Expert Systems with Applications 39 (2012) 1306–1314

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A genetic algorithm for scheduling dual flow shops

Chie-Wun Chiou *, Wen-Min Chen, Chin-Min Liu, Muh-Cherng Wu

Department of Industrial Engineering and Management, National Chiao Tung University, Hsin-Chu 300, Taiwan

ARTICLE INFO

Keywords:
Scheduling
Dual flow shop
Setup time
Due date
Genetic algorithm (GA)

ABSTRACT

This study examines a dual-flow shop-scheduling problem that allows cross-shop processing. The scheduling objective is to minimize the coefficient of variation of slack time (*lateness*), where the slack time (ST) of a job denotes the difference between its due date and total completion time. This scheduling problem involves two decisions: job route assignment (assigning jobs to shops) and job sequencing. This study develops a genetic algorithm (GA) embedded with the earliest due date (*EDD*) dispatching rule for making these decisions. Numerical experiments with the GA algorithm indicate that the performance of adopting a cross-shop production policy may significantly outperform that of adopting a single-shop production policy. This is particularly true when the two flow shops are asymmetrically designed.

This study develops a grouping heuristic algorithm to reduce setup time and due-date-based demand simultaneously. This study uses the proposed genetic algorithm (GA) to prove that the grouping heuristic algorithm performs well. Obtaining an approximate optimal solution makes it possible to decide the route assignment of jobs and the job sequencing of machines.

1. Introduction

Some manufacturing companies must build two plants to fulfill customer demand. This dual-plants strategy arises from two reasons: rapid capacity expansion and capacity sharing mechanisms. In the case of rapid capacity expansion, it is often more difficult to acquire land than equipment. Therefore, many companies will initially build a space large enough for two plants, and then gradually purchase equipment based on market demand.

For capacity sharing reasons, the dual-plants strategy can adopt a *cross-plant* production policy in which two plants mutually support each other in capacity. This policy increases single plant capacity because it allows one plant to utilize its equipment capacity fully by filling orders from the other plant.

The *cross-plant* production policy involves two major sequencing decisions: (1) route assignment—how to allocate jobs to each plant, and (2) job sequencing within a plant, as well as how to sequence allocated jobs for each plant. Most studies on dual-plant scheduling are developed under a *route assignment* assumption (Toba, Izumi, Hatada, & Chikushima, 2005; Wu & Chang, 2007; Wu, Chen, & Shih, 2009; Wu, Shih, & Chen, 2009).

This paper considers the problem of scheduling a dual-flow shop that allows cross-flow shop processing. Each flow shop has three stages that can process any jobs from the previous stage of a dual flow shop. Each stage has two machines capable of processing one job at a time. Each stage in the two flow shops is functionally comprehensive. That is, a job can be completely processed in one stage in either flow shop. Fig. 1 shows that all jobs follow the same sequential processing route: Stage 1, Stage 2, and Stage 3. The purpose of cross-shop production is to increase the total throughput of both flow shops and reduce the average job cycle time.

This study presents a genetic algorithm (GA) embedded with the earliest due date (EDD) dispatching rule (a GA-EDD algorithm) for scheduling dual flow shops. All jobs have two sequencing decisions to be made: (1) route assignment (assigning jobs to stages), and (2) job sequencing within each stage. This study uses the *single*-shop production policy as a benchmark to determine the effectiveness of the proposed dual flow shop algorithm. Single-shop production means that each job can be processed in only one plant (a flow shop). Numerical experiments show that *GA-EDD* for a dual-flow shop is much better than the production of two single shops.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 explains the research problem in detail. Section 4 describes how to compute the coefficient of slack time variation for a job sequence. Section 5 presents the solution architecture of the proposed algorithms. Section 6 reports numerical experiments, while Section 7 provides concluding remarks.

2. Relevant literature

Previous studies on the dual-flow shop-scheduling problem generally fall into two groups: product-level and operational-level. In

* Corresponding author.
E-mail address: cw_chiou@yahoo.com.tw (C.-W. Chiou).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.008

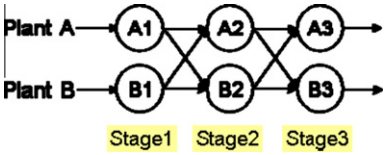

Fig. 1. The production flow of a typical dual-flow shop.

the product-level group, a product can be processed only within single plant. In the operational-level group, each plant can perform a group of operations, distributing its operations among different plants.

Most studies in the product level group-*prohibiting any cross-plant routes* category assume that each plant manufactures products separately, which prohibits cross-plant production. Wu, Erkoc, and Karabuk (2005) provided a comprehensive survey of this topic, and recent studies have extended their findings (Chiang, Guo, Chen, Cheng, & Chen, 2007; Lee, Chung, Lee, & Kang, 2006). Most of these studies use the linear programming (LP) technique to solve the dual flow shop scheduling problem. These production-level studies consider one factor: *job sequencing within a plant*.

Most studies in the operational-level group, which allow *some cross-plant routes*, assume that each plant can obtain mutual capacity support. Toba et al. (2005) studied the route-planning problem in a real-time environment. Wu and Chang (2007) examined the route-planning problem on a weekly schedule. Wu, Shih, et al. (2009) and Wu, Chen, et al. (2009) studied route planning in dual-plant scenarios with different limited transportation capacities and varying transportation times. Paolo et al. (2010) proposed a game theoretic protocol of cooperation and multiagent architecture to share capacity among different plants. These operational-level studies consider one factor between two plants—*route assignment*—.

Most studies consider either *job sequencing within a plant* or *route assignment* decisions. However, this paper considers both of these decisions simultaneously.

3. Problem statement

This section explains the *dual-plant* problem in detail, where the two plants called *Plant_A* and *Plant_B*. The following discussion first explains the assumptions in this study and then describes the problem.

Assumption 1. *The machine in each dual-plants stage is functionally comprehensive*. Each plant is equipped with the same machine functionality in each stage of job production.

Assumption 2. *Each job has eight routes*. The processing routes of each job fall into three segments. The break point between two stages of the processing route is called a *cut-off* point. A job has eight possible routes: $1 \to 1 \to 1$, $1 \to 1 \to 2$, $1 \to 2 \to 1$, $1 \to 2 \to 2$, $2 \to 2 \to 2$, $2 \to 1 \to 1$, $2 \to 1 \to 2$, $2 \to 2 \to 1$. The route $i \to j \to k$ indicates that the first segment i is processed in Stage 1, the second one is processed in Stage 2, and the third one is processed in Stage 3. The number 1 indicates that a job is manufactured in Plant_A, while 2 denotes Plant_B.

The problem addressed in this study consists of a set of n jobs ($j \in J$) to be processed in a dual-flow shop, and each plant has three machines. Each machine m can process one job at a time. The route of each job requires three sequential stages (three machines) to be completed. Each job j has a processing time p_{jm} on machine m. The scheduling objective is to minimize the coefficient of variation of slack time, in which the slack time of a job denotes the difference between its due date and total completion time.

4. Slack time evaluation for job sequences

Given a job sequence, this study uses a procedure to evaluate the slack time for completing all the jobs. This procedure emulates the processing flow of each job, and makes it possible to obtain the slack time (i.e., the difference between its due date and total completion time). The following section presents the notation and details of the procedure in this study.

Notation

j	index of job, $j = 1,2,\ldots,n$
i	index of stage, $i = 1,2,\ldots,m$
f	index of flow shop, $f = 1,2$
M	total number of stages
$p_{i,f,j}$	processing time required for stage i in flow shop f to process job j
$t_{i-1,i}$	transportation time for shipping a job from the $i-1$ stage to the next i stage
$\pi_{i,f}$	a job sequence for n jobs at the stage i of flow shop f, $\pi_{i,x} = [\pi_{i,x}(1),\ldots,\pi_{i,x}(n)]$
$\pi_{i,f}(j)$	the job in the jth position of a sequence at the stage i of flow shop f
d_j	the due date for job j
$C_{i,\pi_{i,f}(j)}$	the completion time of job $\pi_{i,f}(j)$ at stage i of the flow shop f
$A_{i,f,t}$	the time epoch in which machine in stage i of flow shop f becomes available; that is, when the machine (i,f) is free at t, $A_{i,l,t}$ is the last job-*completion-epoch* before t; while the machine (i,f) is in operation at t, $A_{i,f,t}$ is the *first job -completion-time after t*
S_j	slack time of job j
σ_s	the standard deviation of slack time
$\overline{X}_s$	the average slack time
CV_s	the coefficient of variation of slack time, $CV_s = \frac{\sigma_s}{\overline{X}_s}$

4.1. Evaluation procedure

The following five equations govern the procedure of evaluating the coefficient of variation of slack time:

$$C_{i,\pi(j)} = \max_{1 \leqslant f \leqslant 2}\{\max\{A_{i,f,t}, C_{i-1,f,\pi_{i-1}(j)} + t_{i-1,i}\} + p_{i,f,\pi_i(j)}\} \quad (1)$$

where $t = C_{i,\pi(j-1)}$ for $1 \leqslant i \leqslant M$

$$S_j = d_j - C_{M,\pi(j)} \quad (2)$$

$$\overline{X}_s = \frac{\sum_{j=1}^{J} S_j}{J} \quad (3)$$

$$\sigma_s = \sqrt{\frac{\sum_{j=1}^{J}(S_j - \overline{X}_s)^2}{J}} \quad (4)$$

$$CV_s = \frac{\sigma_s}{\overline{X}_s} \quad (5)$$

Eq. (1) indicates the completion time of job j in job $\pi(j)$ at stage i. The term presents the time epoch when machine at stage i is ready for processing job j; and the term $A_{i,f,t}$ denotes the time epoch when job j is ready to be processed at stage i. Eq. (2) indicates the slack time of job $\pi(j)$ at stage M. Eq. (3) indicates the average slack time of a job sequence. Eq. (4) indicates the standard deviation of slack time of a job sequence. Eq. (5) indicates the coefficient of variation of slack time of a job sequence.

5. Algorithms

This study proposes four algorithms (*GA-EDD-C*, *GA-EDD-S*, *GA-FIFO-C*, and *GA-FIFO-S*) to solve the dual-flow shop-scheduling problem. Two proposed cross-flow algorithms are respectively called *GA-EDD-C* and *GA-FIFO-C*, while two single flow shop scheduling algorithms are called *GA-EDD-S* and *GA-FIFO-S*—without any cross-shop routes. The single flow shop algorithm (*GA-FIFO-S*) serves as a benchmark to determine the effectiveness of the other algorithms. While all four algorithms follow the solution architecture of a typical GA and chromosome representation, they vary in their solution encoding schemes. This section first describes the solution architecture of a typical GA, and then describes the details of *GA-EDD-C, GA-EDD-S, GA-FIFO-C*, and *GA-FIFO-S*.

5.1. Solution architecture of a typical GA

A possible solution to a genetic algorithm (GA) is called a chromosome. A chromosome typically represents a string of integers, each of which is called a *gene*. The solution quality of a chromosome is called its fitness, which represents the coefficient of variation of slack time in this study (Holland, 1975). A GA aims to find a near-optimum chromosome using the evolutionary mechanism described below.

Procedure *Typical_GA_Evolution*

Step 1: Initialization
- Set $t = 0$
- Randomly create a set of N chromosomes to form initial population P_0

Step 2: Generate new chromosomes
- Use genetic operators to create N_c new chromosomes randomly
- Create a set $S = N_c \cup P_t$

Step 3: Update the population
- Set $t = t + 1$
- Adopt a strategy to select N chromosomes out of S to form P_t

Step 4: Termination check
- If (Termination = "yes") then **Output** the best chromosome in P_t and **STOP**.

Else go to Step 2.

Two common terminating conditions in Step 4 of the procedure above are (1) the best solution in $P(t)$ remains the same for more than T_b iterations, or (2) more than T_f iterations have been performed (i.e., $t = T_f$).

In Step 3, after completing genetic operations, the GA produces $Q = N_c + N$ chromosomes in $P(t)$ and newly generated ones, but only N of them are selected to form $P(t+1)$. To make this selection, first sort the Q chromosomes in descending order based on their fitness values. Denote the sorted chromosomes as $\pi_1, \ldots, \pi_Q$. In forming $P(t+1)$, the algorithm always selects π_1 and selects the other π_i based on probability by applying the *roulette wheel selection* method (Goldberg, 1989; Michalewicz, 1996).

5.2. GA-EDD-C algorithm

The GA-EDD-C algorithm represents the job sequence sorted by due date, where manufacturing operations adopt a cross-plant mechanism. This GA algorithm can be described as follows.

5.2.1. Chromosome representation and decoding

In the GA algorithm, each chromosome represents a job sequence that must be decoded to model a dual-flow shop scheduling solution. A chromosome is denoted by $\pi = [\pi_1, \ldots, \pi_i]$, $i \leqslant 3$, where π_i represents the job sequence at stage i. Each job sequence π_i is denoted by $\pi_i = [\pi_i(1), \ldots, \pi_i(n)]$, where π_i represents the job in the jth job sequence at stage i. The length of the chromosome equals the product of the total number of jobs and the total number of stages. For the chromosome in Fig. 2, $\pi = [\pi_1, \pi_2, \pi_3]$, $\pi_2 = [J_5, J_3, J_7, \ldots, J_2]$, and $\pi_2(3) = 7$ indicate that job J_7 is in the third job sequence at stage 2. The chromosome is 24 integers $(3 * 8)$ long.

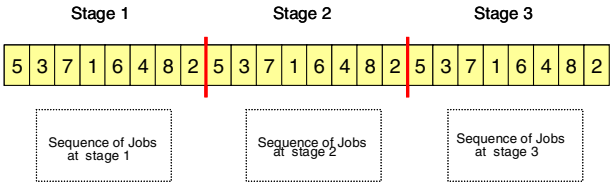

Fig. 2. Chromosome representation.

The chromosome π represents two set of sequencing decisions: (1) sequencing among stages (route assignment, assigning jobs to stages of dual-flow shop), and (2) job sequencing within each stage of a dual-flow shop. The decoding procedure consists of the two phases described below. In the first phase, the job_allocation procedure decodes the sequence of the stages of dual-flow shop. In the second phase, after the initial decoding, the due date-based decoding scheme decodes job sequencing within each stage i.

For job $\pi_i(j)$, the term $p_{i,f,\pi_i(j)}$ represents the processing time for a job j at stage i in flow shop f. Let $T_{i,f}$ denote the total processing time at the stage i in flow shop f for job $\pi_i(j)$. The procedure for interpreting the job allocation decision from a given chromosome is described below.

Procedure Job_Allocation

Step 1: Form the job groups in stage n
  $l = M$, /M is a large number*/
  $k = 1$, /*index of job group*/
  For $i = 1$ to N
    $T_{n,1} = \sum_{j=1}^{i} p_{i,1,\pi(j)}$; /*total processing time of the N jobs in stage n of Plant_A*/
    $T_{n,2} = \sum_{j=i+1}^{N} p_{i,2,\pi(j)}$; /*total processing time of the N jobs in stage n of Plant_B*/
    $\varepsilon = |T_{n,1} - T_{n,2}|$; /*the variance between total processing time in two plants*/
    If ($\varepsilon < l$) then;
      $k = i$; /*a cut-point of job group*/
      $l = \varepsilon$; /*update the least variance*/
      End if
      If ($i = N$) then
        go to Step 2 /*check if job group formation finished*/
      Endif
  Endfor
Step 2: Output job allocation results
  Output $C(k)$, $1 \leqslant k \leqslant f - 1$

Given the job allocation decision $C(k)$, the procedure for determining the job sequence decision for each stage is relatively easy. The job sequence for stage k $(1 \leqslant k \leqslant f-1)$ is $\pi_{ik} = [\pi_i(s), \ldots, \pi_i(e)]$, where $s = C(k-1) + 1$ and $e = C(k)$, where $C(0) = 0$ and $C(f) = \pi(N)$.

Fig. 3(a) illustrates the chromosome-decoding scheme using a two-phase example. In the first phase, the *job allocation of chromosome-decoding scheme*, there are eight jobs to be scheduled

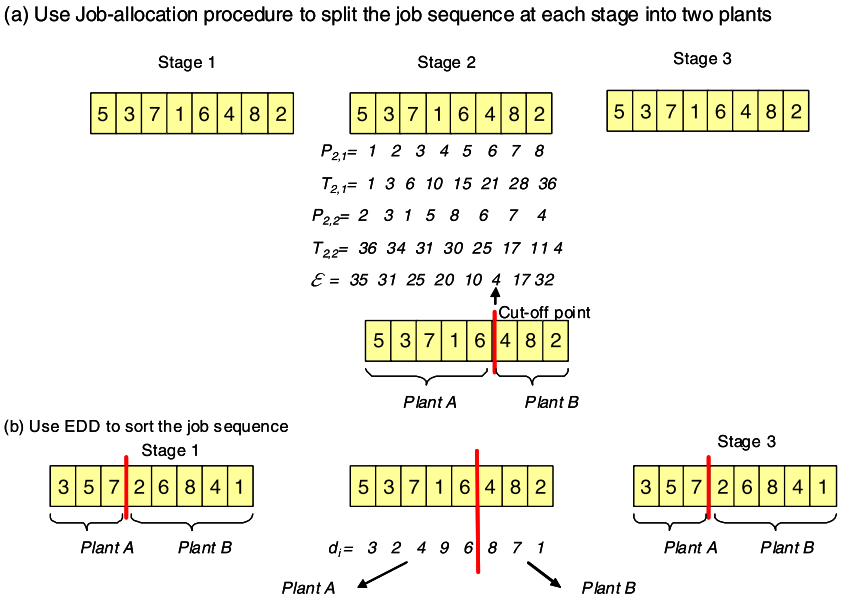

Fig. 3. GA-EDD-C chromosome and decoding schemes.

on two machines in Stage 2. Support that the processing times for these jobs in Plant 1 are $p_{1,i} = \{1,2,3,4,5,6,7,8\}$, and the processing time in Plant 2 are $p_{2,i} = \{2,3,1,5,8,6,7,4\}$. The accumulated processing time in Plant 1 and 2 are $T_{i,1} = \{1,3,6,10,15,21,28,36\}$, and $T_{i,2} = \{36,34,31,30,25,17,11,4\}$, respectively. The variance processing time between two plants is $\varepsilon = \{35,31,25,20,10,4,17,32\}$. The job cutoff point is located between $\pi(J_6)$ and $\pi(J_4)$. The set of jobs allocated to machine 1 and the job sequences are $\pi_{21} = [J_5,J_3,J_7,J_1,J_6]$, while those for the other machine are $\pi_{22} = [J_4,J_8,J_2]$.

In the second phase, the *due-date decode scheme*, the earlier a due date appears in π_{ik}, the higher is its sequencing priority. For instance, the job sequence at Stage 2 is $J_5 - J_3 - J_7 - J_1 - J_6 - J_4 - J_8 - J_2$ and the due-dates are $d_i = \{3,2,4,9,5,6,7,1\}$. Applying the due date-sequence decoded scheme sorts the job sequence at Stage 2 in Fig. 3(b) in *due-date* order, i.e., $\pi_{21} = [J_3,J_5,J_7,J_6,J_1]$, $\pi_{22} = [J_2,J_8,J_4]$. As a result, the post-decoding result of chromosome π is a *due date*-based job sequence. This job sequence first imposes capacity constraints on the sequence among stages and then determines a job sequence within each stage.

5.2.2. Genetic operators

We used two types of *genetic* operators to generate the new chromosomes. The first genetic operator is a crossover operator, and the second is a mutation operator. A crossover operator generates a new pair of chromosomes from an existing pair of chromosomes, while a mutation operator generates a new chromosome from an existing one.

The one-crossover operators involve **C1** (Reeves, 1995). The one-mutation operators are **Swap** (Wang & Zheng, 2003). As stated in Procedure *Typical_GA_Evolution*, each iteration generates N_c new chromosomes. These new chromosomes are generated as follows: $N \cdot P_{c1}$ ones through **C1**, $N \cdot P_{m1}$ ones through **Swap**, where $N_c = N(P_{c1} + P_{m1})$.

The following subsection explains the mechanism of the one-crossover operators, where *parent-1* and *parent-2* represent the parent chromosomes and *child-1* and *child-2* represent the created chromosomes.

C1 operator: Fig. 4 shows that one randomly selected point splits each parent into two sections (*head* and *tail*). To generate an offspring (say, *child-2*), the *head* is copied from the head of *parent-2*—a string (3,5,6) in this case. The *tail* is determined by sequentially referring to the genes of *parent-1*; only the gene values not in the *head* of *child-2* appear in the tail. This yields a string (1,4,9,8,7,2) as the *tail* of *child-2*.

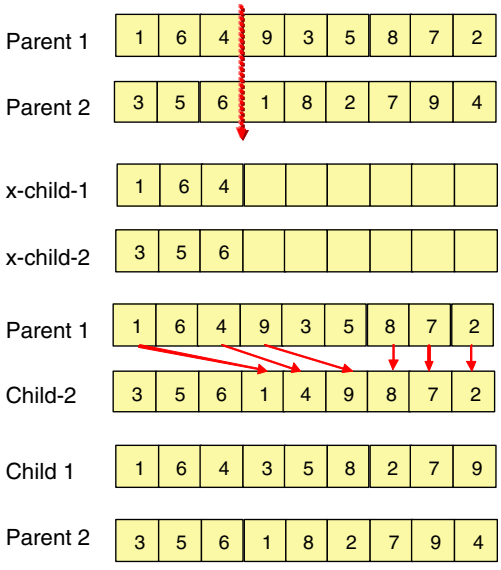

Fig. 4. Crossover operators: C1.

The following subsection describes the one-mutation operator, where π_a denotes the parent chromosomes and π_b denotes the child chromosomes.

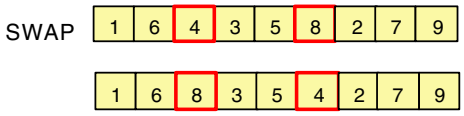

Fig. 5. Mutation operators: Swap.

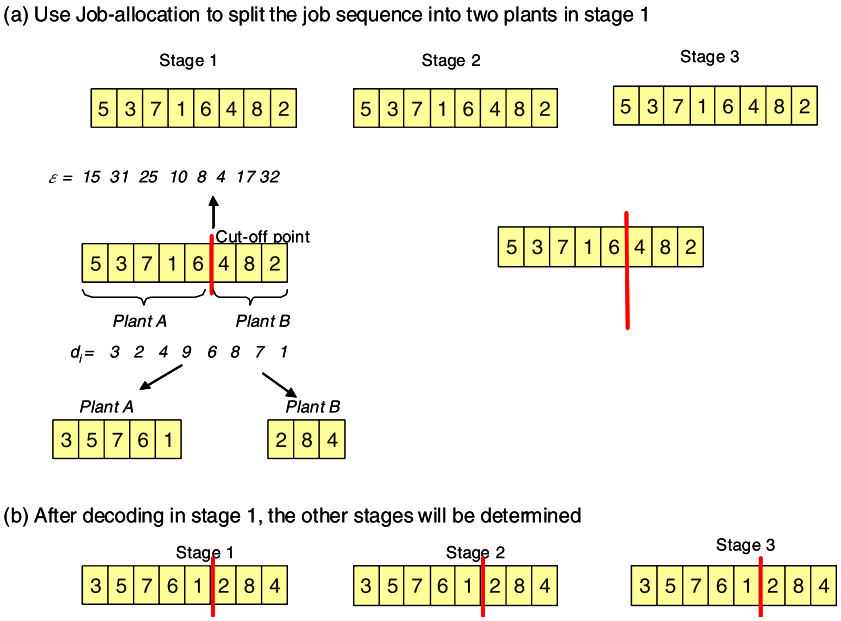

Fig. 6. GA-EDD-S chromosome and decoding schemes.

Swap operator: Fig. 5 shows that we randomly selected two distinct genes in π_a, and then swapped their gene values to generate π_b.

5.3. GA-EDD-S algorithm

The *GA-EDD-S* algorithm sorts the job sequence by due-date, but only a single plant can perform manufacturing operations. The GA algorithm can be described as follows. The *GA-**EDD-S*** algorithm adopts the same chromosome as *GA-EDD-C*, but adopts a different decoding scheme. Refer to the *GA-EDD-S* chromosome in Fig. 6(a). In the first phase, *job allocation of chromosome-decoding scheme*, the first segment denotes that the job sequence in Stage 1 is $J_5 \rightarrow J_3 \rightarrow J_7 \rightarrow J_1 \rightarrow J_6 \rightarrow J_4 \rightarrow J_8 \rightarrow J_2$. Applying the ***Job_Allocation*** procedure to split the jobs into two plants produces the decoding results of $\pi_{11} = [J_5,J_3,J_7,J_1,J_6]$ and $\pi_{12} = [J_4,J_8,J_2]$ for this chromosome. In other words, all the jobs in Stage 1 should be completed before proceeding to jobs in Stage 2 or Stage 3.

In the second phase, the job sequence in three segments of the chromosome is the same because the operation of a job in each plant cannot occur in the other plant using the cross-plant mechanism.

The *GA-EDD-S* algorithm creates a new chromosome by applying a genetic operator to each chromosome segment independently, and then joining the newly generated segments to form a new chromosome. The two genetic operators (C1 and Swap) are similar to those in the *GA-EDD-C* in creating new *GA-EDD-S* chromosomes.

5.4. GA-FIFO-C algorithm

The *GA-FIFO-C* algorithm sorts the job sequence by *first-in first-out*, and allows the cross-plant manufacturing mechanism. The chromosomes in the *GA-FIFO-C* algorithm are similar to those in *GA-EDD-C*, but vary in the second phase of the decoding procedure. Fig. 7 shows that we applied the job_allocation procedure to split the job sequence into two plants at three stages. Thus, a *GA-FIFO-C* chromosome represents a job sequence and imposes no *due-date* based on the sequence among machines.

Fig. 7 shows a *GA-FIFO-C* chromosome, which is much like the pre-coding chromosome of *GA-EDD-C*. This chromosome produces a scheduling solution at three stages by applying the ***Job_Allocation*** procedure. To generate new chromosomes in The *GA-FIFO-C* generates new chromosomes by applying a genetic operator to a whole chromosome, like *GA-EDD-C*. The GA-FIFO-C algorithm uses the two genetic operators mentioned above.

5.5. GA-FIFO-S algorithm

The chromosomes in the *GA-FIFO-S* algorithm are similar to those in the *GA-EDD-S* algorithm, but vary in the second phase of the decoding procedure. Fig. 8 shows that we applied the

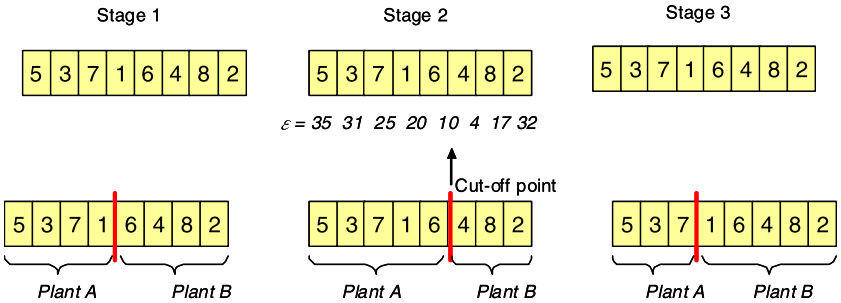

Fig. 7. GA-FIFO-C chromosome and decoding schemes.

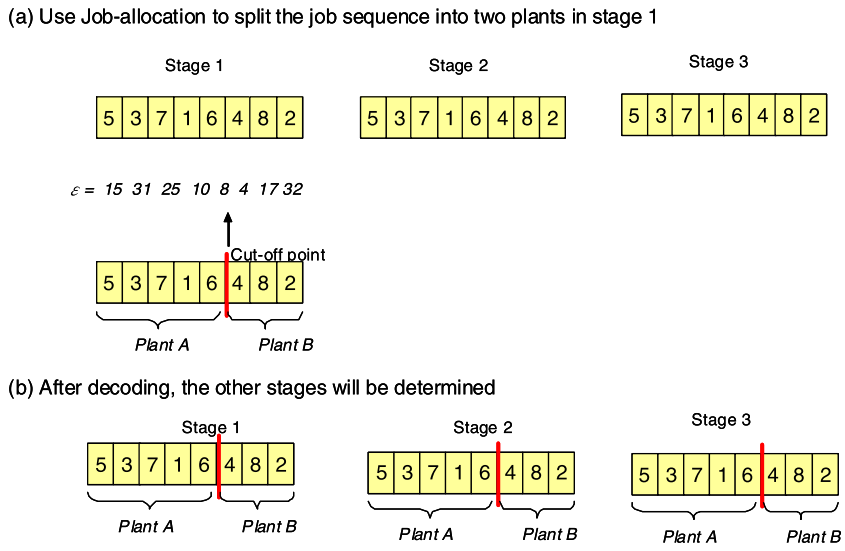

Fig. 8. ***GA-FIFO-S*** chromosome and decoding schemes.

job_allocation procedure to split the job sequence into two plants at Stage 1. In other words, a *GA-FIFO-S* chromosome directly represents a job sequence and imposes no due-date based on the sequence among machines.

Fig. 8(a) shows a chromosome in *GA-FIFO-S* algorithm, which is similar to the pre-coding chromosome of *GA-EDD-S*. Fig. 8 shows that the chromosome at Stage 1 represents a scheduling solution by applying the ***Job_Allocation*** procedure. The job sequences in Stage 2 and Stage 3 then follow the job sequence in Stage 1. The *GA-FIFO-S* algorithm generates new chromosomes by applying a genetic operator to the whole chromosome, much like the *GA-EDD-S* algorithm, and uses the two genetic operators mentioned above.

6. Numerical experiments

Numerical experiments were conducted to compare the four algorithms (*GA-EDD-C*, *GA-EDD-S*, *GA-FIFO-C*, *GA-FIFO-S*). Personal computers with PENTIUM Dual-Core 2.8 GHz CPU and 1 Gb memory were used to run the programs, which were coded in Visual C++. The parameters of the four genetic algorithms were set as follows: $N = 100$, $P_c = 0.8$, $P_m = 0.2$, $T_b = 1000$, $T_f = 100{,}000$.

6.1. Experiment design

Table 1 shows that the dual-flow shop considered in this study had two plants, six stages, and six machines. The operation time of each machine (stage) i is a uniform distribution $[a_i, b_i]$. The due date of each job j is a uniform distribution $[a_i, b_i]$ shown as Table 2.

We used (P,N,T) to represent a test case, where P represents the proportion of processing time at the same stage of plant A to that of plant B, N represents the number of jobs, T represents the ratio of transportation time to processing time. Table 2 shows that P has four options (1:1, 1:1.5, 1:2, 1:3), N has five options (20, 40, 60, 80, 100 jobs), and T has three options (0.1:1, 0.5:1, 2:1). Thus, each algorithm had $4 \times 5 \times 3 = 60$ test cases and each test case was obtained by averaging the experimental results of 15 replicates.

The average coefficients of variation of slack time for the four algorithms were defined as $CV_{GA-EDD-C}$, $CV_{GA-EDD-S}$, $CV_{GA-FIFO-C}$, and $CV_{GA-FIFO-S}$. Experimental results indicate that the *GA-EDD-C* algorithm outperformed the other algorithms in most cases. A performance metric illustrates the degree of variation in solution quality, $\gamma_x = (CV_x - CV_{GA-EDD-C})/CV_x$, to compare the solution quality between the *GA-EDD-C* and a benchmark algorithm (say, x). A positive γ_x indicates that *GA-EDD-C* is better, while a negative value indicates that *GA-EDD-C* is worse.

Table 1
Process times of three stages in a dual-flow shop.

Plant	Stage 1	Stage 2	Stage 3
A	U[1,5]	U[1,5]	U[1,5]
B	U[1,5]	U[1,5]	U[1,5]

Table 2
Due date of each job under the different number of jobs.

	20 jobs	40 jobs	60 jobs	80 jobs	100 jobs
Due date	U[51,70]	U[160,180]	U[270,290]	U[380,400]	U[490,510]

6.2. Experimental results

Table 3 compares the solution quality of the four algorithms by averaging the experimental results obtained under the four processing time ratios and three transportation time ratio options mentioned above. This table indicates that the proposed dual-flow shop algorithm (*GA-EDD-C*) outperforms the three other algorithms (*GA-EDD-S*, *GA-FIFO-C* and *GA-FIFO-S*) in most cases. This is because *GA-EDD-C* adopts the cross-flow shop and due-date based scheduling approach to reduce the total setup number. This in turn reduces the total setup time and alleviates the effects of machine capacity loss. This finding advocates the use of the cross flow shop-based approach to solve the dual-flow shop-scheduling problem.

Table 3 shows the experimental results of γ_x. The *GA-EDD-C* algorithm outperforms the other three algorithms in terms of γ_x in most cases. Of the 60 test cases, γ_x ranged from 0% to 65%. Fig. 9 shows the average of γ_x for each algorithm, indicating that the *GA-EDD-C* algorithm outperforms the other algorithms.

Fig. 10 shows that a lower P (process time ratio between two plants) increases the average of γ_x. When P reached 1:3, the average of γ_x reached 55%. This is due to the large gap in process time

Table 3
Solution qualities of the four algorithms under different *J*, *P*, and *T*.

Jobs	*P*	*T* (%)	$CV_{GA-EDD-C}$	$CV_{GA-EDD-S}$	$r_{GA-EDDS}$ (%)	$CV_{GA-FIFO-C}$	$r_{GA-FIFO-C}$ (%)	$CV_{GA-FIFO-S}$	$r_{GA-FIFOS}$ (%)
20	1:1	1	0.112	0.135	17	0.517	78	0.111	−1
		50	0.108	0.144	25	0.503	79	0.111	3
		200	0.134	0.124	−8	0.679	80	0.123	−9
	1:1.5	1	0.112	0.144	23	0.395	72	0.125	11
		50	0.123	0.142	13	0.523	76	0.128	4
		200	0.156	0.126	−24	0.726	79	0.132	−19
	1:2	1	0.140	0.170	18	0.389	64	0.156	10
		50	0.143	0.172	17	0.562	75	0.149	4
		200	0.144	0.165	13	0.612	77	0.150	4
	1:3	1	0.177	0.348	49	0.653	73	0.218	19
		50	0.182	0.353	49	0.627	71	0.221	18
		200	0.201	0.467	57	0.709	72	0.222	9
40	1:1	1	0.098	0.107	9	0.249	61	0.097	−1
		50	0.097	0.109	10	0.234	58	0.097	−1
		200	0.097	0.114	15	0.272	64	0.097	−1
	1:1.5	1	0.117	0.144	18	0.132	11	0.121	3
		50	0.116	0.138	16	0.129	10	0.121	4
		200	0.113	0.143	21	0.159	29	0.121	7
	1:2	1	0.137	0.215	36	0.194	29	0.147	7
		50	0.140	0.228	39	0.195	28	0.147	5
		200	0.139	0.243	43	0.173	20	0.147	5
	1:3	1	0.157	0.516	69	0.346	54	0.695	77
		50	0.160	0.546	71	0.417	62	0.547	71
		200	0.179	0.575	69	0.568	68	0.308	42
60	1:1	1	0.089	0.101	11	0.157	43	0.090	1
		50	0.089	0.100	11	0.168	47	0.090	2/0
		200	0.087	0.101	13	0.164	47	0.090	3
	1:1.5	1	0.112	0.135	17	0.135	17	0.117	4
		50	0.112	0.134	16	0.133	15	0.117	4
		200	0.109	0.150	27	0.136	19	0.117	6
	1:2	1	0.128	0.213	40	0.176	27	0.137	6
		50	0.132	0.190	31	0.176	25	0.137	3
		200	0.130	0.206	37	0.176	26	0.137	5
	1:3	1	0.149	0.432	66	0.205	27	0.640	77
		50	0.150	0.450	67	0.207	28	0.659	77
		200	0.157	0.458	66	0.212	26	0.664	76
80	1:1	1	0.094	0.103	9	0.163	42	0.089	−5
		50	0.094	0.104	10	0.165	43	0.089	−5
		200	0.093	0.103	10	0.196	53	0.089	−4
	1:1.5	1	0.111	0.129	14	0.117	6	0.108	−2
		50	0.111	0.133	17	0.130	15	0.108	−2
		200	0.108	0.148	27	0.118	9	0.108	1
	1:2	1	0.126	0.196	36	0.147	14	0.125	−1
		50	0.129	0.190	32	0.163	21	0.125	−3
		200	0.128	0.183	30	0.147	13	0.125	−2
	1:3	1	0.136	0.413	67	0.191	29	0.581	77
		50	0.137	0.390	65	0.202	32	0.569	76
		200	0.140	0.366	62	0.195	28	0.605	77
100	1:1	1	0.094	0.103	9	0.159	41	0.090	−4
		50	0.094	0.103	9	0.206	54	0.090	−4
		200	0.093	0.103	10	0.227	59	0.090	−3
	1:1.5	1	0.114	0.142	19	0.126	9	0.113	−1
		50	0.114	0.148	23	0.126	9	0.113	−1
		200	0.112	0.143	22	0.126	11	0.113	1
	1:2	1	0.134	0.214	37	0.231	42	0.132	−2
		50	0.134	0.212	36	0.200	33	0.131	−2
		200	0.131	0.206	36	0.191	32	0.132	0
	1:3	1	0.143	0.370	61	0.196	27	0.555	74
		50	0.146	0.380	62	0.197	26	0.571	74
		200	0.158	0.377	58	0.200	21	0.570	72

between two plants, which benefits the cross-flow shop mechanism.

Fig. 11 shows the average of γ_x for different transportation time ratios for the 60 test cases, indicating that the ratio of transportation time to processing time is not obvious to γ_x. In terms of computation time, Fig. 12 shows that all four algorithms are quite efficient, requiring less than eight minutes for each test case.

7. Concluding remarks

This study examines a dual-flow shop-scheduling problem by comprehensively considering the processing time and transportation time features. We propose four algorithms (*GA-EDD-C*, *GA-EDD-S*, *GA-FIFO-C*, and *GA-FIFO-S*) to solve the dual-flow shop-scheduling problem. The *GA-EDD-C* algorithm serves as a

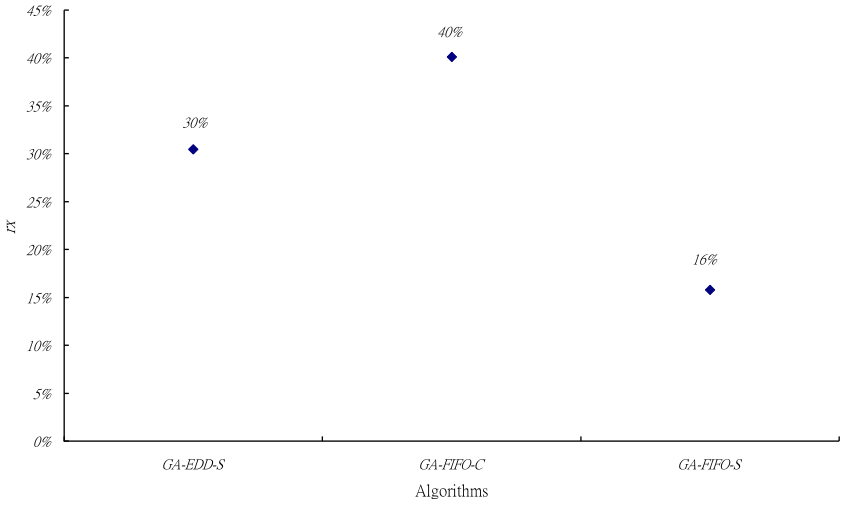

Fig. 9. Average of γ_x for various algorithms.

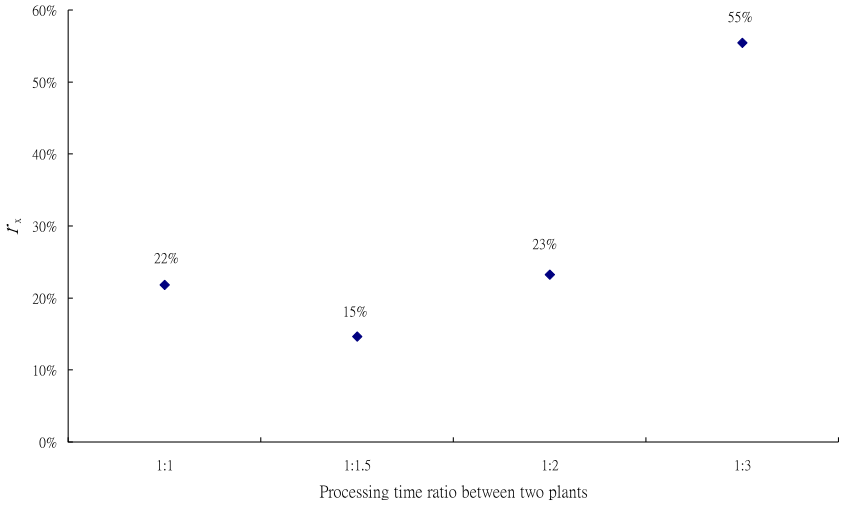

Fig. 10. Average of γ_x at various processing time ratios between two plants.

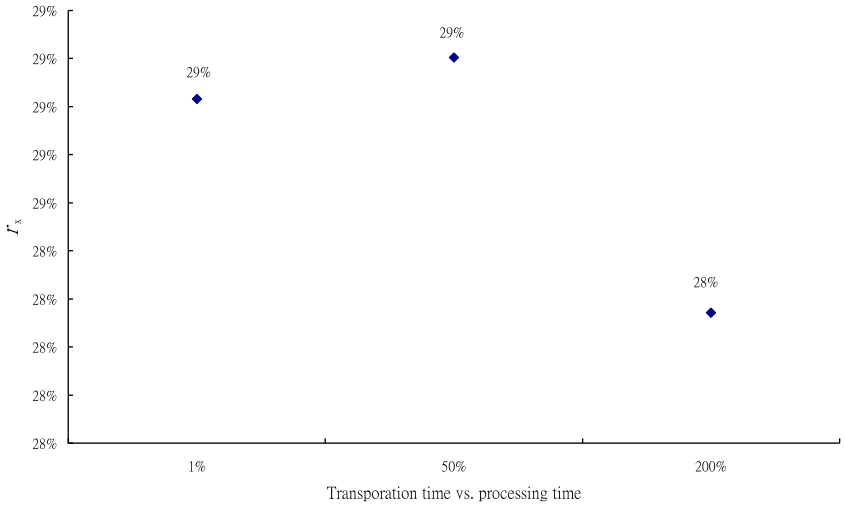

Fig. 11. Average of γ_x at various transportation time-processing time ratios.

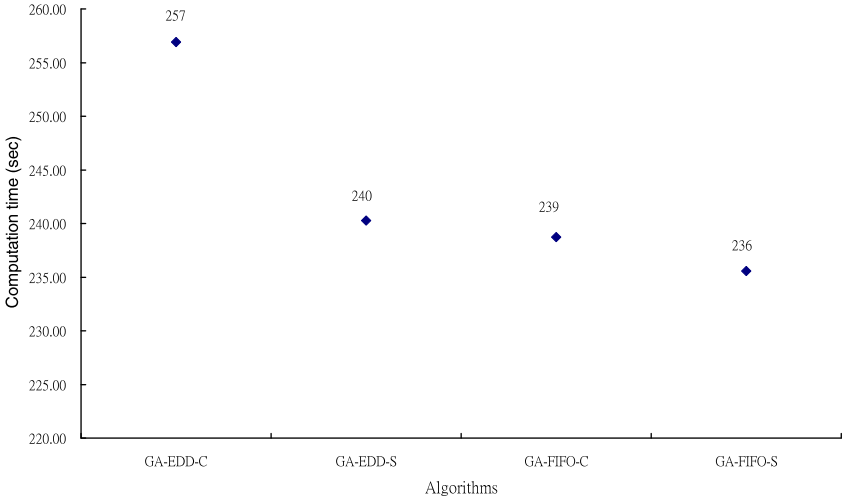

Fig. 12. Computation times for the four types of algorithms.

benchmark. Numerical experiments indicate that the proposed *GA-EDD-C* algorithm outperforms the other algorithms. These results imply that a dual-flow shop approach should be used to solve the scheduling problem.

The experiments in this study indicate that the *GA-EDD-C* algorithm outperforms other algorithms when there is a large gap in processing times between two plants. This performance difference may be due to by the advantages of the cross-flow shop.

An extension to this study is the scheduling of three or more flow shops. Such an extension would require another decision-making criterion: how to allocate jobs to each stage among different flow shops.

Acknowledgements

This research was sponsored by National Science Council, Taiwan under a research contract NSC99-2221-E-009-110-MY3.

References

Chiang, D., Guo, R. S., Chen, A., Cheng, M. T., & Chen, C. B. (2007). Optimal supply chain configurations in semiconductor manufacturing. *International Journal of Production Research, 45*(3), 631–651.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in neural and artificial systems*. Ann Arbor, MI: Univ. Michigan Press.

Lee, Y. H., Chung, S., Lee, B., & Kang, K. H. (2006). Supply chain model for the semiconductor industry in consideration of manufacturing characteristics. *Production Planning and Control, 17*(5), 518–533.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). Berlin Heidelberg New York: Springer.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computation Operation Research, 22*(1), 5–13.

Renna, P., & Argoneto, P. (2010). A game theoretic coordination for trading capacity in multisite factory environment. *International Journal of Advanced Manufacturing Technology, 47*, 1241–1252.

Toba, H., Izumi, H., Hatada, H., & Chikushima, T. (2005). Dynamic load balancing among multiple fabrication lines through estimation of minimum inter-operation time. *IEEE Transactions on Semiconductor Manufacturing, 18*, 202–213.

Wang, L., & Zheng, D. Z. (2003). An effective hybrid heuristic for flowshop scheduling. *International Journal of Advanced Manufacturing Technology, 21*(1), 38–44.

Wu, M. C., & Chang, W. J. (2007). A short-term capacity trading method for semiconductor fabs with partnership. *Expert Systems with Applications, 33*, 476–483.

Wu, M. C., Chen, C. F., & Shih, C. F. (2009). Route planning for two wafer fabs with capacity-sharing mechanisms. *International Journal of Production Research., 47*(20), 5843–5856.

Wu, S. D., Erkoc, M., & Karabuk, S. (2005). Managing capacity in the high-tech industry: A review of literature. *The Engineering Economist, 50*, 125–158.

Wu, M. C., Shih, C. F., & Chen, C. F. (2009). An efficient approach to cross-fab route planning for wafer manufacturing. *Expert Systems with Applications, 36*, 10962–10968.