Sensors & Transducers, Vol. 155, Issue 8, August 2013, pp. 232-241

www.sensorsportal.com

Sensors & Transducers

http://www.sensorsportal.com

Genetic Algorithm-based Hybrid Methods for a Flexible Single-operation Serial-batch Scheduling Problem with Mold Constraints

Hegen Xiong, Huali Fan, Gongfa Li

School of Machine and Automation of Wuhan University of Science and Technology, No. 947, Heping Road, Qingshan District, Wuhan City, 430081,China

Tel.: +86-15697181033, fax: +86-02768862283

E-mail: xionghegen@126.com

Received: 14 July 2013 /Accepted: 12 August 2013 /Published: 20 August 2013

Abstract: Scheduling in a manufacturing system is currently a research hotspot. This paper proposes a flexible, single-operation, serial-batch scheduling problem with mold constraints that arise from injection molding production. In such a scheduling problem, three decision problems exist: product batch splitting, sequencing, and resource selection. A divide-and-conquer framework is introduced to solve the three decision problems. A hybrid algorithm combining the genetic algorithm for sequencing with the heuristic algorithm for resource selection is developed. We design an experimental case to which the corresponding simulation scheduling process is applied. We present a detailed analysis of the simulation results, which indicate that the solution framework and the hybrid algorithm are effective. *Copyright © 2013 IFSA.*

Keywords: Serial-batch scheduling, Mold constraints, Injection molding production, Genetic algorithm, Dispatching rule.

1. Introduction

The scheduling problem in a manufacturing system has been extensively studied over the past decades. Based on the processing method, scheduling problems can be classified into two types: uniprocessing and batch processing scheduling problems. In batch processing scheduling, jobs are processed in batches. This process indicates that either two or more jobs are processed simultaneously in a machine or jobs in the same family are processed continuously and serially in a machine. The first case is generally referred to as parallel batch scheduling, whereas the latter is called serial-batch scheduling (SBS). Batch processing scheduling has attracted considerable research interest for its practical engineering applications.

In 1984, Santos studied single-machine scheduling problems with batching and sequencing decisions with lead-time considerations. Thereafter, over 100 studies on batch scheduling have been conducted. A great deal of research has been performed by Kovalyov, Brucker, and Potts, who studied molds, algorithms, as well as the complexity of SBS for single [1, 2] and for parallel machines [3, 4]. The models presented were characterized by the availability of all jobs at time zero, set-up time consideration, and weighted delay-associated objective functions. The algorithms that were commonly adopted in previous studies include the dynamic programming, branch and bound, simulated annealing, taboo search, and genetic algorithms [2-6], with dynamic programming being the most widely applied. Moreover, for single-machine and multi-

Article number P_SI_1309

machine SBS, Ghosh developed a backward dynamic programming model [7], which is an improvement over dynamic programming [8], as well as a pseudo-polynomial dynamic programming method [9]. Webster *et al.* studied the complexity of SBS for parallel machines [10]. A single-machine SBS for flow time minimization was explored by Mason *et al.* [11]. Cheng also conducted a great deal of research on batch scheduling, particularly on single-machine SBS problems [6, 12, 13]. Cheng proposed several kinds of dynamic programming methods that considered pseudo-polynomial-time.

In this paper, we address a new, flexible, single-operation SBS problem with mold constraints. In this problem, multiple products each have a batch with a predetermined size, a release time, a due date, and an importance measurement (i.e., weight). The jobs for all products have only one operation (e.g., injection processing) that has to be processed on a machine by using a mold based on the product family. The number of available molds for a family job may be more than one. A job can be processed by a machine selected from a specified machine set. However, the matching degree in terms of technology, processing economics, and processing velocity for a machine within the same job-specified machine set may differ. Moreover, differences exist in terms of the colors of jobs belonging to different product families. When a machine processes a deep-color job before a light-color job, this machine has to be cleaned. For the scheduling problem, the objective is to minimize the total weighted tardiness (TWT) of all jobs.

The remainder of this paper is organized as follows: Section 2 states the proposed scheduling problem. Section 3 presents a detailed solution to such problem. A case study on the scheduling problem is described in detail in Section 4. Section 5 presents the scheduling results and corresponding analysis. Finally, the conclusion is presented in Section 6.

2. Problem Description

The problem addressed in this paper can be defined as follows: n batches of products $P = \{P_1, P_2, \cdots, P_i, \cdots, P_n\}$ are to be processed. The families and due dates of different batches are not the same. In a shop, *ma* machines $Ma = \{Ma_1, Ma_2, \cdots, Ma_{ma}\}$ and *mo* sets of molds $Mo = \{Mo_1, Mo_2, \cdots, Mo_{mo}\}$ can be used. For product P_i, we denote n_i as its batch size and w_i, co_i, p_i, r_i, and d_i as its weight, color, job processing time, release time, and due date, respectively. When a job is to be processed, a corresponding mold must be equipped to the corresponding machine. For a job of P_i, the mold can be selected from

$$Mo_i = \{Mo_{i1}, Mo_{i2}, \cdots, Mo_{ij}, \cdots, Mo_{iq_i}\},$$

where q_i denotes the number of available molds for P_i. The processing machine will be one among those in a machine set expressed as the following matrix:

$$Ma_i = \begin{bmatrix} Ma_{i1} \\ Ma_{i2} \\ \vdots \\ Ma_{ik} \\ \vdots \\ Ma_{im} \end{bmatrix} = \begin{bmatrix} Ma_{i11} & Ma_{i12} & \cdots & Ma_{i1l} & \cdots \\ Ma_{i21} & Ma_{i22} & \cdots & Ma_{i2l} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Ma_{ik1} & Ma_{ik2} & \cdots & Ma_{ikl} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Ma_{im1} & Ma_{im2} & \cdots & Ma_{iml} & \cdots \end{bmatrix} \quad (1)$$

In the machine matrix, we divide the machines into *m* grades according to their technological and economic matching degree with P_i. All machines in $Ma_{i1} = \{Ma_{i11}, Ma_{i12}, \cdots, Ma_{i1l}, \cdots\}$ have the best matching degree, whereas all machines in $Ma_{im} = \{Ma_{im1}, Ma_{im2}, \cdots, Ma_{iml}, \cdots\}$ have the worst matching degree.

To solve this scheduling problem, we initially split all product batches into several lots, after which we schedule all the lots. The scheduling process involves the sequencing of lots and the selection of the optimal machine and the optimal mold, such that the TWT of all products is minimized [14]. Let s_i denote the processing starting time of P_i; and let c_i and $\vec{t}_i$ be the processing finishing time and the weighted tardiness, respectively. The TWT $\vec{T}$ of all products can thus be derived as follows:

$$\vec{T} = \sum_{i=1,2,\cdots,n} \vec{t}_i = \sum_{i=1,2,\cdots,n} w_i \cdot \max(0, c_i - d_i) \quad (2)$$

3. Problem Solving Methods

3.1. Solving Framework

To solve the given scheduling problem, three decisions have to be made: (1) splitting of all product batches into lots; (2) optimally sequencing all lots; and (3) selecting the optimal processing machine and mold. A divide-and-conquer framework is introduced to address the aforementioned decisions (Fig. 1).

First, all batches of products were split based on a splitting rule. We then sequenced all product lots by using the genetic algorithm, where a chromosome corresponds to a lot sequence. Finally, a chromosome is decoded according to the decision rule and resource selection, while simultaneously calculating the processing time intervals calculating. The splitting result, the lot sequence, and the schedule comprise the solution to the scheduling problem.

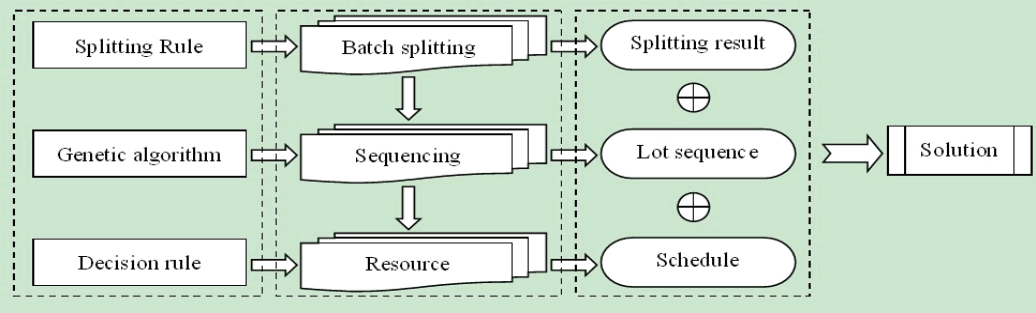

Fig. 1. Framework for solving the scheduling problem.

3.2. Product Batch Splitting

As previously mentioned, when a job is to be processed on a machine, a mold must be equipped beforehand. Thus, this procedure requires a relatively long setup time. When a machine processes a job belonging to one product family that is then immediately preceded by a job from another product family, the disassembly of the previous mold and the assembly of the subsequent one would cost a great deal of time. To reduce setup times while improving processing efficiency by fully utilizing mold resources, we split the product batches based on the following rules:

For P_i, if only one mold is available, i.e., $q_i=1$, then the batch of product P_i need not be split, and the products should consecutively be processed on a machine with the same mold.

For P_i, if more than one mold is available, i.e., $q_i > 1$, the batch of product P_i can possibly be split into several lots for concurrent processing on different machines with different molds. Thus, we split the batch according to the equipartition principle into q_i lots that are denoted as $P_i^{x_i}$, $x_i = 1,2,\cdots,q_i$. Let $b_i^{x_i}$ denote the size of $P_i^{x_i}$. Then,

$$b_i^{x_i} = \begin{cases} \lfloor n_i/q_i \rfloor & x_i = 1,2,\cdots;q_i-1 \\ n_i-(q_i-1)\cdot\lfloor n_i/q_i \rfloor & x_i = q_i \end{cases}, \quad (3)$$

where $\lfloor x \rfloor$ is the floor function about x.

Notably, given that splitting precedes resource selection, we split a product batch into small lots, such that for product P_i, the number of lots equals the static available quantity of mold q_i. Thus, the decision space for subsequent scheduling will be enlarged. However, considering dynamic factors such as mold maintenance and breakdown when scheduling, the dynamic available quantity of a mold may be only one. Thus, all lots may be assigned to the same mold and the same machine for consecutive processing.

3.3 Genetic Algorithm for Lot Sequencing

To sequence all product lots, we adopt the genetic algorithm based on symbol encoding, where every chromosome represents a sequencing scheme. By scheduling the sequencing scheme, we can make the processing resource decision as well as calculate the processing times and TWT.

3.3.1. Chromosome Encoding

By splitting all product batches, we can derive the number of product lots $P_i^{x_i}$, where $i = 1,2,\cdots,n$, and $x_i = 1,2,\cdots,q_i$. Let N_{sub} denote the number of lots of all product batches. We thus obtain

$$N_{sub} = \sum_{i=1,2,\cdots,n} q_i \quad (4)$$

Every legal chromosome is an orderly symbol string comprising $P_i^{x_i}$ ($i = 1,2,\cdots,n; x_i = 1,2,\cdots q_i$) with length N_{sub}. For example, three product batches exist: P_1, P_2, and P_3, each of which are split into three lots: P_1^1, P_1^2, and P_1^3; P_2^1, P_2^2, and P_2^3; and P_3^1, P_3^2, and P_3^3, respectively. Fig. 2 illustrates a case of a legal chromosome.

P_1^2	P_3^3	P_2^1	P_1^3	P_2^2	P_3^2	P_2^3	P_3^1	P_1^1

Fig. 2. Case of a legal chromosome.

3.3.2. Selection Operator

In this paper, we use tournament selection as the selection operator. Commonly used methods include binary and ternary tournament selection. Evidently, a greater number of chromosomes participating in a tournament results in the higher homoplasy and higher average fitness of such chromosomes. Thus,

the evolutionary procedure will facilitate rapid convergence. However, this procedure may cause population diversity to worsen and catastrophe to occur. To achieve a tradeoff between convergence rate and population diversity, we found that an appropriate number of individuals to be considered in tournament selection may be derived according to the population size as follows:

$$\Pi = \begin{cases} 2 & if: popu \le 20 \\ 4 & f: popu \ge 40 \\ 3 & else \end{cases}, \quad (5)$$

where $popu$ denotes the population size. For example, let $popu = 20$, and let binary tournament selection be adopted. For each instance that two chromosomes are drawn through to simple random sampling with replacement, the fitness of each chromosome is evaluated, and the individual with the higher fitness is retained for the subsequent genetic operation. This process is repeated until 20 chromosomes are selected.

3.3.3. Crossover Operator

In the proposed genetic algorithm, a chromosome is a symbol string comprising the representation of all product lots. Thus, a legal chromosome must satisfy the completeness and exclusiveness criteria. Part mapping crossover is adopted to ensure that the offspring are legal. We take a sample that includes three product batches: P_1, P_2, and P_3, each of which are split into three lots. Fig. 3 (a) represents two parent individuals. After applying part mapping crossover, we derive the offspring individuals as shown in Fig. 3(b).

parent 1	P_1^2	P_3^3	P_2^1	P_1^3	P_2^2	P_2^3	P_3^2	P_3^1	P_1^1
parent 2	P_2^3	P_1^1	P_1^3	P_3^2	P_3^1	P_2^2	P_1^2	P_3^3	P_2^1

(a) parent chromosomes.

offspring 1	P_2^1	P_3^3	P_1^3	P_3^2	P_3^1	P_2^2	P_1^2	P_2^3	P_1^1
offspring 2	P_3^1	P_1^1	P_2^1	P_1^3	P_2^2	P_2^3	P_3^2	P_3^3	P_1^2

(b) offspring chromosomes.

Fig. 3. Example of part mapping crossover.

3.3.4. Mutation Operator

Mutation is implemented in the manner similar to a basic swap operation, i.e., a chromosome is randomly selected while two loci are randomly generated. Genes located in the two loci are then mutually exchanged. Taking the offspring chromosome 1 in Fig. 3 as an example, let the loci be 3 and 6. The resulting mutation is then shown in Fig. 4.

P_2^1	P_3^3	P_2^2	P_3^2	P_3^1	P_1^3	P_1^2	P_2^3	P_1^1

Fig. 4. An example of mutation.

3.4. Resource Decision

In Subsection 3.3, we discussed that sequencing only provides the dispatching order for all product lots. Hence, we should schedule all lots so that the most suitable resources, i.e., processing machine and mold, could be assigned and processing time intervals could be calculated. Considering the technological and economic matching degree of machines with respect to the product family and TWT objective, we introduce a two-step combination rule: Earliest Tardiness Time (ETT) + Best Matching Degree (BMD), which operates as follows: when a product lot is to be scheduled, a processing resource combination (a machine and a mold) with the least weighted tardiness will be assigned for the lot. If more than one processing resource combination with the same least weighted tardiness exists, then the one with the highest matching degree will be selected. On the other hand, if more than one processing resource combination with the same least weighted tardiness and the same highest matching degree exists, any one can be selected at random. When deciding on resources, we must calculate the weighted tardiness of the lot. Thus, the setup, starting, and finishing times are introduced in the following subsections.

3.4.1. Setup Time

Molds are indispensable technological equipment for injection processing. Before a job is processed, a mold must be set up on a machine. Setup time, which is usually exceeds the processing time of a single job, is this required. Given that a mold can be assembled before a job arrives and that setup times are independent of the number of jobs processed on the mold, setup time should not be integrated with processing time. The composition of setup times can be classified into four cases: a machine processes a product lot without any forerunner, because of which assembling time occurs; a machine successively processes two lots of products belonging to the same family, because of which no setup time occurs; a machine successively processes two lots of products belonging to different families and the color of the former product is the same as or lighter than that of the latter, because of which the disassembling time of the mold for the former product lot and the assembling time of the mold for the subsequent product occur; and finally, the color of the former product is darker than that of the latter, because of which the setup time will include the disassembling,

assembling, and washing times. Let the product lot to be scheduled now be P_i^*. The selected machine is Ma_i^* ($Ma_i^* \in Ma_i$); the selected mold is Mo_i^* ($Mo_i^* \in Mo_i$); and the previous product lot and its mold are P_{i-1}^* and Mo_{i-1}^*, respectively. We then provide the following conventions:

If two molds are the same, $Mo_i^* = Mo_{i-1}^*$;

If the color of product P_i is lighter than that of product P_{i-1}, $co_i < co_{i-1}$; and if the color of product P_i is the same as or darker than that of product P_{i-1}, $co_i \geq co_{i-1}$.

For product P_i, we denote the assembling, disassembling, and washing times as t_{i-asm}, t_{i-dasm}, and t_{i-wash}, respectively. The setup time, denoted as $t_{i-setup}$, can then be presented as

$$t_{i\_aux} = \begin{cases} t_{i-asm} & if: P_{i-1}^* = empty \\ 0 & if: Mo_i^* = Mo_{i-1}^* \quad and \quad co_i \geq co_{i-1} \\ t_{i-wash} & if: Mo_i^* = Mo_{i-1}^* \quad and \quad co_i < co_{i-1} \\ t_{i-asm} + t_{i-dasm} & if: Mo_i^* \neq Mo_{i-1}^* \quad and \quad co_i \geq co_{i-1} \\ t_{i-asm} + t_{i-dasm} + t_{i-wash} & if: Mo_i^* \neq Mo_{i-1}^* \quad and \quad co_i < co_{i-1} \end{cases} \tag{6}$$

3.4.2. Starting Time

In the proposed scheduling problem, for each product lot to be processed, the following conditions must be satisfied: (1) the product lot reaches the corresponding machine; (2) at least one corresponding mold is available at each moment; (3) at least one corresponding machine is available at each moment; and (4) the mold is assembled on the machine. With respect to product lot P_i^*, the available time of the mold is denoted as $t_{Mo_i^*-avail}$. Thus, the starting time, denoted as s_i^*, can be computed by

$$s_i^* = \max\{r_i, t_{i-asm} + \max(t_{Mo_i^*-avail}, c_{i-1} + t_{i-wash} + t_{i-dasm})\} \tag{7}$$

which is illustrated in Fig. 5.

3.4.3. Processing Times of a Product Lot

In injection molding production, a mold usually has a multi-cavity feature, such that it can simultaneously process more than one job according the number of cavities. Let ω_i^* denote the number of cavities of Mo_i^*, such that ω_i^* identical jobs can be processed in an injection technological cycle. The processing time for these ω_i^* jobs is equal to that for processing a single job, i.e., p_i. As previously mentioned, the processing velocity for a machine within the same job-specified machine set may be different. Thus, we adopt the velocity factor v_i^* for Ma_i^* to measure processing velocity. For the product lot P_i^*, the size of which is b_i^*, the processing time bp^{i*} is calculated by

$$bp_i^* = \left\lceil \frac{b_i^*}{\omega_i^*} \right\rceil \cdot p_i / v_i^*, \tag{8}$$

where $\lceil x \rceil$ is the ceiling function about x.

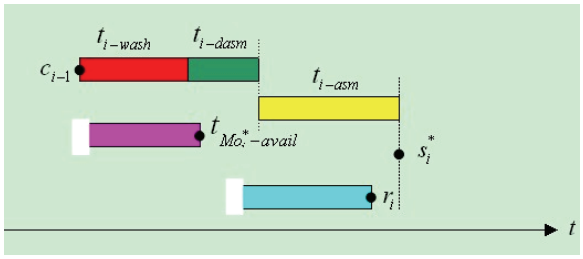

Fig. 5. Illustration of starting time computation.

3.4.4. Finishing Time and Weighted Tardiness

For a product lot P_i^*, the finishing time c_i^* can be calculated based on the starting and processing times by using the following formula:

$$\begin{aligned} c_i^* &= s_i^* + bp_i^* \\ &= \max\{r_i, t_{i-asm} + \left\lceil \frac{b_i^*}{\omega_i^*} \right\rceil \cdot p_i / v_i^* \\ &+ \max(t_{Mo_i^*-avail}, c_{i-1} + t_{i-wash} + t_{i-dasm})\} \end{aligned} \tag{9}$$

and the weighted tardiness $\vec{t}_i^*$ can be computed by

$$\vec{t}_i^* = w_i \cdot \max(0, c_i^* - d_i) \tag{10}$$

3.4.5. Resource Selection

After generating a processing sequence for the product lots, the most suitable resources should be assigned for each lot. According to the resource

decision combination rule EET+BMD, we can derive the resource selection algorithm as procedure 1.

Procedure 1 [Procedure for resource selection]

$\vec{t}_{\text{benchmark}}$: benchmark of weighted tardiness

$Ma_i^{selected}$: selected machine

$Mo_i^{selected}$: selected mold

Begin

$\vec{t}_{\text{benchmark}} = +\infty$

For k=1 to k=m

For l=1 to l= y_k

$Ma_i^* := Ma_{ikl}$

For j=1 to q_i

$Mo_i^* := Mo_{ij}$

$\vec{t}_i^* = w_i \cdot \max(0, c_i^* - d_i)$

If ($\vec{t}_i^* < \vec{t}_{\text{benchmark}}$)

$\vec{t}_{\text{benchmark}} := \vec{t}_i^*$

$Ma_i^{selected} := Ma_i^*$

$Mo_i^{selected} := Mo_i^*$

End if

End for

End for

End for

End

Referring to the definition of the technological and economic matching degree of machines, we can easily determine that matching degree can be measured by using the loop variable k. A smaller value of k denotes a higher matching degree. In the resource selection procedure, loop variable k progressively increases as the loop runs, such that resources with a higher matching degree will be selected automatically.

4. Case Study

4.1. Case Information

In this section, we discuss a case to illustrate the scheduling problem and the proposed solving methods. In the proposed case, 15 batches of products are to be processed in a shop with eight sets of machines and 13 sets of molds. The information on product batches is presented in Table 1. The details on the machines and molds are presented in Table 2 and Table 3, respectively. The washing times necessary for a machine to process two different colors of products successively are given in Table 4. Sometimes, a color taboo exists for when two products are processed successively in the same machine. For example, if the color of the previously processed product is black, the immediate successor should not be a white product. To implement such a taboo, we set an appropriate duration for the washing times, e.g., 1000 units of time, as shown in Table 4.

Table1. Information on product batches.

Basic information								Resource information	
ID	**Color**	**Batch size**	**Weight**	**Processing time**	**Release time**	**Due date**		**Machine set**	**Mold set**
P1	white	500	1	2	0	600		$\begin{bmatrix} Ma_1 & Ma_2 \\ Ma_3 & Ma_4 \end{bmatrix}$	$\{Mo_1, Mo_2\}$
P2		500			0	800			
P3	yellow	1000	2	3	10	600		$\begin{bmatrix} Ma_1 & Ma_2 \\ Ma_5 & Ma_6 \end{bmatrix}$	$\{Mo_3\}$
P4	blue	1000	1		5	1300		$[Ma_3 \quad Ma_4]$	$\{Mo_4\}$
P5		1000			5	1800			
P6	red	800	3	4	20	1000		$\begin{bmatrix} Ma_5 & Ma_6 \\ Ma_7 & Ma_8 \end{bmatrix}$	$\{Mo_5\}$
P7	black	500	2	3	10	800		$[Ma_1 \quad Ma_2]$	$\{Mo_6\}$
P8		500			10	900			
P9		500			10	1000			
P10	white	2000	2	6	50	2600		$\begin{bmatrix} Ma_1 & Ma_2 \\ Ma_5 & Ma_6 \end{bmatrix}$	$\{Mo_7, Mo_8\}$
P11	white	1200	1	4	0	1200		$\begin{bmatrix} Ma_7 & Ma_8 \\ Ma_3 & Ma_4 \end{bmatrix}$	$\{Mo_9\}$
P12	red	800	1	3	0	1500		$[Ma_7 \quad Ma_8]$	$\{Mo_{10}\}$
P13		800			0	2000			
P14	blue	1500	1	2	30	1000		$\begin{bmatrix} Ma_5 & Ma_6 \\ Ma_1 & Ma_2 \end{bmatrix}$	$\{Mo_{11}, Mo_{12}\}$
P15	yellow	600	2	6	20	1200		$[Ma_3 \quad Ma_4]$	$\{Mo_{13}\}$

Table 2. Information on machines.

ID	Initial available time	Velocity factor
Ma_1	10	0.8
Ma_2	0	1.0
Ma_3	0	1.2
Ma_4	50	1.0
Ma_5	20	1.0
Ma_6	10	1.5
Ma_7	100	0.8
Ma_8	50	1.0

Table 3. Information on molds.

ID	Number of cavities	Initial available time	Assembling times	Disassembling times
Mo_1	4	0	50	20
Mo_2	4	10	50	20
Mo_3	4	15	40	10
Mo_4	4	8	60	20
Mo_5	4	0	80	20
Mo_6	4	20	50	20
Mo_7	4	12	70	30
Mo_8	4	18	70	30
Mo_9	4	0	30	10
Mo_{10}	4	30	35	10
Mo_{11}	4	0	50	20
Mo_{12}	4	0	50	20
Mo_{13}	4	20	45	20

Table 4. Washing time for color switching.

Color of previous product	Color of subsequent product				
	white	yellow	red	blue	black
white	0	0	0	0	0
yellow	10	0	0	0	0
red	20	10	0	0	0
blue	30	20	10	0	0
black	*1000*	30	20	10	0

4.2. Product Batch Splitting and Chromosome Encoding

According to the proposed rules for the splitting of product batches, P1, P2, P10 and P14 can each be split into two product lots, such that all batches would initially have two sets of available molds. However, other product batches cannot be split, given that each of them has only one set of mold is available. Thus, we have 19 product lots to be scheduled. A legal chromosome with 19 genes is illustrated in Fig. 6.

P_1^2	P_3^3	P_2^1	P_1^3	P_2^2	P_3^2	P_2^3	P_3^1	P_1^1

Fig. 6. Example of a legal chromosome.

5. Simulation Scheduling and the Analysis of the Results

5.1. Simulation Scheduling and the Results

To analyze and evaluate the proposed scheduling methods, we develop a simulation scheduling system using Java programming language. We conduct 10 runs of simulation scheduling for the above case using the system. In our simulation scheduling runs, the operation parameters of the genetic algorithm are set as follows: population size is 20, crossover probability is 0.8, mutation probability is 0.1, and the maximum iterations are 200. The elitist selection strategy is adopted in the genetic algorithm. Results of the 10 runs of simulation scheduling are shown in Table 5. In these results, the optimal fitness, i.e., the minimal TWT, is 937.6, and the corresponding solution is illustrated in Fig. 7.

Table 5. Results of 10 runs of simulation scheduling based on the genetic algorithm.

Number of runs	Results		
	Best fitness	Generationth for the best fitness	Run times(s)
1	937.6	29	0.422
2	937.6	29	0.438
3	937.6	26	0.375
4	937.6	43	0.438
5	937.6	18	0.344
6	937.6	37	0.438
7	1356.3	69	0.594
8	937.6	14	0.344
9	1356.3	59	0.516
10	937.6	18	0.422

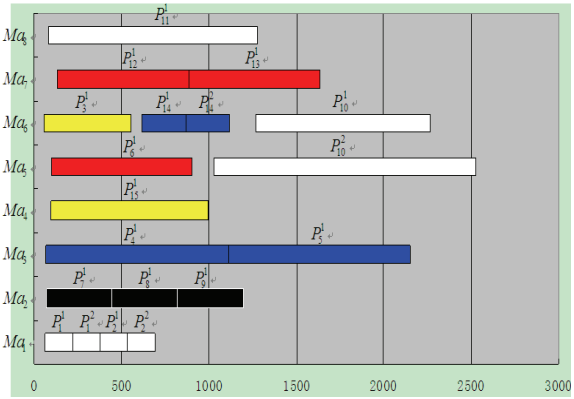

Fig. 7. Gantt chart of the best scheduling solution based on the genetic algorithm.

In our simulation scheduling, dispatching rules, with which we aim at evaluating the proposed genetic algorithm, are also adopted in product lot sequencing. Based on the features of the proposed scheduling problem, we propose two dispatching rules: Shortest

Estimate of Batch Processing Time (SEBPT) and Least Estimate of Batch Slack (LEBSL). For a product lot $P_i^{x_i}$, the estimates of batch processing time and of batch slack, respectively denoted by $b\hat{p}_i^{x_i}$ and $b\hat{s}_i^{x_i}$, are defined as follows.

Estimate of batch processing time: For any of a product lot, multiple processing machines are available, such that the processing times are machine-dependent. We can only derive an estimate of the batch processing times before the resource decision is made. By ignoring the velocity factor, $b\hat{p}_i^{x_i}$ can be derived as

$$b\hat{p}_i^{*} = \left\lceil \frac{b_i^{*}}{\omega_i^{*}} \right\rceil \cdot p_i \tag{11}$$

Estimate of batch slack: Given that the starting and processing times of a product lot are dependent on the selected machine and the selected mold, slack cannot be calculated accurately before the resource decision is made. We thus provide the estimate of batch slack as follows:

$$b\hat{s}_i^{x_i} = d_i - (r_i + b\hat{p}_i^{x_i}) \tag{12}$$

With respect to SEBPT, the priority of a product lot is given as $Z_i^{x_i} = b\hat{p}_i^{x_i}$, and with respect to LEBSL, the priority is given as $Z_i^{x_i} = b\hat{s}_i^{x_i}$. The product lot with the least $Z_i^{x_i}$ is chosen for loading.

Table 6 presents the detailed results of the rule-based simulation scheduling. The Table 6 shows that the TWTs for SEBPT and LEBSL are 2285 and 4343.8, respectively.

Table 6. Results of rule-based simulation scheduling.

Product lots	Results based on SBPT (TWT=2285.0)				Results based on LEBSL (TWT=4343.8)			
	Machine	Mold	Starting time	Finishing time	Machine	Mold	Starting time	Finishing time
P_1^1	Ma_1	Mo_1	60.0	217.5	Ma_2	Mo_1	50.0	176.0
P_1^2	Ma_1	Mo_1	217.5	375.0	Ma_2	Mo_1	176.0	302.0
P_2^1	Ma_1	Mo_1	375.0	532.5	Ma_1	Mo_1	1218.8	1376.3
P_2^2	Ma_1	Mo_1	532.5	690.0	Ma_1	Mo_1	1376.3	1533.8
P_3^1	Mo_6	Mo_3	55.0	555.0	Ma_6	Mo_3	55.0	555.0
P_4^1	Ma_3	Mo_4	895.0	1936.7	Ma_4	Mo_4	110.0	1360.0
P_5^1	Ma_3	Mo_4	1936.7	2978.3	Ma_4	Mo_4	1360.0	2610.0
P_6^1	Ma_8	Mo_5	130.0	930.0	Ma_5	Mo_5	100.0	900.0
P_7^1	Ma_2	Mo_6	70.0	445.0	Ma_1	Mo_6	70.0	538.8
P_8^1	Ma_2	Mo_6	445.0	820.0	Ma_2	Mo_6	588.8	963.8
P_9^1	Ma_2	Mo_6	820.0	1195.0	Ma_2	Mo_6	963.8	1338.8
P_{10}^1	Ma_5	Mo_7	972.0	2472.0	Ma_6	Mo_8	1015.7	2015.7
P_{10}^2	Ma_6	Mo_8	645.0	1645.0	Ma_5	Mo_7	1030.0	2530.0
P_{11}^1	Ma_4	Mo_9	80.0	1280.0	Ma_3	Mo_9	30.0	1030.0
P_{12}^1	Ma_7	Mo_{10}	135.0	885.0	Ma_7	Mo_{10}	135.0	885.0
P_{13}^1	Ma_7	Mo_{10}	885.0	1635.0	Ma_7	Mo_{10}	885.0	1635.0
P_{14}^1	Ma_5	Mo_{11}	70.0	446.0	Ma_1	Mo_{12}	618.8	1088.8
P_{14}^2	Ma_5	Mo_{11}	446.0	822.0	Ma_6	Mo_{11}	615.0	865.7
P_{15}^1	Ma_3	Mo_{13}	65.0	815.0	Ma_3	Mo_{13}	1085.0	1835.0

5.2. Analysis of Results

5.2.1. Comparative Analysis of the Results of Genetic Algorithm-based and the Rule-based Scheduling

The minimal TWT for genetic algorithm-based scheduling is 937.6, and those for SEBPT and LEBSL are 2285 and 4343.8, respectively. Genetic algorithm-based scheduling evidently outperforms rule-based scheduling. An effective dispatching rule should generally reflect the effects of the main factors on scheduling. However, in the proposed scheduling problem, several factors affect the scheduling results including product batch information, multiple machines with different technological and economic matching degree, multiple molds, and sequence-dependent washing times. Thus, developing a rule with single-valued priority including all influencing factors is difficult. Thus, achieving a satisfactory scheduling result by using rule-based sequencing for the scheduling problem proposed in this paper is difficult.

5.2.2. Analysis of the Results of Genetic Algorithm-based Scheduling

Table 5 shows that when population size is 20 and the maximum iterations are 200, eight out of 10 runs of simulation scheduling achieve the best fitness, i.e., 937.6 and two runs achieve a fitness of 1356.3. We conduct another scheduling test with population size 50 and maximum iterations of 200, in which all 20 runs achieved the best fitness at 937.6. The results indicate that the genetic algorithm-based solving methods proposed in this paper have good stability.

Fig. 7 illustrates the best scheduling solution through a Gantt chart. Based on Fig. 7, the detailed analyses are given as follows:

In the case with mold assembling times, the starting time for the first product lot on a machine lags behind the initial available time of the machine. For example, the initial available time of Ma_1 is 10. However, the assembling time of Mo_1, which is assigned to the first product lot P_1^1, is 50. Thus, the starting time of P_1^1 is 60 (10 plus 50).

On Ma_5, the red product lot P_6^1 is processed immediately before the white product lot P_{10}^2. This order is unreasonable because red is darker than white, thus requiring washing times. However, we note that no tardiness occurred with the two product lots in this processing sequence. However, if we switch the sequence, P_6^1 will be delayed. Furthermore, given that the weight of P_6^1 is relatively large (w_i =3), a large weighted tardiness will be incurred for P_6^1. The same condition also appears on Ma_6 (P_{14}^2 precedes P_{10}^1).

As mentioned in Subsection 3.2, splitting a product batch into small lots aims at enlarging the decision space for scheduling. If an idle mold is available for a product lot at its scheduling moment, parallel processing of the same family of product lots may occur. Otherwise, different lots belonging to the same product batch will be scheduled to be processed consecutively on the same machine to minimize setup times. For example, product batch P_1 is split into two lots, P_1^1 and P_1^2. However, these lots are assigned to the same machine (Ma_1) and processed consecutively. The same condition also occurs for P_2 and P_{14}. With respect to P_1 and P_2, such results not only reduce setup times but also ensure their due date (i.e., tardiness is zero). For P_{14}, tardiness occurs for product lot P_{14}^2. However, to prevent the delay of other product lots, the tardiness of P_{14}^2 will be larger if we assign it to another machine. Fig. 7 shows that parallel processing occurs for product batch P_{10}, which is split into two lots and assigned to different machines (Ma_5 and Ma_6) to processed using different equipped molds (Mo_7 and Mo_8). Such an arrangement ensures P_{10}'s due date. Otherwise, if P_{10} is not split, tardiness will be inevitable.

P_3^1 is arranged for processing on machine Ma_6. This arrangement seems unreasonable because the matching degree of Ma_6 is lower than those of Ma_1 and Ma_2, which also belong to the processing machine set of P_3^1. However, Ma_6 has the largest velocity factor (i.e., 1.5), such that the due date of P_3^1 can be ensured. If P_3^1 is assigned to Ma_1 or Ma_2, the respective velocity factors of which are 0.8 and 1.0, tardiness will inevitably occur for P_3^1.

6. Conclusions

This paper addressed a flexible, single-operation SBS problem with mold constraints arising from injection molding production. To solve this problem, we propose solving methods including genetic algorithm-based sequencing and rule-based resource decision. A case with 15 product batches is studied by conducting simulation scheduling. In the case study, we also propose two rules, SEBPT and

LEBSL, which are adopted during product lot sequencing. The results and the analysis show that the genetic algorithm-based sequencing method has good stability and evidently outperforms the rule-based sequencing method.

Acknowledgements

This research was support by Research Foundation of Education Bureau of Hubei Province, China (Grant No. D20121102). The authors are grateful to the anonymous reviewers for their valuable suggestions and comments.

References

[1]. S. Albers, P. Brucker, The complexity of one-machine batching problem, *Discrete Applied Mathematics*, Vol. 47, Issue 2, 1993, pp. 87-107.

[2]. P. Brucker, M. Y. Kovalyov, Single machine batch scheduling to minimize the weighted number of late jobs, *Mathematical Methods of Operations Research*, Vol. 43, Issue 1, 1995, pp. 1-8.

[3]. M. Y. Kovalyov, Y. M. Shafransky, Batch scheduling with deadlines on parallel machines: An NP-hard case, *Information Processing Letters*, Vol. 64, Issue 2, 1997, pp. 69–74.

[4]. P. Brucker, M. Y. Kovalyov, Y. M. Shafransky, F. Werner, Batch scheduling with deadlines on parallel machines, *Annals of Operations Research*, Vol. 83, Issue 0, 1995, pp. 23–40.

[5]. C. N. Potts, V. A. Strusevich, T. Tautenhahn, Scheduling batches with simultaneous job processing for two-machine shop problems, *Journal of Scheduling*, Vol. 4, Issue 1, 2001, pp. 25–51.

[6]. T. C. E. Cheng, M. Y. Kovalyov, Single machine batch scheduling with sequential job processing, *IIE Transactions*, Vol. 33, Issue 5, 2001, pp. 413-420.

[7]. J. B. Ghosh, Batch scheduling to minimize total completion time, *Operations Research Letters*, Vol. 16, Issue 5, 1994, pp. 271–275.

[8]. J. B. Ghosh, Gupta J. N. D., Batch scheduling to minimize maximum lateness, *Operations Research Letters*, 1997, 21, pp. 77–80.

[9]. E. Erel, J. B. Ghosh, Batch scheduling to minimize the weighted number of tardy jobs, *Computers & Industrial Engineering*, Vol. 53, Issue 3, 2007, pp. 394–400.

[10]. S. T. Webster, The complexity of scheduling job families about a common due date, *Operations Research Letters,* Vol. 20, Issue 2, 1997, pp. 65-74.

[11]. A. J. Mason, E. J. Anderson, Minimizing flow time on a single machine with job classes and setup times, *Naval Research Logistics*, Vol. 38, Issue 3, 1991, pp. 333–350.

[12]. T. C. E. Cheng, Z. L. Chen, C. Oguz, One-machine batching and sequencing of multiple-type items, *Computers & Operations Research*, Vol. 21, Issue 7, 1994, pp. 717–721.

[13]. L. L. Liu, C. T. Ng, T. C. E. Cheng, Scheduling jobs with agreeable processing times and due dates on a single batch processing machine, *Theoretical Computer Science*, Vol. 374, Issue 1-3, 2007, pp. 159–169.

[14]. H. A. J. Crauwels, C. N. Potts, L. N. V. Wassenhove, Local search heuristics for single-machine scheduling with batching to minimize the number of late jobs, *European Journal of Operational Research*, Vol. 90, Issue 2, 1996, pp. 200–213.