/10.1016/j.ijpe.2007.10.002

Storn R, Price K, 1997. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. *J Glob Optim*, 11(4):341-359. https://doi.org/10.1023/a:1008202821328

Tasgetiren MF, Pan QK, Suganthan PN, et al., 2013. A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. *Comput Oper Res*, 40(7):1729-1743. https://doi.org/10.1016/j.cor.2013.01.005

Tseng CT, Liao CJ, 2008. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *Eur J Oper Res*, 191(2):360-373. https://doi.org/10.1016/j.ejor.2007.08.030

Ventura JA, Yoon SH, 2013. A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. *J Intell Manuf*, 24(6):1185-1196. https://doi.org/10.1007/s10845-012-0650-9

Wang L, Pan QK, Suganthan PN, et al., 2010. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Comput Oper Res*, 37(3):509-520. https://doi.org/10.1016/j.cor.2008.12.004