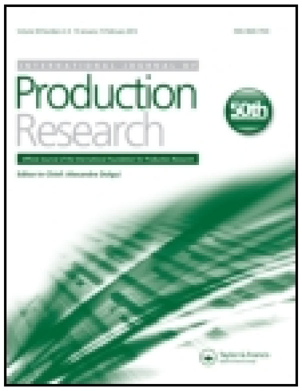


International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

A combined dispatching criteria approach to scheduling dual flow-shops

Chie-Wun Chiou [a] , Wen-Min Chen [a] & Muh-Cherng Wu [a]
[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

International Journal of Production Research
Vol. 51, No. 3, 1 February 2013, 927–939

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

*Corresponding author. Email: cw_chiou@yahoo.com.tw

ISSN 0020–7543 print/ISSN 1366–588X online
© 2013 Taylor & Francis
http://dx.doi.org/10.1080/00207543.2012.693636
http://www.tandfonline.com

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

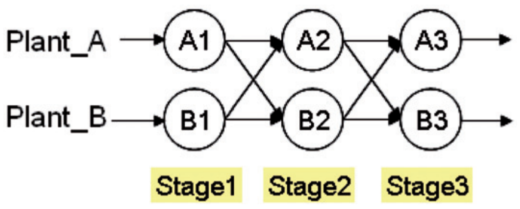


Figure 1. Production flow of typical dual flow shop.

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_i 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 σ_s is the standard deviation of L_j and $\bar{X}_s$ is the mean of L_j.

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 ($\sigma_s = 0$), 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 σ_s. In addition, $CV_s = \frac{\sigma_s}{\bar{X}_s}$ denotes that we attempt to minimise σ_s and also maximise $\overline{X}_s$.

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

Table 1. Due dates and processing times of eight jobs for example.

	Processing time						
Stage	1		2		3		
Jobs \ Plant	*A*	*B*	*A*	*B*	*A*	*B*	Due date d_j
J_1	4	5	4	2	4	2	8
J_2	8	4	5	3	5	21	9
J_3	2	3	7	4	7	14	6
J_4	6	6	6	2	6	22	10
J_5	5	8	1	5	1	5	4
J_6	5	8	8	3	8	3	7
J_7	3	1	6	1	6	5	5
J_8	7	7	7	1	7	12	8

a the-smaller-the-better metric while $\bar{X}_s > 0$. To resolve such a conflict in justification, we use $\frac{1}{CV_s} = \frac{\bar{X}_s}{\sigma_s}$ as a performance metric in the GA search. Apparently, $\frac{1}{CV_s} = \frac{\bar{X}_s}{\sigma_s}$ is a the-larger-the-better metric whatever $\bar{X}_s$ 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.

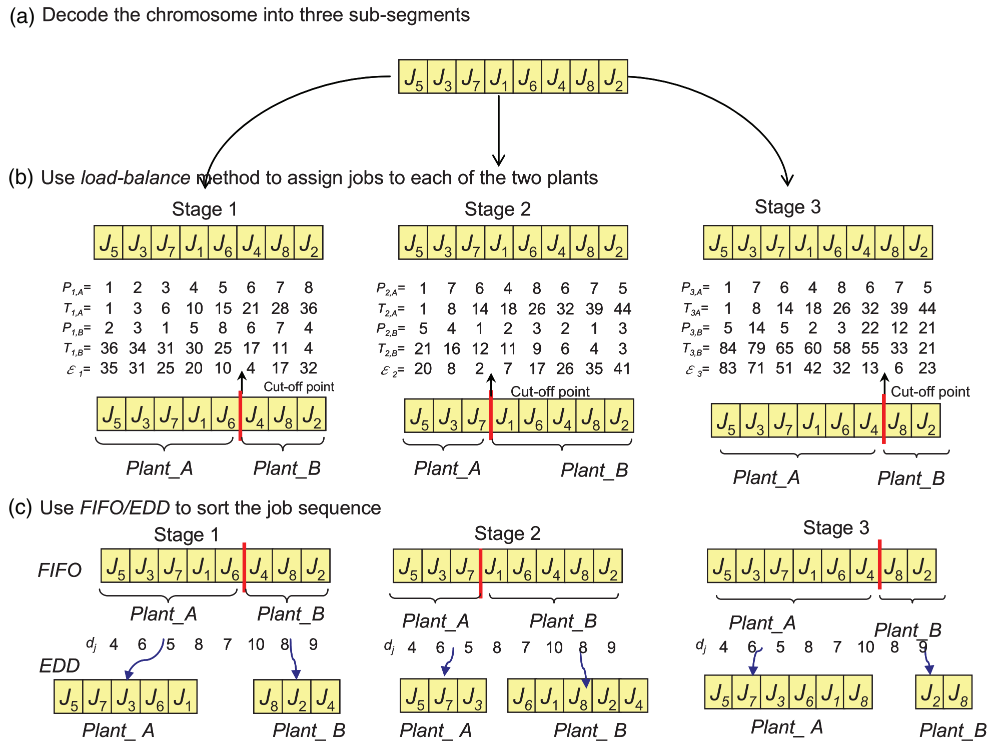
In this section, we firstly describe the chromosome representation scheme and then present the decoding method used to obtain the route assignment decision of a chromosome. Secondly, given a route assignment decision, we describe how to use a dispatching rule to determine the job sequence. Thirdly, we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



(a) Decode the chromosome into three sub-segments
(b) Use load-balance method to assign jobs to each of the two plants
Stage 1
Stage 2
Stage 3
Cut-off point
Plant_A
Plant_B
(c) Use FIFO/EDD to sort the job sequence
FIFO
EDD

Figure 2. Chromosome and decoding schemes ($p_{j,k}$: the processing time of job j required at *Plant_k*; $T_{j,k}$: the accumulated job processing times at *Plant_k* starting from the leftmost job to job j; $\varepsilon_j = |T_{j,A} - T_{j,B}|$).

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, $P_{j,A}$ 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 $\varepsilon_j = |T_{j,A} - T_{j,B}|$, and identify the job j^* which has the smallest ε_j ($j^* = J_4$ in this example, with $\varepsilon_j^* = 4$). Then, for the left-hand and right-hand neighbors of job j^*, select the job which is relatively smaller in ε_j, (J_6 is selected in this example, with $\varepsilon_j = 10$). 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.

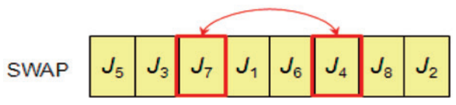


Figure 3. Mutation operator: SWAP.

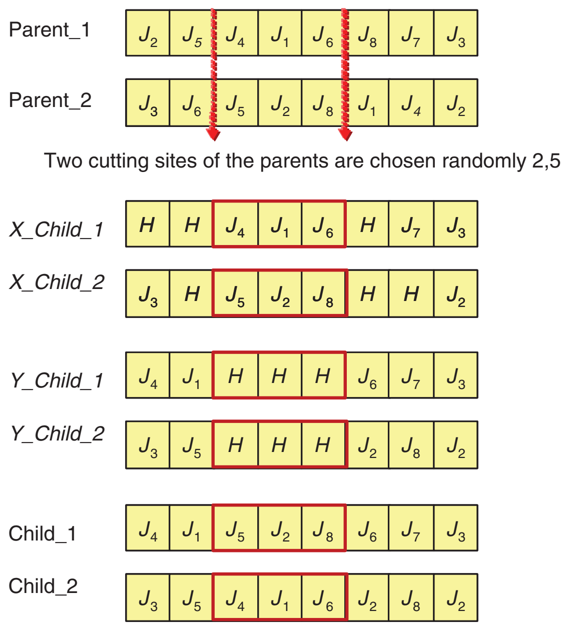

Parent_1 J_2 J_5 J_4 J_1 J_6 J_8 J_7 J_3
Parent_2 J_3 J_6 J_5 J_2 J_8 J_1 J_4 J_2
Two cutting sites of the parents are chosen randomly 2,5
X_Child_1 H H J_4 J_1 J_6 H J_7 J_3
X_Child_2 J_3 H J_5 J_2 J_8 H H J_2
Y_Child_1 J_4 J_1 H H H J_6 J_7 J_3
Y_Child_2 J_3 J_5 H H H J_2 J_8 J_2
Child_1 J_4 J_1 J_5 J_2 J_8 J_6 J_7 J_3
Child_2 J_3 J_5 J_4 J_1 J_6 J_2 J_8 J_2

Figure 4. Crossover operator: LOX.

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 J4, J1, and J6, 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 α, β and γ.

Table 2. Simplex centroid design points for a mixture experiment.

Design point	α	β	γ
1	1	0	0
2	0	1	0
3	0	0	1
4	1/2	1/2	0
5	1/2	0	1/2
6	0	1/2	1/2
7	1/3	1/3	1/3
8	2/3	1/3	1/3
9	1/3	2/3	1/3
10	1/3	1/3	2/3

5.1 *Dispatching priority indicator*

In the GA-COMBO, we define a dispatching priority indicator $p = \alpha \cdot n_{DD} + \beta \cdot n_{TCT} + \gamma \cdot n_{ST}$ $p = \alpha \cdot n_{DD} + \beta \cdot n_{TCT} + \gamma \cdot n_{ST}$ $p = \alpha \cdot n_{DD} + \beta \cdot n_{TCT} + \gamma \cdot n_{ST}$ for each job at each split-chromosome to make the dispatching decision. Herein, $\alpha, \beta,$ and γ are a set of weightings, each of which is a positive real number and $\alpha + \beta + \gamma = 1$. Moreover, n_{DD} is defined as $n_{DD} = (x_{DD} - \overline{x_{DD}})/\sigma_{DD}$ where x_{DD} represents the due date of a job, $\overline{x_{DD}}$ and σ_{DD} respectively represent the mean of and standard deviation of x_{DD} for all jobs. Accordingly, we define $n_{TT} = (x_{TT} - \overline{x_{TT}})/\sigma_{TT}$ and $n_{ST} = (x_{ST} - \overline{x_{ST}})/\sigma_{ST}$, 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 ($x_{DD}, x_{TT},$ and x_{ST}) might be 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 $x_{DD}, x_{TT},$ and x_{ST} have been normalised as dimensionless metrics $n_{DD}, n_{TT},$ and n_{ST} in order to reduce the effect of using incompatible unit. Moreover, α, β and γ 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 α, β and γ*

As stated, α, β and γ 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 α, β and γ.

In the DOE methodology, we firstly choose 10 sets of (α, β, γ) as shown in Table 2. Different choices of (α, β, γ) imply that different weightings are imposed on the three metrics. Therefore, each set of (α, β, γ) 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 $\{(\alpha_i, \beta_i, \gamma_i)|y_i\}$ $i = 1, \ldots, 10$, where y_i is the obtained performance of ith GA-COMBO program.

With the 10 data sets $\{(\alpha_i, \beta_i, \gamma_i)|y_i\}$, 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 3. Due dates and processing times of four various scenarios.

Scenarios	Due date (min.) of each job	Processing time (min.) in Stage 1 of dual flow-shop	Processing time (min.) in Stage 2 of dual flow-shop	Processing time (min.) in Stage 3 of dual flow-shop
S_1	$U(400, 600)$	$U(1, 6)$	$U(5, 105)$	$U(5, 105)$
S_2	$U(400, 420)$	$U(5, 105)$	2.741	2.741
S_3	$U(150, 250)$	$U(1, 6)$	$U(1, 6)$	$U(1, 6)$
S_4	$U(130, 150)$	$U(10, 30)$	2.741	$U(25, 30)$

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-COMOBO 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: $N = 100$, $P_c = 0.8$, $P_m = 0.2$, $T_b = 1{,}000$, $T_f = 100{,}000$, where N represents the population size, P_c 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 ($T = 10$ min, 50 min, 100 min, 200 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 (algorithms) take their average processing efficiency ratio) $\times$ 5 (job size) $\times$ 4 (transportation times) $\times$ 15 (replications).

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 $CV_{\text{GA-EDD}}$, $CV_{\text{GA-FIFO}}$, $CV_{\text{GA-SPT}}$, $CV_{\text{GA-LFO}}$, and $CV_{\text{GA-Combo}}$. To compare the five algorithms, we take GA-Combo as the benchmark, and define a solution quality indicator $\gamma_x = (CV_x - CV_{\text{GA-Combo}})/CV_x$, where the subscript x denotes an algorithm other than GA-COMBO. A positive γ_x value denotes that GA-COMBO outperforms the x algorithm; conversely, a negative γ_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 γ_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.

Scenarios	Jobs	P	$CV_{\text{GA-COMBO}}$	$CV_{\text{GA-EDD}}$	$r_{\text{GA-EDD}}$	$CV_{\text{GA-SPT}}$	$r_{\text{GA-SPT}}$	$CV_{\text{GA-FIFO}}$	$r_{\text{GA-FIFO}}$	$CV_{\text{GA-LFO}}$	$r_{\text{GA-LFO}}$	α	β	γ
S_1	20	1:1	0.890	0.890	0%	2.221	150%	4.079	359%	9.038	916%	1.000	0.000	0.000
		1:3	3.243	3.243	0%	6.577	103%	4.924	52%	4.984	54%	1.000	0.000	0.000
	40	1:1	0.781	0.781	0%	1.173	50%	1.147	47%	1.368	75%	1.000	0.000	0.000
		1:3	1.931	1.931	0%	2.611	35%	2.146	11%	2.400	24%	1.000	0.000	0.000
	60	1:1	1.190	1.190	0%	2.125	79%	2.059	73%	1.738	46%	1.000	0.000	0.000
		1:3	1.458	1.517	4%	2.952	103%	4.753	226%	1.866	28%	0.879	0.000	0.121
	80	1:1	0.932	0.932	0%	1.406	51%	2.242	141%	3.850	313%	1.000	0.000	0.000
		1:3	1.708	1.709	0%	4.577	168%	2.828	66%	1.993	17%	1.000	0.000	0.000
	100	1:1	0.907	0.907	0%	1.372	51%	1.868	106%	2.066	128%	1.000	0.000	0.000
		1:3	2.990	2.990	0%	4.461	49%	5.279	77%	3.089	3%	1.000	0.000	0.000
	Average		1.603	1.609	0%	2.947	84%	3.133	95%	3.239	102%			
S_2	20	1:1	1.185	2.807	137%	1.185	0%	2.666	125%	1.666	41%	0.000	1.000	0.000
		1:3	1.205	2.453	104%	1.205	0%	2.137	77%	1.237	3%	0.000	1.000	0.000
	40	1:1	1.086	2.862	164%	1.086	0%	2.365	118%	3.365	210%	0.000	1.000	0.000
		1:3	1.096	2.746	151%	1.096	0%	2.159	97%	4.159	280%	0.000	1.000	0.000
	60	1:1	1.052	2.843	170%	1.052	0%	2.984	184%	1.984	89%	0.000	1.000	0.000
		1:3	1.058	1.752	66%	1.058	0%	2.708	156%	1.708	61%	0.000	1.000	0.000
	80	1:1	0.457	1.039	127%	0.457	0%	1.051	130%	1.051	130%	0.000	1.000	0.000
		1:3	0.967	1.136	17%	0.966	0%	2.952	205%	1.952	102%	0.000	1.000	0.000
	100	1:1	1.043	1.697	63%	1.043	0%	2.248	116%	1.248	20%	0.000	1.000	0.000
		1:3	1.046	1.117	7%	1.046	0%	2.821	170%	1.821	74%	0.000	1.000	0.000
	Average		1.019	2.045	101%	1.019	0%	2.409	136%	2.019	98%			
S_3	20	1:1	0.432	0.560	30%	0.621	44%	0.452	5%	0.435	1%	0.000	0.000	1.000
		1:3	0.474	0.619	30%	0.793	67%	0.475	0%	0.486	3%	0.000	1.000	0.000
	40	1:1	0.333	0.411	23%	0.403	21%	0.364	9%	0.332	0%	0.000	0.141	0.859
		1:3	0.322	0.409	27%	0.360	12%	0.324	1%	0.324	1%	0.295	0.392	0.313
	60	1:1	0.454	0.573	26%	0.534	18%	0.488	8%	0.454	0%	0.000	0.000	1.000
		1:3	0.455	0.658	45%	0.526	16%	0.487	7%	0.470	3%	0.376	0.321	0.303
	80	1:1	0.533	0.668	25%	0.605	14%	0.580	9%	0.533	0%	0.000	0.000	1.000
		1:3	0.475	0.658	39%	0.910	92%	0.493	4%	0.477	1%	0.364	0.312	0.323
	100	1:1	0.506	0.625	24%	0.562	11%	0.554	9%	0.506	0%	0.000	0.000	1.000
		1:3	0.517	0.726	40%	0.994	92%	0.537	4%	0.533	3%	0.375	0.322	0.303
	Average		0.450	0.591	31%	0.631	40%	0.476	6%	0.455	1%			
S_4	20	1:1	1.167	2.071	77%	1.773	52%	1.452	24%	1.879	61%	0.288	0.424	0.288
		1:3	1.290	2.461	91%	1.922	49%	1.721	33%	2.280	77%	0.296	0.394	0.310
	40	1 : 1	1.109	1.618	46%	1.850	67%	1.314	18%	1.909	72%	0.333	0.352	0.314
		1 : 3	1.147	1.750	53%	1.907	66%	1.261	10%	1.440	26%	0.253	0.374	0.374
	60	1 : 1	1.098	1.776	62%	2.111	92%	1.292	18%	1.525	39%	0.296	0.401	0.303
		1 : 3	1.106	1.862	68%	2.094	89%	1.290	17%	1.550	40%	0.000	0.434	0.566
	80	1 : 1	1.173	1.572	34%	1.881	60%	1.267	8%	1.613	38%	0.332	0.374	0.294
		1 : 3	1.082	1.653	53%	1.901	76%	1.161	7%	1.212	12%	0.243	0.353	0.404
	100	1 : 1	1.003	1.379	37%	1.965	96%	1.208	20%	1.506	50%	0.313	0.362	0.325
		1 : 3	0.985	1.426	45%	2.003	103%	1.196	21%	1.179	20%	0.000	0.377	0.623
	Average		1.116	1.757	57%	1.941	74%	1.316	18%	1.609	44%			
Total average:			0.838	1.200	43%	1.308	56%	1.467	75%	1.465	75%			

Table 5. Average r_x for each of the four scenarios under different T.

Transportation time	$T = 10$ min	$T = 50$ min	$T = 100$ min	$T = 200$ min
Scenarios	r_x	r_x	r_x	r_x
S_1	70%	49%	48%	64%
S_2	84%	94%	104%	110%
S_3	20%	23%	20%	16%
S_4	48%	47%	54%	67%
Average:	56%	53%	56%	64%

Table 6. Single-shop routing indicator (ρ) and cross-shop routing indicator $(1 - \rho)$ under different transportation time.

Transportation time	$T = 10$ min	$T = 50$ min	$T = 100$ min	$T = 200$ min
$1 - \rho$	509	319	263	196
ρ	91	281	337	404
Total:	600	600	600	600

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 (ρ), 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, $600 = A * S * P * J * T * R = 1 * 4 * 2 * 5 * 1 * 15$. Of the 600 experiment runs, the values of ρ and $(1 - \rho)$ at various settings of transportation time are shown in Table 6. The table indicates that ρ 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 α, β, γ 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 (α, β, γ) for all instances in a particular scenario appear to be similar; and (α, β, γ) 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 α, β, γ 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

Chen, J.C., *et al.*, 2008. Price negotiation for capacity sharing in two-factory environment using genetic algorithm. *International Journal of Production Research*, 46 (7), 1847–1868.

Chiang, A.H., Chou, Y.C., and Chen, W.H., 2010. A capacity evaluation and trading model for semiconductor foundry manufacturing. *International Journal of Advanced Manufacturing Technology*, 54, 1–10.

Chiang, D., *et al*., 2007. Optimal supply chain configurations in semiconductor manufacturing. *International Journal of Production Research*, 45 (3), 631–651.

Croce, F.D., Tadei, R., and Volta, G., 1995. A genetic algorithm for the job shop problem. *Computation Operation Research*, 22 (1), 15–24.

Dabbas, R.M., *et al*., 2003. Multiple response optimisation using mixture-designed experiments and desirability functions in semiconductor scheduling. *International Journal of Production Research*, 41 (5), 939–961.

Lee, Y.H., *et al*., 2006. Supply chain model for the semiconductor industry in consideration of manufacturing characteristics. *Production Planning & Control*, 17 (5), 518–533.

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

Holland, J.H., 1975. *Adaptation in neural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

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

Minitab Inc., 2003. MINITAB manual. Available from: http://www.minitab.com [Accessed 18 June 2012].

Renna, P. and Argoneto, P., 2010. A game theoretic coordination for trading capacity in multisite factory environment. *International Journal of Advanced Manufacturing Technology*, 47 (9), 1241–1252.

Renna, P. and Argoneto, P., 2011. Capacity sharing in a network of independent factories: a cooperative game theory approach. *Robotics and Computer-Integrated Manufacturing*, 27 (2), 405–417.

Toba, H., *et al*., 2005. Dynamic load balancing among multiple fabrication lines through estimation of minimum inter-operation time. *IEEE Transactions on Semiconductor Manufacturing*, 18 (1), 202–213.

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

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

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