A combined dispatching criteria approach to scheduling dual flow-shops

Chie-Wun Chiou\textsuperscript{a}, Wen-Min Chen\textsuperscript{a} & Muh-Cherng Wu\textsuperscript{a}

\textsuperscript{a}Department of Industrial Engineering and Management, National Chiao Tung University, Hsin-Chu 300, Taiwan
Published online: 22 Jun 2012.

To cite this article: Chie-Wun Chiou, Wen-Min Chen & Muh-Cherng Wu (2013) A combined dispatching criteria approach to scheduling dual flow-shops, International Journal of Production Research, 51:3, 927-939, DOI: 10.1080/00207543.2012.693636
To link to this article: http://dx.doi.org/10.1080/00207543.2012.693636

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the "Content") contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions
A combined dispatching criteria approach to scheduling dual flow-shops

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

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

(Received 28 December 2010; final version received 17 April 2012)

This study investigates a dual flow-shops scheduling problem. In the scheduling context, there are two flow shops and each shop involves three processing stages. The two shops are functionally identical but their stage processing times for a job are different. While sequentially going through the three processing stages, each job is allowed to travel between the two shops. That is, for a job, each of its three stages could be processed in any of the two shops. Such a context is called dual flow-shops in the sense that the two flow shops’ capacities are completely shared. The scheduling problem involves two decisions: (1) route assignment (i.e. assigning the processing stages of a job to a shop), and (2) job sequencing (i.e. sequencing the jobs waiting before each stage). The scheduling objective is to minimise the coefficient of variation of slack time ($CV_S$), in which the slack time (also called lateness) denotes the difference between the due date and total completion time of a job.

We propose five genetic-algorithm-based (GA-based) solution methods to solve the scheduling problem, which are called GA-EDD, GA-FIFO, GA-SPT, GA-LFO, and GA-COMBO respectively. Numerical experiments indicate that GA-COMBO outperforms the other four methods.

Keywords: scheduling; cross-shop production; dual flow-shops; combined dispatching criteria; genetic algorithm (GA)

1. Introduction

Dual plants are a common production scenario for some manufacturing companies. The context of dual-plants denotes that a company has two distinct plants, which are physically located in neighbouring regions so that their capacity could be mutually supported through transporting jobs between the two plants.

The emergence of dual plants is essentially due to the risk of high investment in capacity expansion. An over-investment decision may lead to financial crisis; therefore, manufacturing companies tend to expand their capacity in a step-wise manner. That is, if a company has two plants, the first plant may have been in operation over a few years before the second plant emerges.

This research addresses a production scenario of dual plants with the following three features. Firstly, the two plants are functionally identical flow shops; each shop involves a sequence of three distinct processing stages. Due to the advance of manufacturing technology, the processing efficiency in the two shops may be different. Secondly, the two plants must be scheduled in a capacity-sharing paradigm in order to alleviate the problem of machine underutilisation. This paradigm implies that a job has to undergo three operations and each operation could be processed in any of the two shops. Thirdly, the transportation time between the two plants is a substantial amount (compared against the processing times) and cannot be ignored in scheduling.

Compared with prior literature on flow shop scheduling, the addressed production scenario is unique in considering the cross-plant transportation time. That is, if the transportation time is so trivial and can be ignored, then such a dual plants scenario can be seen as a single flow shop in which each workstation involves two machines with various processing efficiencies. The scheduling of such a single flow shop has been extensively studied. Yet, scheduling for the addressed dual plants scenario has been rarely examined.

This research intends to study the scheduling of the addressed dual plants scenario. Due to the aforementioned route flexibility in job processing, the scheduling of the dual plants scenario involves two decisions: (1) route assignment (i.e. assigning the processing stages of a job to one of the two plants), and (2) job sequencing (i.e. sequencing the jobs waiting before each stage in each plant). We define the scheduling objective as minimising...
the coefficient of variation of slack time ($CV_S$), in which the slack time (also called lateness) denotes the difference between the due date and total completion time of a job.

This research proposes five genetic-algorithm-based (GA-based) solution methods to solve the scheduling problem, which are called GA-EDD, GA-FIFO, GA-SPT, GA-LFO, and GA-COMBO respectively. These five algorithms are based on a common GA algorithmic flow but are distinct in adopting different sequencing rules. The abbreviations of these algorithms are explained below: EDD (earliest due date), FIFO (first-in-first-out), SPT (shortest processing time), LFO (least slack first) and COMBO (a combination of multiple sequencing rules). Extensive numerical experiments have been carried out, and the results indicate that GA-COMBO outperforms the other four algorithms.

The remainder of this paper is organised as follows. Section 2 reviews literature on capacity sharing/trading between two neighbouring plants. Section 3 describes the scheduling problem in more detail. Section 4 explain why the objective function $CV_S$ is adopted and how to compute $CV_S$. Section 5 presents the common GA solution architecture of these five GA-based algorithms. Section 6 discusses how to adopt design of experimental (DOE) methodology to GA-COMO algorithms as well as their distinctions in algorithms. Section 6 reports experiments and results, and concluding remarks are in Section 7.

2. Relevant literature

Prior studies on how to effectively utilise capacity between two neighbouring plants can be grouped into two categories. The first category assumes that the capacities of the two plants cannot be shared; that is, the production of a job can only be carried out in one plant (i.e. cross-plant production is prohibited). In contrast, the second category assumes that capacities of the two plants can be shared or traded; namely, a cross-plant route for the production of a job is tolerable.

In the first category (i.e. capacity cannot be shared), most studies focused on solving a job assignment problem; that is, given two or more neighbouring plants, how to effectively assign each job to a particular plant. Prior literature typically develops mathematical programming models to solve such problems in various contexts (e.g. Lee et al. 2006, Chiang et al. 2007). A comprehensive survey of such studies has been published by Wu et al. (2005).

In the second category (i.e. capacity can be shared), prior studies can be classified into two tracks. One track focused on the route assignment decisions. The other track focused on the capacity-trading decisions between the two plants. Example studies on the route assignment decisions include Toba et al. (2005) and Wu et al. (2009a, 2009b). Example studies on the capacity-trading decisions include Wu and Chang (2007), Chen et al. (2008), Chiang et al. (2010) and Renna and Argoneto (2011).

In the route assignment studies, Wu et al. (2009a, 2009b) focused on how to plan the production routes of jobs in advance to optimise capacity utilisation. In contrast, Toba et al. (2005) addressed the route assignment problem in shop floor control context; that is, how to determine the production route of a job in a real time manner by dynamically considering machine breakdown status.

In the capacity-trading studies, Wu and Chang (2007) investigated how to determine the capacity-trading amount for each workstation of the two plants in a weekly manner. Chen et al. (2008) focused on how to determine the capacity-trading price. In addressing such capacity trading decisions, Chiang et al. (2010) adopted the concept of auction while Paolo et al. (2010) and Renna and Argoneto (2010) applied the methodology of game theory. In summary, such studies intended to periodically regulate the work station capacity of each plant by a capacity-trading mechanism in order to optimise the overall capacity utilisation. The obtained capacity-trading decision in turn can be used to plan the production routes of jobs (i.e. making the route assignment decision).

Prior studies on capacity-sharing of two plants have established significant milestones on the route assignment decisions. However, the job sequencing issue has been rarely addressed in these studies. In this research, we attempt to study a scheduling problem in a dual-plant capacity-sharing context, in which both the route assignment decision and the job sequencing decision must be considered.

3. Problem statement

This section explains the dual-plant scheduling problem in more detail. The scheduling problem comes from a real case faced by a manufacturing company that produces substrates for mounting semiconductor devices. The three
assumptions of the scheduling context are first presented. Then, we describe the two scheduling decisions and finally discuss the objective function.

**Assumption 1:** The two plants are functionally identical but may be different in processing efficiency. As shown in Figure 1, in the scheduling context, there are two neighbouring plants (Plant_A and Plant_B). Each plant is composed of three sequential stages, and each stage represents a manufacturing process. Of the stages, A_i is functionally identical with B_i but the processing time of B_i might be smaller because the manufacturing process of B_i is relatively advanced than that of A_i.

**Assumption 2:** Each job has eight possible processing routes due to the adoption of a capacity-sharing paradigm. Each of the two plants is a flow shop. The completion of a job must go through the three sequential stages, each of which is either located at Plant_A or Plant_B. This implies that there are eight possible processing routes for manufacturing a job. Namely, the processing route of a job can be represented by \( S_1 \rightarrow S_2 \rightarrow S_3 \), in which \( S_1 \) is either \( A_i \) or \( B_i \).

**Assumption 3:** The transportation time between the two plants is a substantial amount. The two plants are physically neighbouring but the required transportation time is a substantial amount (compared against the job processing times at each stage) and cannot be ignored in scheduling. In contrast, the transportation time between any two stages in a particular plant is trivial can be ignored in scheduling. This assumption implies that the scheduling context is not a flexible flow shop which has been extensively studied in literature.

The scheduling problem can be described below. Consider a set of \( N \) jobs to be processed by the two plants. The scheduling of the \( N \) jobs involves two decisions: (1) route assignment and (2) job sequencing decisions. The first decision is to assign each processing stage of a job to either Plant_A or Plant_B. The assignment results imply that one of the eight processing routes for manufacturing a job has been selected. For example, the processing route for a particular job might be like \( A_1 \rightarrow A_2 \rightarrow B_3 \). After the route assignment decision, the second decision is to determine the processing sequence of those jobs that have been assigned to each stage (i.e., each \( A_i \) and \( B_i \)).

The objective of the scheduling problem is to minimise the coefficient of variation of lateness (\( CV_s \)). Suppose \( c_j \) represent the completion time of job \( j \) and \( d_j \) represent its due date. Then, the lateness of job \( j \) is \( L_j = d_j - c_j \). And the objective function is \( CV_s = \frac{\sigma_s}{\bar{x}_s} \), where \( \sigma_s \) is the standard deviation of \( L_s \) and \( \bar{x}_s \) is the mean of \( L_s \).

Herein, the aforementioned \( L_j \) (the lateness of job \( j \)) is also called the slack time of job \( j \). If \( L_j > 0 \), it implies that \( L_j \) is the slack (time buffer) that has been allocated to job \( j \). In turn, such a slack serves a time buffer for dealing with unexpected events requested by the customer of the job. Consider a production environment where unexpected events requested by customers frequently appear. In such an environment, we have to reserve a slack time for each job in order to deal with the possible coming of unexpected events. Such a slack reservation policy tends to prohibit frequent changes of the production schedule.

Consider the cases with \( \bar{x}_s > 0 \). The objective function (\( CV_s = \frac{\sigma_s}{\bar{x}_s} \)) is a the-smaller-the-better metric, which has two implications. Firstly, a larger \( \bar{x}_s \) denotes that on average we have allocated a larger slack time to each customer. Namely, a larger \( \bar{x}_s \) implies that the average time buffer is longer; and the service quality (in terms of quickly responding unexpected requirements from customers) is better. Secondly, it implies that the slack of each job shall be as equally treated as possible. That is, if the slack time of each job is identical (\( \sigma_j = 0 \)), then \( CV_s = 0 \). That is, the service quality (time buffer) provided to each customer shall be as equal as possible. In summary, the metric (\( CV_s = \frac{\sigma_s}{\bar{x}_s} \)) is a multiple objective function, which includes two objectives. One is \( \bar{x}_s \) and the other is \( \sigma_s \). In addition, \( CV_s = \frac{\sigma_s}{\bar{x}_s} \) denotes that we attempt to minimise \( \sigma_s \) and also maximise \( \bar{x}_s \).

Notice that in some test cases we may obtain \( \bar{x}_s < 0 \), which implies that many jobs have no slack time at all; namely they will be lately-delivered. In such cases, \( CV_s = \frac{\sigma_s}{\bar{x}_s} \) becomes a negative number because we always have \( \sigma_s \geq 0 \). This implies that \( CV_s = \frac{\sigma_s}{\bar{x}_s} \) shall be a the-larger-the-better metric while \( \bar{x}_s < 0 \); in contrast, \( CV_s = \frac{\sigma_s}{\bar{x}_s} \) shall be.
To resolve such a conflict in justification, we use \( \frac{1}{CV_s} = \frac{X_s}{s} \) as a performance metric in the GA search. Apparently, \( \frac{1}{CV_s} = \frac{X_s}{s} \) is a the-larger-the-better metric whatever \( X_s \) is positive, zero, or negative. While the near-optimal solution is obtained from GA, we proceed to compute its \( CV_s \).

4. Algorithms

As stated, we proposed five GA-based algorithms to solve the scheduling problem. In the five algorithms, a scheduling solution is modelled by a sequence of jobs (called a chromosome). By a decoding method, one chromosome will yield a route assignment decision. Given such a route assignment decision, we could use a dispatching rule to make the job sequencing decision. These five algorithms are essentially the same in using a GA algorithmic flow to generate new chromosomes but are different in embedding various dispatching rules (e.g. FIFO, EDD, SPT, LSF, COMBO). These five algorithms are thus named GA-FIFO, GA-EDD, GA-SPT, GA-LSF, and GA-COMBO.

In this section, we firstly describe the chromosome representation scheme and then present the decoding method used to obtain the route assignment decision of a chromosome. Secondly, given a route assignment decision, we describe how to use a dispatching rule to determine the job sequence. Thirdly, we describe the shared GA algorithmic architecture and some of its details.

4.1 Chromosome representation and decoding

We first present how to represent a chromosome. As stated, each job must go through three operations (stages); and each operation can be processed either in Plant_A or Plant_B. Consider a scheduling problem that has \( N \) jobs. To represent a scheduling solution, a chromosome is composed of a sequence of the \( N \) jobs. Consider an example problem that has eight jobs in total. The processing times and due dates of these eight jobs are shown in Table 1.

In the context of the example problem, we can have a chromosome as shown in Figure 2(a). We then present how to decode a chromosome to obtain its route assignment decision. Given a chromosome, we first duplicate the chromosome for each stage and try to split each duplicated one into two parts by applying a load-balance paradigm. Consider the example chromosome shown in Figure 2(a). The three duplicated ones are shown in Figure 2(b). For any stage, each of the eight jobs must be assigned either to Plant_A or Plant_B. The method for making the job assignment decision is based on a load-balance paradigm. That is, for any stage, the job loading assigned to each of the two plants shall be as equal as possible. The load-balancing paradigm is carried out by splitting the duplicated chromosome into two parts, and each part shall be as equal as possible in terms of job loading.

For example, see the stage 1 in Figure 2(b), the duplicated chromosome now has been split into two parts. The first parts involves the first five jobs \( \{J_5, J_3, J_7, J_1, J_6\} \) which shall be assigned to Plant_A, while the remaining three jobs \( \{J_4, J_8, J_2\} \) are in the remaining part and shall be assigned to Plant_B. With such a job assignment,
the loadings for the two plants are quite close. The total processing times required for Plant_A is 15 units while that for Plant_B is 17 units.

The method for determining the cut-off point required for splitting a chromosome is relatively simple. As shown in Figure 2(b), we travel through the chromosome from left to right and compute $T_{j,A}$ (the accumulated processing times from left to right until completing job $j$ at Plant_A) and do it likewise from right to obtain $T_{j,B}$ (the accumulated processing times from right to left until completing job $j$ at Plant_B). In the figure, $p_{j,k}$, $T_{j,k}$, $e_j$ and $\varepsilon_r$ are the processing time for job $j$ respectively required at Plant_k, the accumulated job processing times at Plant_k starting from the leftmost job to job $j$, $e_j = |T_{j,A} - T_{j,B}|$.

The method for determining the cut-off point required for splitting a chromosome is relatively simple. As shown in Figure 2(b), we travel through the chromosome from left to right and compute $T_{j,A}$ (the accumulated processing times from left to right until completing job $j$ at Plant_A) and do it likewise from right to obtain $T_{j,B}$ (the accumulated processing times from right to left until completing job $j$ at Plant_B). In the figure, $P_{j,A}$ and $P_{j,B}$ are the processing time for job $j$ respectively required at Plant_A and Plant_B. To determine the cut-off point, we first compute an index $e_j = |T_{j,A} - T_{j,B}|$, and identify the job $j^*$ which has the smallest $e_j$ ($j^* = J_4$ in this example, with $e_4 = 4$). Then, for the left-hand and right-hand neighbors of job $j^*$, select the job which is relatively smaller in $e_j$, ($J_6$ is selected in this example, with $e_6 = 10$). As a result, the cut-off point is set to split the two identified jobs ($J_4$ and $J_6$ in this example).

4.2 Dispatching and job sequencing

As stated above, we can yield the route assignment decision of a given chromosome by the decoding method and the load-balancing paradigm. With the decision obtained, the duplicated chromosome at each stage now has been spitted into two parts (also called two split-chromosomes); and jobs in a particular split-chromosome shall be processed at the same plant. In turn, we need to make the job sequencing decision for each split-chromosome. The method for determining the job sequence of a split-chromosome is by a dispatching rule. For example, see Figure 2(c), if we apply the EDD (earliest due date) rule for the job dispatching decisions, the job sequence of the first split-chromosome at Stage_1 now has changed and becomes $J_5 \rightarrow J_7 \rightarrow J_3 \rightarrow J_6 \rightarrow J_1$. Noticeably, the application of different dispatching rules would result in different job sequence in each split-chromosome.

To justify which dispatching rules would be more effective, we use five dispatching rules (FIFO, EDD, SPT, LSF, and COMBO) in this research. This in turn yields five GA-based algorithms; that is, GA-FIFO, GA-EDD, GA-SPT, GA-LSF, and GA-COMBO. The first four dispatching rules, relatively simpler and widely used in
literature, are briefly introduced below. The last one COMBO, developed by this research, is relatively complex and will be explained in the section.

Consider a number of jobs waiting to be processed by a machine, and we need to determine the priority for processing these jobs. FIFO (first-in-first-out) denotes that the job which arrives at the machine earlier has a higher priority; EDD (earliest-due-date) denotes that the job with an earlier due date has a higher priority; SPT (shortest-processing-time) denotes that the job with a smaller processing time has a higher priority; LSF (least-slack-time) denotes that the job with a smaller slack time has a higher priority. Herein, the slack time of a job is defined as $RT - RPT$, where $RT$ is the remaining time of the job and $RPT$ is the remaining processing time.

4.3 Algorithmic architecture

In the five GA-based algorithms, a chromosome denotes a scheduling solution. Given a chromosome, we can obtain its route assignment decision and job sequencing decisions by the chromosome decoding method, the load-balance paradigm, and the adopted dispatching rule. In turn, we can evaluate the objective function (also called fitness function) of the chromosome. The five GA-based algorithms share a common algorithmic flow adapted from a pioneer one (Holland 1975), which is designed to iteratively generating and selecting chromosomes in order to ultimately obtain a near-optimum scheduling solution.

The algorithmic flow shared by the five GA-based algorithms is presented by a procedure called $GA_Evolution$ as below.

**Procedure $GA_Evolution$**

**Step 1:** Generate initial population
- Randomly create a set of $N$ chromosomes to form initial population $P_0$;
- Set $t = 0$ (i.e. $P_t = P_0$);

**Step 2:** Create new chromosomes
- Use a crossover operator to create $N_c = P_c \cdot N$ new chromosomes;
- Use a mutation operator to create $N_m = P_m \cdot N$ new chromosomes;
- Create a set $S = P_t \cup N_c \cup N_m$

**Step 3:** Update population $P_t$
- Set $t = t + 1$; /* update the iteration index*/
- Select $N$ chromosomes from $S$ to form $P_t$ by tournament selection method
- Record the best quality chromosome in $P_t$ (called $X_{best}$).

**Step 4:** Check termination
- If ($X_{best}$ keeps the same for $T_b$ iterations) or ($t = T_f$)
  then output $X_{best}$ (the obtained solution) and STOP.
- Else, go to Step 2

Some details of **Procedure $GA_Evolution$** are further explained below. In Step 3, the tournament selection method, which has been widely used in literature, is adopted for selecting $N$ chromosome out of $N + N_c + N_m$ chromosomes. The basic idea of this selection strategy is intentionally giving a higher selection probability to a chromosome which is better in solution quality (Goldberg 1989, Michalewicz 1996). This implies that a relatively inferior chromosome still has a probability to be selected. Such a strategy is to avoid the GA solution-search process terminate too early, due to being trapped into a local optimum.

In Step 2, we use two operators (crossover and mutation operators) to generate new chromosomes, in which $0 < P_c < 1$ and $0 < P_m < 1$ are parameters given by users. Noticeably, while any the two operators is applied on a chromosome, only one segment on the chromosome is manipulated and the other two ones are unchanged (reminded that a chromosome comprise three segments). The mechanisms of the two operators are explained below, in which the chromosomes used to generate new ones are called parents and those newly generated ones are called children.
The mutation operator (also called SWAP), widely used in literature, is a one-to-one conversion. That is, given one parent chromosome, we can generate one child chromosome with an example as shown in Figure 3. For the parent chromosome, we randomly choose any two jobs (e.g., $J_4$ and $J_7$ are chosen) and swap them to form a new chromosome.

The crossover operator, also called LOX (linear order crossover) operator, is adapted from (Croce et al. 1995). This operator is a two-to-two conversion. That is, given two parents chromosomes, we can generate two children chromosomes with an example as shown in Figure 4. In the figure, we randomly select two cutoff points to split the chromosome into three parts. Now, the middle part of Parent_1 involves $J_4$, $J_1$, and $J_6$, which implies that these corresponding jobs in Parent_2 must be replaced by ‘H’ to generate X_Child_2; and X_Child_1 is generated accordingly. Then, we move the ‘H’ symbols to the middle parts while keeping the sequence of the remaining jobs unchanged; the results are Y_Child_1; and Y_Child_2. Finally, we place the middle part of Parent_2 on the ‘H’ symbols of Y_Child_1 and obtain Child_1, Child_2 can accordingly be obtained.

5. GA-COMBO algorithm

This section presents the dispatching decision of the GA-COMBO algorithm. As stated in Section 4.2, the dispatching decision is to be applied on each split-chromosome at each stage. Referring to Figure 2, the jobs in each split-chromosome shall be sequenced. In the GA-COMBO, we make the dispatching decision based on a dispatching priority indicator $p = \alpha \cdot n_{DD} + \beta \cdot n_{TCT} + \gamma \cdot n_{ST}$. In the following, we first explain the indicator $p$ and proceed to explain how to appropriately determine the three parameters $\alpha$, $\beta$ and $\gamma$.
5.1 Dispatching priority indicator

In the GA-COMBO, we define a dispatching priority indicator

\[ p = \alpha \cdot n_{DD} + \beta \cdot n_{TCT} + \gamma \cdot n_{ST} \]

for each job at each split-chromosome to make the dispatching decision. Herein, \( \alpha \), \( \beta \), and \( \gamma \) are a set of weightings, each of which is a positive real number and \( \alpha + \beta + \gamma = 1 \). Moreover, \( n_{DD} \) is defined as \( n_{DD} = \frac{(x_{DD} - \bar{x}_{DD})}{\sigma_{DD}} \) where \( x_{DD} \) represents the due date of a job, \( \bar{x}_{DD} \) and \( \sigma_{DD} \) respectively represent the mean of and standard deviation of \( x_{DD} \) for all jobs. Accordingly, we define \( n_{TT} = \frac{(x_{TT} - \bar{x}_{TT})}{\sigma_{TT}} \) and \( n_{ST} = \frac{(x_{ST} - \bar{x}_{ST})}{\sigma_{ST}} \), where \( x_{TT} \) represents the total processing time of a job and \( x_{ST} \) represents the slack time (i.e. slack time = due date – total processing time) of a job.

The management implication of the dispatching priority indicator \( p \) is explained below. Firstly, a job with a lower \( x_{DD} \) value implies that the job is relatively more urgent and shall be given a higher priority. Secondly, a job with a lower \( x_{TT} \) implies that the job relatively takes less time to finish and shall be given a higher priority in order to reduce the waiting times of other jobs (i.e., consider the advantage of using shortest processing time as the dispatching rule). Thirdly, a job with a lower \( x_{ST} \) implies that the job relatively has less slack time (available time resource) and shall be given a higher priority.

Respectively highlighting a unique criterion in making job dispatching decision, the three dispatching metrics \( (x_{DD}, x_{TT}, \text{and } x_{ST}) \) might be in conflict in suggesting the sequence of two given jobs. To resolve such a conflict, we develop the dispatching priority indicator \( p = \alpha n_{DD} + \beta n_{TT} + \gamma n_{ST} \) to combine the three dispatching criteria, in which the three dimensional metrics \( x_{DD}, x_{TT}, \text{and } x_{ST} \) have been normalised as dimensionless metrics \( n_{DD}, n_{TT}, \text{and } n_{ST} \) in order to reduce the effect of using incompatible unit. Moreover, \( \alpha, \beta, \text{and } \gamma \) are given weightings (i.e. given parameters) of the three dimensionless criteria. In summary, the lower is the \( p \) value of a job; the higher is its sequencing priority.

5.2 Determine parameters \( \alpha, \beta \text{ and } \gamma \)

As stated, \( \alpha, \beta \text{ and } \gamma \) are given weightings (given parameters). To obtain an effective dispatching result, we have to make an appropriate choice of these three parameters. Inspired by the applications proposed by (Dabbas et al. 2003), we adopt the design of experiment (DOE) methodology to determine the three parameters \( \alpha, \beta \text{ and } \gamma \).

In the DOE methodology, we firstly choose 10 sets of \( (\alpha, \beta, \gamma) \) as shown in Table 2. Different choices of \( (\alpha, \beta, \gamma) \) imply that different weightings are imposed on the three metrics. Therefore, each set of \( (\alpha, \beta, \gamma) \) essentially represents a unique GA-COMBO program; the scheduling results and performance obtained by each of these 10 GA-COMBO programs are then very likely to be different. That is, by running the 10 GA-COMBO programs, we could obtain 10 such data sets \( \{(\alpha_i, \beta_i, \gamma_i)\} \) \( i = 1, \ldots, 10 \), where \( y_i \) is the obtained performance of \( i \)th GA-COMBO program.

With the 10 data sets \( \{(\alpha_i, \beta_i, \gamma_i)\} \), we could apply the response surface method (RSM) to obtain a polynomial function \( y = f(\alpha, \beta, \gamma) \), typically called the response surface. Then, by the application of typical DOE software, we could obtain an optimal set of parameters \( (\alpha^*, \beta^*, \gamma^*) \). In turn, we could propose a GA-COMBO program embedded with \( (\alpha^*, \beta^*, \gamma^*) \) which is seemingly optimal to solve the scheduling problem.

<table>
<thead>
<tr>
<th>Design point</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>( 1/2 )</td>
<td>( 1/2 )</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>( 1/2 )</td>
<td>0</td>
<td>( 1/2 )</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>( 1/2 )</td>
<td>( 1/2 )</td>
</tr>
<tr>
<td>7</td>
<td>( 1/3 )</td>
<td>( 1/3 )</td>
<td>( 1/3 )</td>
</tr>
<tr>
<td>8</td>
<td>( 2/3 )</td>
<td>( 1/3 )</td>
<td>( 1/3 )</td>
</tr>
<tr>
<td>9</td>
<td>( 1/3 )</td>
<td>( 2/3 )</td>
<td>( 1/3 )</td>
</tr>
<tr>
<td>10</td>
<td>( 1/3 )</td>
<td>( 1/3 )</td>
<td>( 2/3 )</td>
</tr>
</tbody>
</table>
6. Numerical experiments

Numerical experiments are carried out to compare the five proposed scheduling algorithms (GA-EDD, GA-SPT, GA-FIFO, GA-LFO, and GA-COMBO). These five algorithms are coded with Visual C++, and the DOE and RSM methods used in the GA-COMBO are implemented with Minitab (Minitab Inc. 2003). Personal computers equipped with PENTIUM Dual-Core 2.8 GHz CPU and 1 Gb memory are used in the experiments. Parameters used in the five GA-algorithms are given as follows: \( N = 100, \ P_e = 0.8, \ P_m = 0.2, \ T_b = 1,000, \ T_f = 100,000 \), where \( N \) represents the population size, \( P_e \) is the mutation rate, \( P_m \) is the crossover rate, \( T_b \) and \( T_f \) respectively define the two criteria for terminating the GA program.

6.1 Experimental design

As stated, the scheduling context addresses a dual flow shop, which is composed of two neighbouring flow shops. In the numerical experiments, the transportation time between the two flow shops is 10 min, 50 min, 100 min, and 200 min. We consider four scenarios (denoted by \( S_1, S_2, S_3 \) and \( S_4 \)), which involve four sets of processing times and job due dates as shown in Table 3. Consider that there are \( N \) jobs to be scheduled. The processing times and due date of each job are sampled from the uniform distributions of each scenario in Table 3.

The two flow shops may have different processing efficiency; that is, the processing time of an operation required in one shop may be multiple times that required in the other shop. We consider two cases of processing efficiency: \( P_1 = (1 : 1) \) and \( P_2 = (1 : 3) \), in which \( P_1 \) denotes that the two shops have the same efficiency, and \( P_2 \) denotes that one shop is three times that of the other shop in processing an operation.

In summary, an experiment run can be represented by a 6-tuple vector \((A, S, P, J, T, R)\), where \( A \) denotes an algorithm, \( S \) denotes a processing time scenario, \( P \) denotes a processing efficiency ratio between the two flow shops, \( J \) denotes the options of job size, \( T \) denotes the setting of transportation time, and \( R \) denotes the number of experiment replicates. As stated, we have five algorithms, four processing time scenarios, two processing efficiency ratios, five options of job size \((J = 20, 40, 60, 80, 100)\), four settings of transportation time \((T = 10 \text{ min}, 50 \text{ min}, 100 \text{ min}, 200 \text{ min})\), and for each test case we perform 15 replicates and take their average outcome as the obtained result. As a result, we have to carry out 12,000 experiment runs; that is, 5 (algorithm) × 4 (job size) × 4 (transportation time) × 15 (replications) × 5 (algorithm) = 90,000.

6.2 Experimental results

As stated, the five algorithms are compared by a performance metric: the coefficient of variation of lateness \( (CV_S) \). Herein, the performance metric of each algorithm is respectively denoted by \( CV_{GA-EDD}, CV_{GA-FIFO}, CV_{GA-SPT}, CV_{GA-LFO}, \) and \( CV_{GA-COMBO} \). To compare the five algorithms, we take GA-Combo as the benchmark, and define a solution quality indicator \( \gamma_x = (CV_x - CV_{GA-COMBO})/CV_x \), where the subscript \( x \) denotes an algorithm other than GA-COMBO. A positive \( \gamma_x \) value denotes that GA-COMBO outperforms the \( x \) algorithm; conversely, a negative \( \gamma_x \) value denotes that GA-COMBO is worse.

Table 4 shows the experiments results in which the transportation time is set at 10 min. Notice that in the table \( \gamma_x \) in each cell is the average of 15 replicates. The table indicates that GA-COMBO outperforms the other four algorithms. In addition, at four various settings of transportation time, we found that GA-COMBO also outperforms the other four algorithms (see Table 5).
Table 4. Solution quality comparisons of the five algorithms, $T = 10$ min.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Jobs</th>
<th>$P$</th>
<th>$CV_{GA-COMBO}$</th>
<th>$CV_{GA-EDD}$</th>
<th>$r_{GA-EDD}$</th>
<th>$CV_{GA-SPT}$</th>
<th>$r_{GA-SPT}$</th>
<th>$CV_{GA-FIFO}$</th>
<th>$r_{GA-FIFO}$</th>
<th>$CV_{GA-LFO}$</th>
<th>$r_{GA-LFO}$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>20</td>
<td>1:1</td>
<td>0.890</td>
<td>0.890</td>
<td>0%</td>
<td>2.221</td>
<td>150%</td>
<td>4.079</td>
<td>359%</td>
<td>9.038</td>
<td>916%</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1:1</td>
<td>3.243</td>
<td>3.243</td>
<td>0%</td>
<td>6.577</td>
<td>103%</td>
<td>4.924</td>
<td>52%</td>
<td>4.984</td>
<td>54%</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1:1</td>
<td>0.781</td>
<td>0.781</td>
<td>0%</td>
<td>1.173</td>
<td>50%</td>
<td>1.147</td>
<td>47%</td>
<td>1.368</td>
<td>75%</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>1:1</td>
<td>1.931</td>
<td>1.931</td>
<td>0%</td>
<td>2.611</td>
<td>35%</td>
<td>2.146</td>
<td>11%</td>
<td>2.400</td>
<td>24%</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1:1</td>
<td>1.190</td>
<td>1.190</td>
<td>0%</td>
<td>2.125</td>
<td>79%</td>
<td>2.059</td>
<td>73%</td>
<td>1.738</td>
<td>46%</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1:3</td>
<td>1.458</td>
<td>1.517</td>
<td>4%</td>
<td>2.952</td>
<td>103%</td>
<td>4.753</td>
<td>226%</td>
<td>1.866</td>
<td>28%</td>
<td>0.879</td>
<td>0.000</td>
<td>0.121</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>1.603</td>
<td>1.609</td>
<td>0%</td>
<td>2.947</td>
<td>84%</td>
<td>3.133</td>
<td>95%</td>
<td>3.239</td>
<td>102%</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$S_2$</td>
<td>20</td>
<td>1:1</td>
<td>1.185</td>
<td>2.807</td>
<td>137%</td>
<td>1.185</td>
<td>0%</td>
<td>2.666</td>
<td>125%</td>
<td>1.666</td>
<td>41%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1:1</td>
<td>1.205</td>
<td>2.453</td>
<td>104%</td>
<td>1.205</td>
<td>0%</td>
<td>2.137</td>
<td>77%</td>
<td>1.237</td>
<td>3%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1:1</td>
<td>1.086</td>
<td>2.862</td>
<td>164%</td>
<td>1.086</td>
<td>0%</td>
<td>2.365</td>
<td>118%</td>
<td>3.365</td>
<td>210%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>1:1</td>
<td>1.096</td>
<td>2.746</td>
<td>151%</td>
<td>1.096</td>
<td>0%</td>
<td>2.159</td>
<td>97%</td>
<td>4.159</td>
<td>280%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1:1</td>
<td>1.052</td>
<td>2.843</td>
<td>170%</td>
<td>1.052</td>
<td>0%</td>
<td>2.984</td>
<td>184%</td>
<td>1.894</td>
<td>89%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>1.019</td>
<td>2.045</td>
<td>101%</td>
<td>1.019</td>
<td>0%</td>
<td>2.409</td>
<td>136%</td>
<td>1.821</td>
<td>74%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$S_3$</td>
<td>20</td>
<td>1:1</td>
<td>0.432</td>
<td>0.560</td>
<td>30%</td>
<td>0.621</td>
<td>44%</td>
<td>0.452</td>
<td>5%</td>
<td>0.435</td>
<td>1%</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1:1</td>
<td>0.474</td>
<td>0.619</td>
<td>30%</td>
<td>0.793</td>
<td>67%</td>
<td>0.475</td>
<td>0%</td>
<td>0.486</td>
<td>3%</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1:1</td>
<td>0.333</td>
<td>0.411</td>
<td>23%</td>
<td>0.403</td>
<td>21%</td>
<td>0.364</td>
<td>9%</td>
<td>0.332</td>
<td>0%</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>1:1</td>
<td>0.322</td>
<td>0.409</td>
<td>27%</td>
<td>0.360</td>
<td>12%</td>
<td>0.324</td>
<td>1%</td>
<td>0.324</td>
<td>1%</td>
<td>0.295</td>
<td>0.392</td>
<td>0.313</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1:1</td>
<td>0.454</td>
<td>0.573</td>
<td>26%</td>
<td>0.534</td>
<td>18%</td>
<td>0.488</td>
<td>8%</td>
<td>0.454</td>
<td>0%</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>0.450</td>
<td>0.591</td>
<td>31%</td>
<td>0.631</td>
<td>40%</td>
<td>0.476</td>
<td>6%</td>
<td>0.455</td>
<td>1%</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>$S_4$</td>
<td>20</td>
<td>1:1</td>
<td>1.167</td>
<td>2.071</td>
<td>77%</td>
<td>1.773</td>
<td>52%</td>
<td>1.452</td>
<td>24%</td>
<td>1.879</td>
<td>61%</td>
<td>0.288</td>
<td>0.424</td>
<td>0.288</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1:1</td>
<td>1.290</td>
<td>2.461</td>
<td>91%</td>
<td>1.922</td>
<td>49%</td>
<td>1.721</td>
<td>33%</td>
<td>2.280</td>
<td>77%</td>
<td>0.296</td>
<td>0.394</td>
<td>0.310</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1:1</td>
<td>1.109</td>
<td>1.618</td>
<td>46%</td>
<td>1.850</td>
<td>67%</td>
<td>1.314</td>
<td>18%</td>
<td>1.909</td>
<td>72%</td>
<td>0.333</td>
<td>0.352</td>
<td>0.314</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>1:1</td>
<td>1.147</td>
<td>1.750</td>
<td>53%</td>
<td>1.907</td>
<td>66%</td>
<td>1.261</td>
<td>10%</td>
<td>1.440</td>
<td>26%</td>
<td>0.253</td>
<td>0.374</td>
<td>0.374</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1:1</td>
<td>1.098</td>
<td>1.776</td>
<td>62%</td>
<td>2.111</td>
<td>92%</td>
<td>1.292</td>
<td>18%</td>
<td>1.525</td>
<td>39%</td>
<td>0.296</td>
<td>0.401</td>
<td>0.303</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>1.082</td>
<td>1.523</td>
<td>53%</td>
<td>1.901</td>
<td>76%</td>
<td>1.161</td>
<td>7%</td>
<td>1.212</td>
<td>12%</td>
<td>0.243</td>
<td>0.355</td>
<td>0.404</td>
</tr>
<tr>
<td>Total average:</td>
<td>0.838</td>
<td>1.200</td>
<td>43%</td>
<td>1.308</td>
<td>56%</td>
<td>1.647</td>
<td>75%</td>
<td>1.465</td>
<td>75%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Now we proceed to analyse the effect of transportation time on job routes. Intuitively, the increase of transportation time tends to increase the cycle time of a job which undergoes a cross-shop travelling route. Therefore, while the transportation time is increased, the number of jobs with cross-shop travelling routes tends to be reduced. As a result, the number of jobs wholly processed in a single-shop would increase.

To characterise the effect of transportation time on job routes, we define a single-shop routing indicator \((\rho)\), which denotes the percentage of experiment runs that include at least one job travelling with a single-shop route. In turn, \((1 - \rho)\), defined as the cross-shop routing indicator, denotes the percentage of experiment runs in which all job travels in a cross-shop route.

As stated, an experiment run can be described by a vector \((A, S, P, J, T, R)\). Consider the GA-COMBO algorithm at a particular setting of transportation time, we have to carry out 600 experiment runs, \(600 = A \times S \times P \times J \times T \times R = 1 \times 4 \times 2 \times 5 \times 1 \times 15\). Of the 600 experiment runs, the values of \(\rho\) and \((1 - \rho)\) at various settings of transportation time are shown in Table 6. The table indicates that \(\rho\) consistently increase while the transportation time constantly increases. This implies that the GA-COMO scheduling algorithm tends to advocate the use of single-shop routings while the transportation time is increased. Consider an extreme case, in which the transportation time is close to infinite, we would expect that each job will undergo a single-shop route; that is, no job will undergo a cross-shop route.

### 6.3 Analysis of results

The experiment results are further analysed herein. See Table 4, the optimal weighing factors of \(\alpha, \beta, \gamma\) appears to be problem dependent. In addition, the GA-COMBO suggest to give a high weighting to EDD in scenario \(S_1\), a high weighting to SPT in scenario \(S_2\), a high weighting to LSF in scenario \(S_3\), and about equal weighting to the three dispatching rules in scenario \(S_4\). This implies that \((\alpha, \beta, \gamma)\) for all instances in a particular scenario appear to be similar; and \((\alpha, \beta, \gamma)\) are significantly varied among different scenarios. Such profoundly interesting results are due to that the four scenarios are intentionally designed based on some criteria. By referring to Table 3, we shall explain the criteria for designing each scenario.

Scenario \(S_1\) involves three design criteria. First, jobs are with high variation in their due dates (i.e. the range is 200). Second, the job processing times of the first stage are with low variation (i.e. the range is only five).
Third, the total processing time of a job is proportional to its due date. Under such a design criteria, the processing times and slack (due date − total processing time) of all jobs are with lower variation. In contrast, the due dates of jobs are with higher variations. The GA-COMBO suggest the use of \((\alpha, \beta, \gamma) = (1, 0, 0)\), which implies that EDD seems to be the best dispatching rule in scenario \(S_1\).

Scenario \(S_2\) involves three design criteria. First, jobs are with low variation in their due dates (i.e. the range is 20 only). Second, the job processing times of the first stage are with high variation (i.e. the range is 100). Third, the total processing time of a job is proportional to its due date. Under such a design criteria, the due dates and slack (due date − total processing time) of all jobs are with lower variation. In contrast, the processing times of jobs are with higher variations. The GA-COMBO suggest the use of \((\alpha, \beta, \gamma) = (0, 1, 0)\), which implies that SPT seems to be the best dispatching rule in scenario \(S_2\).

Scenario \(S_3\) involves three design criteria. First, jobs are with relatively high variation in their due dates (i.e. the range is 100). Second, the job processing times of the first stage are with low variation (i.e. the range is only five). Third, the total processing time of a job is not proportional to its due date; this is intended to result in a high variation in slack (due date − total processing time). Under such a design criteria, the due dates and processing times of all jobs are with lower variation. In contrast, the slack of jobs are with higher variations. The GA-COMBO mostly suggest the use of \((\alpha, \beta, \gamma) = (0, 0, 1)\), which implies that LSF seems to be the best dispatching rule in scenario \(S_3\).

Scenario \(S_4\) involves three design criteria. First, jobs are with low variation in their due dates (i.e. the range is 20). Second, the job total processing times of the three stages are with low variation (i.e. the range is 20). This in turn leads to that the slacks (due date − total processing time) of all jobs are also with low variation. The GA-COMBO suggests that each value of \(\alpha, \beta, \gamma\) is a substantial amount, and cannot be ignored in most cases (see Table 4).

7. Concluding remarks

This study examines a dual flow shop scheduling problem. In the scheduling context, there are two neighbouring flow shops and each flow shop involves three stages. Compared with processing times, the transportation time between the two shops are significant while that between any two consecutive stages are trivial and can be ignored. Such a scheduling problem, unlike a single flow shop with parallel machines, has been rarely examined in literature.

The scheduling problem involves two decisions: (1) assigning operation to stages, and (2) sequencing operations at each stage. We propose five GA-based algorithms (GA-EDD, GA-SPT, GA-FIFO, GA-LSF and GA-COMBO) to solve the scheduling problem. These five GA-based algorithms adopt a common algorithmic architecture in searching an optimal operation assignment decision, in which different operation sequencing criteria are embedded. The four sequencing criteria (EDD, SPT, FIFO, and LSF) have been widely used in literature, while the COMBO criterion is developed by applying the DOE and RSM methods.

Numerical experiments indicate that the GA-COMBO algorithm outperformed the other four proposed algorithms; in particular it appears much better in a complex environment with high variation processing times. That is, the GA-COMBO algorithm is quite adaptive, adaptive to dealing with the high variation situations. This advocates the use of the GA-COMBO algorithm in scheduling complex scenarios.

One extension to this research is the scheduling of three or more flow shops. This extension increases the complexity of job assignment decisions – how to assign an operation of a job to one among the various flow shops. To reduce computational complexity, we may have to consider an alternative approach; that is, most jobs (say, 80%) are assigned to a particular shop (i.e. cross-shop production is prohibited), and the remaining 20% jobs are allowed for cross-shop production.

Acknowledgements

This research is financially sponsored by National Science Council, Taiwan, under a research contract NSC 100-2221-E-009-059-MY3.

References


