Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (12): 2217-2225, 2008
ISSN 1812-5654

Two-Machine Flexible Flow-Shop Scheduling with Setup Times

[1]Ming-Cheng Lo, [2]Jen-Shiang Chen and [3]Yong-Fu Chang
[1]Department of Business Administration, Ching Yun University, Taiwan, Republic of China
[2]Department of Business Administration, Far East University, Taiwan, Republic of China
[3]Department of Marketing and Logistics Management, Far East University, Taiwan, Republic of China

Abstract: This investigation studies two-machine flow-shop scheduling problems in which both machines are versatile and setup time is considered. First, a modified branch and bound algorithm for determining the optimal schedule is developed to minimize the makespan of jobs for these problems. Second, a genetic algorithm is used to rapidly find near-optimal schedules for large scale problems. Finally, computational experiments are performed to illustrate the effectiveness and efficiency of the proposed algorithms.

Key words: Scheduling, flow-shop, alternative operations, setup time, branch and bound, genetic algorithm

INTRODUCTION

A review of the literature shows a trend in industry away from grandiose schemes such as Flexible Manufacturing System (FMS) to much more focused, smaller, manageable levels of factory automation (Kumar and Shankar, 2000). Presently, because of changing market demands, manufacturing strategy has switched from high-volume production of narrow product lines to medium-volume to low-volume batches of various products. Consequently, the FMS which offers the flexibility to produce economically in lots of any size is becoming increasingly important in modern industrial settings (Pan and Chen, 1997). An FMS, according to Saygin and Kilic (1999), consists of a group of machine centers, interconnected via a set of automated material handling and in-process storage systems, which are all control by an integrated computer system. The flexibility of FMS is achieved through the use of versatile machine centers. Unlike the conventional assumption that only one of each type of machine is available, some of the machines can perform alternative operations in addition to their own primary ones.

Ahn *et al.* (1993) incorporated alternative operations into a scheduling system to increase resource utilization and reduced the makespan of manufacturing products. Liao *et al.* (1995) presented two integer programming formulations for a permutation flow-shop in which processors are flexible to perform other operations besides their own. Gere (1996) investigated several heuristic algorithms for traditional job-shop scheduling and concluded that alternative operations improve productivity. They showed that the production requirements can be completed earlier by employing alternative machines. Jeong and Kim (1998) studied a real-time scheduling methodology which uses simulation and dispatching rules for flexible manufacturing systems. They developed a scheduling mechanism in which job dispatching rules vary dynamically based on information from discrete event simulation that is used for evaluating candidate dispatching rules. Pan and Chen (1997) studied the problem of scheduling a set of jobs each of which consists of two consecutive operations. The jobs are processed in a two-machine flow-shop in which either or both machines are versatile and the processing times of the operations of each job are independent of job sequence. The objective was to minimize the makespan. They showed that this scheduling problem is NP-complete and developed a branch and bound algorithm to solve it. Guerrero *et al.* (1999) examined the influence of alternative operations on FMS performance. They applied a linear programming model to prescribe production plans and employed adaptive control mechanisms to implement these plans. Shanker and Modi (1999) proposed a branch and bound technique to determine an inter-dependent multiple-product resource-constrained scheduling problem with the objective of makespan minimization in a flexible manufacturing system with resource flexibility. For the same problems, Cheng and Wang (1998) provided a general pseudo-polynomial dynamic programming scheme which solves the problems analytically.

The earlier studied consider scheduling with alternative operations, but do not include machine setup times. Schaller *et al.* (2000) considered the problem of scheduling part families and jobs within each part family in a flow line manufacturing cell where the setup times for

Corresponding Author: Ming-Cheng Lo, Department of Business Administration, Ching Yun University, 229, Chien-Hsin Road, Jung-Li, 320, Taiwan, Republic of China

each family were sequence-dependent and it was desired to minimize the makespan while processing parts (jobs) in each family together. Rajendran and Ziegler (2003) presented efficient heuristics for scheduling jobs in a static flow-shop with sequence-dependent setup times of jobs. The objective is to minimize the sum of weighted flow time and weighted tardiness of jobs. Kurz and Askin (2004) examined scheduling in flexible flow lines with sequence-dependent setup times to minimize makespan. Ruiz *et al.* (2005) dealt with the permutation flow-shop scheduling problem in which there were sequence-dependent setup times on each machine and the optimization criterion considered was the minimization of the makespan. Chen and Pan (2005) developed the development of a mixed Binary Integer Programming (BIP) model for scheduling alternative operations in two-machine flow-shop problems with mean tardiness as the criterion.

The literature considering setup time on scheduling manufacturing systems with alternative operations is scarce. Lee and Mirchandani (1988) were the first to study a two-machine flexible flow-shop problem and showed that the problem could be reduced to three versions of zero-setup, one-setup and two-setup problems. The zero-setup problem meant that there was no operation shift on both machines, the one-setup problem meant that there was just one operation shift on one of the machines and the two-setup problem meant that there were two operation shifts with one on each machine. Cheng and Wang (1999) derived a worst-case error bound for the heuristic presented by Lee and Mirchandani (1988) for the NP-complete one-setup version of the two-machine flexible manufacturing cell scheduling problem and proposed another heuristic with a worst-case error bound of 3/2.

This investigation examines the scheduling in a two-machine flow-shop in which both machines are versatile and where setup time is considered such that alternative operations and machine setups can take place. This study focuses on the two-setup problem presented by Lee and Mirchandani (1988). A modified branch and bound algorithm based on the Pan and Chen's (1997) branch and bound technique is developed to find the optimal schedule. A genetic algorithm is used to rapidly find the near-optimal schedule for larger problems. Computational experiments are conducted to illustrate the effectiveness and efficiency of these algorithms.

PROBLEM DESCRIPTION

A scheduling problem can be described in terms of its job characteristics, its shop characteristics and the optimality criterion with which the evaluation of each schedule can be made (Pinedo, 2002). The specific two-machine flexible flow-shop that is considered in this study is described below:

Job characteristics

- Each job J_i has two operations, X and Y.
- Each job requires the processing of firstly operation type X and then operation type Y.
- The processing time of the operations of each job are known and fixed.
- The operations are not pre-emptable.
- All jobs are immediately available for processing once production begins.

Shop characteristics

- The shop consists of two machines, M_1 and M_2.
- Each machine can perform both operation types. M_1 and or M_2 may be versatile.
- Each machine can perform only one operation at a time.
- The setup operation, S_{XY} (S_{YX}), changes the tool-set for operation type X (Y) with the tool-set for operation Y (X). Both setup operations are assumed to take the same amount of time to perform.
- A setup operation must be performed when the machine needs to perform the other operation type.
- The transfer or transport time of a job from one machine to the other is negligible.

Optimality criterion

- The objective is to schedule the operations and setups so as to minimize the makespan.

The plan for processing these n jobs requires two distinct operations performed in the order operation type X followed by operation type Y, where M_1 has operation type X as its primary operation and M_2 has operation type Y as its primary operation. Therefore, all the jobs are loaded and processed in the sequence M_1 followed by M_2, if no alternative operations occur. Although alternative operations may suffer from efficiency penalties, they can be used to improve machine utilization and system performance when one machine is overloaded and the alternative machine is idle.

Let p_{i1} and p_{i2} denote the processing times of X of J_i on machines M_1 and M_2, while q_{i1} and q_{i2} are the processing times of Y of J_i on M_1 and M_2, respectively. For two-machine flow-shop scheduling with alternative operations, Pan and Chen (1997) investigated the following three problems:

- The $M_1 \neg M_2$ problem. In this problem, M_1 is a versatile machine and M_2 is dedicated to operation Y and thus $p_{[i]2} = \infty$, i = 1, …, n.
- The $M_2 \neg M_1$ problem. In this problem, M_1 is dedicated to operation X and M_2 is a versatile machine and thus $q_{[i]1} = \infty$, i = 1, …, n.
- The $M_1 \leftrightarrow M_2$ problem. In this problem, both M_1 and M_2 are versatile.

This study builds upon the research of Pan and Chen (1997) and focuses on the $M_1 \leftrightarrow M_2$ problem. This study also considers setup time while the machine needs to perform the other operation type. The objective is to sequence a set of jobs on the two machines to minimize the maximum completion time, or makespan.

Modified branch and bound algorithm: Pan and Chen (1997) presented a branch and bound algorithm to solve two-machine flexible scheduling problems. However, they neglected the machine setup times. This study considers setup times and extends the branch and bound approach of Pan and Chen (1997).

Undoubtedly, for any feasible schedule for the two-machine flexible flow-shop with setup times problem, all jobs can be divided into four disjoint sets E_{uv}, where u = 1, 2 and v = 1, 2, such that the X and Y operations of all jobs in E_{uv} are scheduled on machines M_u and M_v, respectively. Initially, all jobs are set to belong to E_{12} and the optimal schedule is identified using Johnson's rule (Johnson 1954). If the operations of the jobs belong to E_{11} or E_{22} or E_{21}, they are called alternative operations. Pan and Chen (1997) proved this problem belongs to the class of difficult problems known as NP-complete. This study considers the scheduling problem of minimizing makespan in a two-machine flexible flow-shop with setup times. Clearly, the proposed problem is NP-complete since the problem of scheduling alternative operations in two-machine flow-shops is NP-complete (Pan and Chen, 1997).

Lemma 1: For the two-machine flexible flow-shop with the setup time scheduling problem, an optimal schedule exists with at most two setups, of which at most one is in each machine and both are E_{XY} type.

Proof: The proof is straightforward.

Lemma 2: For the two-machine flexible flow-shop with the setup time scheduling problem, an optimal schedule s* exists in which there is no idle time before any operations assigned to at least one of the machines.

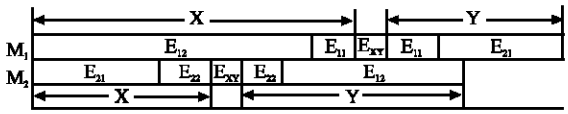

Fig. 1: A feasible schedule

Proof: Similar to the proof of Lemma 3 by Cheng and Wang (1998).

According to Lemmas 1 and 2, E_{XY} and all jobs in each set E_{uv} (u, v = 1, 2) can be scheduled as shown in Fig. 1.

Lemma 3: If alternative operations are known, the two-machine flexible flow-shop with the setup time scheduling problem can be solved in polynomial time.

Proof: In situations involving predefined alternative operations, the machine setup(s) and machine processes for each operation are known and fixed, so the two-machine flexible flow-shop with the setup time scheduling problem reduces to $n/2/G/C_{max}$ problem, which can be solved in polynomial time by Johnson's rule.

Let $J_{[i]}$ denote the job scheduled at the ith position in a particular processing sequence. Moreover, let $p_{[i]1}$ and $p_{[i]2}$ be the processing times of X of $J_{[i]}$ on machines M_1 and M_2 and let $q_{[i]1}$ and $q_{[i]2}$ represent the processing times of Y of $J_{[i]}$ on machines M_1 and M_2, respectively. According to Lemma 3, the C_{max} of the two-machine flexible flow-shop with the setup time scheduling problem can be calculated as follows.

Procedure for calculating C_{max}

Step 1: Set $TM_1 = TM_2 = 0$, TM_1 and TM_2 as the current time when M_1 and M_2 finish the job processing, respectively.
Set $CE_{12}[\cdot] = CE_{21}[\cdot] = 0$, $CE_{12}[\cdot]$ and $CE_{21}[\cdot]$ as the current time when M_1 and M_2 finish processing the operations belonging to E_{12} and E_{21}, respectively.
Set NE_{uv} as the number of jobs belonging to type E_{uv}, where u = 1, 2 and v = 1, 2.

Step 2: If $NE_{12} \neq 0$, according to Johnson's rule, we obtain optimal schedule $(J_{[1]}, J_{[2]}, \ldots, J_{[NE_{12}]})$.
For $(i = 1; i \leq NE_{12}; i{+}{+})$ $\{TM_1 = TM_1 + p_{[i]1}; CE_{12}[J_{[i]}] = TM_1\}$.

Step 3: If $NE_{11} \neq 0$, the optimal schedule is arbitrarily, e.g., $(J_{[1]}, J_{[2]}, \ldots, J_{[NE_{11}]})$. For $(i = 1; i \leq NE_{11}; i{+}{+})$ $\{TM_1 = TM_1 + p_{[i]1}\}$.

Step 4: If $NE_{21} \neq 0$, according to Johnson's rule, we obtain optimal schedule $(J_{[1]}, J_{[2]}, \ldots, J_{[NE_{21}]})$.
For $(i = 1; i \leq NE_{21}; i{+}{+})$ $\{TM_2 = TM_2 + p_{[i]2}; CE_{21}[J_{[i]}] = TM_2\}$.

Step 5: If $NE_{22} \neq 0$, the optimal schedule is arbitrarily, e.g., $(J_{[1]}, J_{[2]}, \ldots, J_{[NE_{22}]})$.
For $(i = 1; i \leq NE_{22}; i++)$ $\{TM_2 = TM_2 + p_{[i]2}\}$.

Step 6: Let t denote the setup time. If ($NE_{21} \neq 0$ or $NE_{22} \neq 0$), $TM_2 = TM_2 + t$.

Step 7: If ($NE_{11} \neq 0$ or $NE_{21} \neq 0$), $TM_1 = TM_1 + t$.

Step 8: If $NE_{11} \neq 0$, calculate TM_1 according to the schedule as step 3.
For $(i = 1; i \leq NE_{11}; i++)$ $\{TM_1 = TM_1 + q_{[i]1}\}$.

Step 9: If $NE_{21} \neq 0$, calculate TM_1 according to the schedule as step 4.
For $(i = 1; i \leq NE_{21}; i++)$ $\{TM_1 = \max(TM_1, CE_{21}[J_{[i]}]) + q_{[i]1}\}$.

Step 10: If $NE_{22} \neq 0$, calculate TM_2 according to the schedule as step 5.
For $(i = 1; i \leq NE_{22}; i++)$ $\{TM_2 = TM_2 + q_{[i]2};\}$.

Step 11: If $NE_{12} \neq 0$, calculate TM_1 according to the schedule as step 2.
For $(i = 1; i \leq NE_{12}; i++)$ $\{TM_2 = \max(TM_2, CE_{12}[J_{[i]}]) + q_{[i]2}\}$.

Step 12: Set $C_{max} = \max(TM_1, TM_2)$.

An illustrative example: To illustrate Lemma 3 and the procedure for calculating C_{max}, consider the five-job problem shown in Table 1. First, all jobs are set to belong to E_{12}. The initial schedule is obtained by Johnson's rule as $(J_1, J_4, J_3, J_5, J_2)$ with $C_{max} = 47$. Finally, we set $\{J_3, J_5\} \in E_{12}$, $J_2 \in E_{11}$, $J_1 \in E_{22}$ and $J_4 \in E_{21}$, then according to Lemma 3 and the procedure for calculating C_{max}, we obtain new $C_{max} = 39$. The Gantt chart of the example problem is shown in Fig. 2.

For any given schedule of the two-machine flexible flow-shop with n jobs, the total number of possible sequences when alternative operations are considered is 2^{2n}. If no alternative operations are performed, an optimal solution can be obtained by applying Johnson's rule for the $n/2/G/C_{max}$ problem. Consequently, this solution can be used as an initial feasible schedule and is designated by S_0. Figure 3 shows a typical branching tree with 16 nodes for scheduling the two-machine flexible flow-shop with two jobs. These nodes correspond to the 16 possible sequences for a given initial schedule when both machines are versatile. Node 0, representing the initial solution, is the initial node and the remaining nodes are numbered in the order they are produced by the following branching tree generation procedure. Let node s be a node in the branching tree and let $\bar{s}$ denote the unique parent node of s. Moreover, define $o_{[i]j}$ as the operation number j of job $J_{[i]}$, where i = 1, 2,. .., n and j = l, 2. At node s, let A_s represent the job whose second operation is an alternative operation while its first operation is not; and let B_s denote the job whose first operation is an alternative operation and whose second operation is not. Similarly, C_s denotes the job of which both operations are processed by alternative machines. Finally, D_s is the job in which both operations are performed by their respective primary machines. Clearly, A_s, B_s, C_s and D_s are mutually exclusive and together they constitute the entire alternative operation type associated with the job. The job number k_x^s in Fig. 3 shows that node s has alternative operation type x of job k_x^s, where x = 1 denotes

Table 1: Data of the example problem

	Operation type X	Operation type Y
Job i	p_{i1}/p_{i2} (M_1/M_2)	q_{i1}/q_{i2} (M_1/M_2)
J_1	3/4	10/13
J_2	8/11	6/7
J_3	5/6	9/12
J_4	3/4	9/10
J_5	6/9	10/15

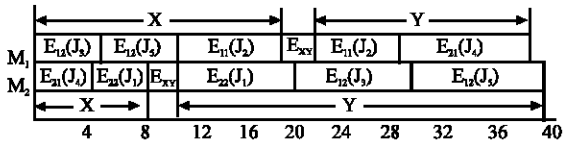

Fig. 2: Gantt chart of the example problem

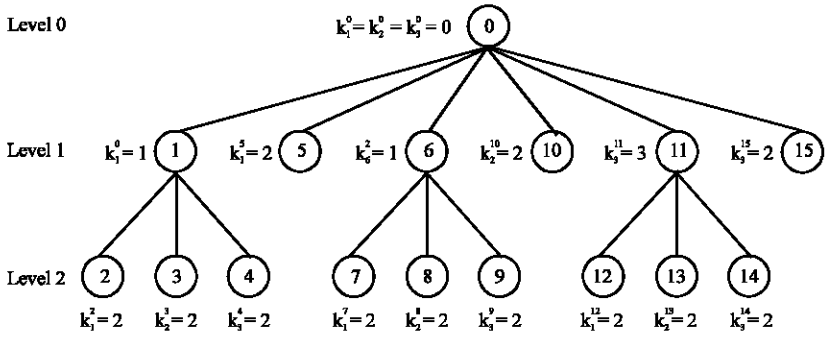

Fig. 3: Branching tree of an M1↔M2 problem

the job belongs to A_s, x = 2 represents that the job belongs to B_s and x = 3 denotes that the job belongs to C_s.

Let S_s denote the schedule represented by node s. Node s depicts the introduction of an alternative operation for $o_{[i]1}$ and $o_{[i]2}$ into the schedule represented by node $\bar{s}$. For example, node 0 represents the initial schedule S_0 which contains no alternative operations. Moreover, node l indicates that $o_{[1]2}$ in S_0 is an alternative operation performed by M_1. Descending from $o_{[1]2}$, node 2 includes alternative operation for $o_{[2]2}$ in the schedule since its parent is node l in the branching tree and so on. Therefore, given S_0, the set of all the alternative operations performed at node s is the collection of all the k_x^s values by tracing node s upward until node 0 is reached in Fig. 3 and its schedule is determined by Johnson's rule due to Lemma 3.

The branching tree in Fig. 3 can be produced by the following procedure.

The branching tree generation procedure

Step 1 : Set child node s = 0, parent node $\bar{S}$ = 0, level L = 0 and job k_x^s = 0.
Step 2 : Set x = 1 to denote the jobs in A_s.
Step 3 : Increase L by 1. Set $\bar{s} = s$, s = s+1 and generate a child of $\bar{S}$, node s. Set $k_x^s = k_x^{\bar{s}} + 1$.
Step 4 : If k_x^s = n, go to step 5. Otherwise, go to step 3.
Step 5 : Set x = 2 to denote the jobs in B_s.
Step 6 : Set s = s+1, = $k_x^s = k_x^{\bar{s}} + 1$.
Step 7 : If k_x^s = n, go to step 9.
Step 8 : Increase L by 1. Set $\bar{S}$ = s, s = s+1 and generate a child of , node s. Set $k_x^s = k_x^{\bar{s}} + 1$. Go to step 7.
Step 9 : Set x = 3 to denote the jobs in C_s.
Step 10 : Set s = s + 1, $k_x^s = k_x^{\bar{s}} + 1$.
Step 11 : If k_x^s = n, go to step 13.
Step 12 : Increase L by 1. Set $\bar{S}$ = s, s = s+1 and generate a child of $\bar{S}$, node s. Set $k_x^s = k_x^{\bar{s}} + 1$. Go to step 11.
Step 13 : If L = 1, stop. Otherwise, go to step 14.
Step 14 : Decrease L by 1, set s = s+1 and generate node s. Set x = 1 and $k_x^s = k_x^{\bar{s}} + 1$. Go to step 4.

Define I_{s1} and I_{s2} as the idle times of M_1 and M_2 in schedule S_s, respectively. In S_s, we then have

$$I_{s1} = C_{max}(s) - \sum_{J_i \in D_s} p_{i1} - \sum_{J_i \in A_s} (p_{i1} + q_{i1}) - y - \sum_{J_i \in C_s} q_{i1}$$

Where:

$$y = \begin{cases} t & \text{if } (A_s \text{ or } C_s) \neq \varnothing, \varnothing \text{ denotes the null set} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

and

$$I_{s2} = C_{max}(s) - \sum_{J_i \in C_s} p_{i2} - \sum_{J_i \in B_s} (p_{i2} + q_{i2}) - z - \sum_{J_i \in D_s} q_{i2}$$

Where:

$$z = \begin{cases} t & \text{if } (B_s \text{ or } D_s) \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Define T_{sq} as the time when M_q finishes the processing of all he jobs that are assigned to it in S_s, where q = 1, 2. Moreover, let I'_{sq} denote the idle time of M_q based on its T_{sq}.
Then

$$I'_{s1} = T_{s1} - \sum_{J_i \in D_s} p_{i1} - \sum_{J_i \in A_s} (p_{i1} + q_{i1}) - y - \sum_{J_i \in C_s} q_{i1}$$

Where:

$$y = \begin{cases} t & \text{if } (A_s \text{ or } C_s) \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

and

$$I'_{s2} = T_{s2} - \sum_{J_i \in C_s} p_{i2} - \sum_{J_i \in B_s} (p_{i2} + q_{i2}) - z - \sum_{J_i \in D_s} q_{i2}$$

Where:

$$y = \begin{cases} t & \text{if } (B_s \text{ or } D_s) \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The 2^{2n} number of possible sequences is not the only contributor to the complexity of the problem. The idle times can be used as a basis for fathoming dominated sequences. For $M_1 \leftrightarrow M_2$ with setup time problems, an increase in the C_{max} value at node s does not mean that node s can be fathomed. A more alternative operation down the branching tree may reduce the C_{max}. Consequently, only the nodes at the bottom level of the branching tree can be fathomed.

Lemma 4: Consider node s which includes the alternative operation for Y of $J_{[n]} \in A_s$ in the schedule. Node s is fathomed by its parent node, $\bar{S}$, if any of the following two conditions hold.

- $(A_{\bar{s}}$ or $C_{\bar{s}}) \neq \varnothing$ and $q_{[n]1} \geq I_{\bar{s}1}$.
- $(A_{\bar{s}}$ or $C_{\bar{s}}) = \varnothing$ and $(q_{[n]1}+t) \geq I_{\bar{s}1}$.

Proof: Based on the definition of $C_{max}(\bar{S})$ and $C_{max}(s)$, the following two situations exist.

- If $(A_{\bar{s}}$ or $C_{\bar{s}}) \neq \varnothing$, then $C_{max}(s) - C_{max}(\bar{S}) = I_{s1} - I_{\bar{s}1} + q_{[n]1} \geq 0$. Since $I_{s1} \geq 0$ and $q_{[n]1} \geq I_{\bar{s}1}$. Hence, the total

production time $C_{max}(s)$ is not decreased if the alternative operation occurs.

- If $(C_{\bar{s}}$ or $D_{\bar{s}}) = \emptyset$, then $C_{max}(s) - C_{max}(\bar{S}) = I_{s1} - I_{\bar{s}1} + (q_{[n]1} + t) \geq 0$. Since $I_{s1} \geq 0$ and $(q_{[n]1} + t) \geq I_{\bar{s}1}$. Hence, the maximum completion time $C_{max}(s)$ is not decreased if the alternative operation occurs.

Lemma 5: Consider node s which includes the alternative operation for X of $J_{[n]} \in B_s$ in the schedule. Node s is fathomed by its parent node, $\bar{S}$, if any of the following two conditions hold.

- $(B_{\bar{s}}$ or $D_{\bar{s}}) \neq \emptyset$ and $p_{[n]2} \geq I_{\bar{s}2}$.
- $(B_{\bar{s}}$ or $D_{\bar{s}}) = \emptyset$ and $(p_{[n]2} + t)\ I_{\bar{s}2}$.

Proof: Similar to the proof of Lemma 4.

Lemma 6: Consider node s which includes the alternative operation of $J_{[n]} \in C_s$ in the schedule. Node s is fathomed by its parent node, $\bar{S}$, if both the following conditions hold.

- Either $(A_{\bar{s}}$ or $C_{\bar{s}}) \neq \emptyset$ and $q_{[n]1} - p_{[n]1} \geq I'_{\bar{s}1}$, or $(A_{\bar{s}}$ or $C_{\bar{s}}) = \emptyset$ and $q_{[n]1} - p_{[n]1} + t \geq I'_{\bar{s}1}$.
- Either $(B_{\bar{s}}$ or $D_{\bar{s}}) \neq \emptyset$ and $p_{[n]2} - q_{[n]2} \geq I_{\bar{s}2}$, or $(B_{\bar{s}}$ or $D_{\bar{s}}) = \emptyset$ and $p_{[n]2} - q_{[n]2} + t \geq I'_{\bar{s}1}$.

Proof : For any schedule S_s of node s, there exist

- $T_{s1} = I'_{s1} + \sum_{J_i \in D_s} p_{i1} + \sum_{J_i \in A_s} (p_{i1} + q_{i1}) + y + \sum_{J_i \in C_s} q_{i1}$

Where:

$$y = \begin{cases} t & \text{if } (A_s \text{ or } C_s) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

- $T_{s2} = I'_{s2} + \sum_{J_i \in C_s} p_{i2} + \sum_{J_i \in B_s} (p_{i2} + q_{i2}) + z + \sum_{J_i \in D_s} q_{i2}$

Where:

$$z = \begin{cases} t & \text{if } (B_s \text{ or } D_s) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Suppose both operations of $J_{[n]}$ are processed by alternative machines. From the definition of T_{s1}, T_{s2}, $T_{\bar{s}1}$ and $T_{\bar{s}2}$ the following two situations exist.

- Since $q_{[n]1} - p_{[n]1} \geq I'_{\bar{s}1}$ if $(A_{\bar{s}}$ or $C_{\bar{s}}) \neq \emptyset$ and $q_{[n]1} - p_{[n]1} + t \geq I'_{\bar{s}1}$ if $(A_{\bar{s}}$ or $C_{\bar{s}}) = \emptyset$, then $T_{s1} \geq T_{\bar{s}1}$.
- Since $p_{[n]2} - q_{[n]2} \geq I'_{\bar{s}2}$ if $(B_{\bar{s}}$ or $D_{\bar{s}}) \neq \emptyset$ and $p_{[n]2} - q_{[n]2} + t \geq I'_{\bar{s}2}$ if $(B_{\bar{s}}$ or $D_{\bar{s}}) = \emptyset$, then $T_{s2} \geq T_{\bar{s}2}$.

Hence, $C_{max}(s) - C_{max}(\bar{S}) = \max(T_{s1}, T_{s2}) - \max(T_{s1}, T_{s2}) \geq 0$. Thus, the introduction of the alternative operations for both $o_{[n]1}$ and $o_{[n]2}$ does not decrease the maximum completion time.

Modified branch and bound algorithm

Step 1: Set s = 0 and set all jobs to belong to D_s. Apply Johnson's rule for the $n/2/F/C_{max}$ problem to find an initial schedule without considering any alternative operations. Calculate the idle time I_{s1}, I_{s2}, I'_{s1} and I'_{s2}

Step 2: Generate a new node s by the branching tree generation procedure. If no such node can be generated, go to step 4. Otherwise, go to step 3.

Step 3: Apply Lemmas 4-6 to node s. If node s can be fathomed, go to step 2. Otherwise, find makespan according to the "procedure for calculating C_{max}". Calculate the idle time and go to step 2.

Step 4: Stop.

A genetic algorithm approach: Previous researchers have assumed that a genetic algorithm contains the following main components: solution representation, initial population and population size, fitness of the members in a population, selection of parents, genetic operators and a termination criterion (Chen *et al*., 1995). The following discusses the ingredients of a genetic algorithm based heuristic for a two-machine flexible flow-shop scheduling problem with consideration of setup time.

Solution representation: For a scheduling problem, a structure can be easily described as a sequence of the jobs in the problem. However, in a two-machine flexible flow-shop scheduling problem, the structure must be slightly modified to display the operation's alternative situation. The structure can be described as a disjointed job set (e.g., E_{12}, E_{11}, E_{22} and E_{21}).

Generation of initial population and population size: The efficiency of genetic algorithms can be markedly increased by selecting a good initial population and reasonable population size. This heuristic sets the population size to equal n. The initial population uses the n schedules produced by method of each schedule has only one job with an alternative operation. For $M_1 \leftrightarrow M_2$ with the setup problem, the alternative operation type can be generated randomly by selecting alternative machines M_1, or M_2, or both M_1 and M_2. For instance, if we consider a six-job problem, then the six schedules in the initial population

are 100000, 010000, 001000, 000100, 000010 and 000001. The schedule 100000, denotes $J_{[i]} \in E_{11}$ and the other jobs $J_{[i]} \notin E_{11}$, where i = 2, 3, …, 6. Moreover, the digit 0 indicates one job that has no alternative operations, i.e., the job belongs to E_{12}. The digit 1 denotes that one job has alternative operation(s) and that its (their) type may randomly select one of M_1, M_2, or both M_1 and M_2, i.e., the job may randomly select one of E_{11}, E_{22}, or E_{21}.

Fitness function and the selection of parents: For a maximization problem, the measure of performance generally constitutes the fitness function. However, the objective of scheduling problems is usually to minimize a certain measure of performance. For minimization problems, the method of determining the fitness function differs slightly from that for maximization problems. This study now discusses the method used to determine the fitness function for two-machine flexible flow-shop problems.

The makespan for all the members in a population is calculated using the Procedure for calculating C_{max}. From this the largest makespan is determined and is denoted as C_{max}. The deviation of the makespan of each member in the population from the C_{max} is the fitness value of that particular member. This procedure ensures a high probability of selection for a schedule with lower makespan. This is also the criterion used for the selection of parents for the reproduction of children.

Genetic operators: Genetic algorithms contain two common genetic operators: crossover and mutation. A crossover operator involves combining the elements from two parent structures into one or more child structures. Mutation typically works with a single structure leaving the parent intact with the population. Goldberg's (1989) Partially Mapped Crossover (PMX) operator, edge recombination operator, subtour-swap operator, subtour-chunk operator, subtour-replace operator and weighted chunking operator are some of the popular crossover operators for scheduling problems (Chen *et al*., 1995). This heuristic uses Goldberg's (1989) PMX operator for the purpose of crossover.

Termination criterion: The proposed heuristic uses the number of generations as the termination criterion in our heuristic. Again, based on the trial examples, we found that the solutions become stable after twenty generations. Therefore, twenty generations is used as the termination criterion in the proposed heuristic.

Genetic algorithm based heuristic: We are now at the position to present the genetic algorithm based heuristic for the two-machine flexible flow-shop problem. The following notation will be used in describing the heuristic:

G(r)	:	The population in the r-th generation.
$g_i(r)$	:	The i-th member in G(r).
$C(g_i(r))$	:	The makespan of $g_i(r)$.
$C_{max}(r)$	:	The max $\{C(g_i(r))\}$, for all $g_i(r) \in G(r)$.
$f(g_i(r))$	:	The fitness of $g_i(r)$, which equals $C_{max}(r)$-$C(g_i(r))$.
SUMFIT(r)	:	The sum of $f(g_i(r))$, for all $g_i(r) \in G(r)$.

Procedure for implementing genetic algorithm

Step 1: Determine the initial population, G(r), where r = 0. The size of the population, POPSIZE, is n and the number of generations considered, GENER, is twenty.

Step 2: Calculate the fitness value of each member, $f(g_i(r))$, for population, G(r).

Step 3: Calculate the selection probability for each $g_i(r)$, where the selection probability is defined as:

$$P(g_i(r)) = \frac{f(g_i(r))}{SUMFIT(r)}$$

Step 4: Select a pair of members (parents) that will be used for reproduction via the selection probability.

Step 5: Apply the PMX operator to the parents. Replace the parents with the resulting offspring to form a new population, G(r + 1), for generation r + 1. If the size of the new population equals the POPSIZE, then go to Step 6, else go to Step 4.

Step 6: If current generation, r+1, equals GENER, then stop, otherwise go to step 2.

NUMERICAL INVESTIGATIONS

Computational experiments were conducted to test the effectiveness of the proposed modified branch and bound, as well as the efficiency of the genetic algorithms. The processing times of operations performed by respective primary machines were generated from a uniform distribution of integers in [1, 100]. Meanwhile, the processing times of operations performed by respective alternative machines are v times those by the primary machines, where v = 1.2, 1.4 and 1.6. All of the alternative processing times were rounded to the nearest integers.

The algorithms were tested over three different problem sizes, n = 20, 25 and 30. Twenty replications were randomly generated for each problem size. A total of 180 problems were thus tested. The computer programs were coded in Visual C++ language and run on an Intel P4/2.67GHz with 512 M SDRAM. Table shows the average CPU time (sec) of branch and bound algorithm (B and B) and Genetic Algorithm (GA), as well as the average percentage error of GA for each 20 replications.

Table 2: Computational results for B and B and GA model

		Avg. CPU time (sec)		
v†	n	B and B	GA	Avg. percentage error of GA (%)
1.2	20	100.39	28.28	4.87
	25	1031.51	70.57	4.37
	30	12201.28	227.12	3.52
1.4	20	95.57	21.79	4.43
	25	996.45	66.35	4.18
	30	10201.25	208.64	3.24
1.6	20	91.41	15.70	53.93
	25	914.44	60.34	3.21
	30	9801.51	197.61	2.98

†The processing times of operations by respective alternative machines are v times those by the primary machines

The effectiveness of the GA is measured by the percentage error, which is defined as:

$$\text{Error } (\%) = \frac{C_{max}(\text{GA}) - C_{max}(\text{B and B})}{C_{max}(\text{B and B})} \times 100\%$$

where, C_{max}(GA) denotes the makespan obtained by the GA and C_{max}(B and B) represents the optimal makespan. Table 2 yields the following observations.

- The modified branch and bound algorithm can solve optimal schedule with 30 jobs in a reasonable time. The time required to find an optimal schedule increased nearly ten times for every increment of five jobs in the problem size. The Branch and bound algorithm can fathom a majority of the nodes, but the total number of nodes increases at an exponential rate.
- The genetic algorithm is efficient and the average percentage error is less than 5%.
- The magnitude of v decides the average CPU time. That is, the larger the value of v the less the average CPU time.
- The average percentage error of GA decreases with increasing v.

CONCLUSIONS

This study considers a scheduling problem in a two-machine flow-shop in which both machines are versatile and setup time is considered. A modified branch and bound algorithm is developed to minimize the makespan of jobs for these problems. The genetic algorithm is also used to rapidly find near-optimal schedules for large scale problems. The modified branch and bound algorithm and genetic algorithm were tested over three different problem sizes, n = 20, 25 and 30, for the two-machine flexible flow-shop scheduling with setup times. For the modified branch and bound algorithm, the computation time increases at an exponential rate as the problem scale grows. The genetic algorithm is effective and that the average percentage error is smaller than 5%.

Future research may address problems under different shop environments, including flow-shop and job-shop. Problems with other performance measures, such as mean flow time or mean tardiness, may also be considered.

REFERENCES

Ahn, J., W. He and A. Kusiak, 1993. Scheduling with alternative operations. IEEE Trans. Robotic Automation, 9: 297-303.

Chen, C.L., V.S. Vempatei and N. Aljaber, 1995. An application of genetic algorithms for flow shop problems. Eur. J. Operat. Res., 80: 389-396.

Chen, J.S. and J.C.H. Pan, 2005. Minimising mean tardiness with alternative operations in two-machine flow-shop scheduling. Int. J. Syst. Sci., 36: 757-766.

Cheng, T.C.E. and G. Wang, 1998. A note on scheduling alternative operations in two-machine flowshops. J. Operat. Res. Soc., 49: 670-673.

Cheng, T.C.E. and G. Wang, 1999. A note on scheduling the two-machine flexible flowshop. IEEE Trans. Robotic Automation, 15: 187-190.

Gere, W.S., 1996. Heuristics in job shop scheduling. Manage. Sci., 13: 167-190.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1st Edn. Addison-Wesley, Canada, .

Guerrero, F., S. Lozano, T. Koltai and J. Larran~eta, 1999. Machine loading and part type selection in flexible manufacturing systems. Int. J. Prod. Res., 37: 1303-1317.

Jeong, K.C. and Y.D. Kim, 1998. A real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules. Int. J. Prod. Res., 36: 2609-2626.

Johnson, S.M., 1954. Optimal two-and three-stage production schedules with setup times included. Naval Res. Logistics Q., 1: 61-68.

Kumar, N. and K. Shanker, 2000. A genetic algorithm for FMS part type selection and machine loading. Int. J. Prod. Res., 38: 3861-3887.

Kurz, M.E. and R.G. Askin, 2004. Scheduling flexible flow lines with sequence-dependent setup times. Eur. J. Operat. Res., 159: 66-82.

Lee, E.J. and P.B. Mirchandani, 1988. Concurrent routing, sequencing and setups for a two-machine flexible manufacturing cell. IEEE J. Robotic Automation, 4: 256-264.

Liao, C., C. Sun and W. You, 1995. Flow-shop scheduling with flexible processors. Comput. Operat. Res., 22: 297-306.

Pan, C.H. and J.S. Chen, 1997. Scheduling alternative operations in two-machine flow-shops. J. Operat. Res. Soc., 48: 533-540.

Pinedo, M., 2002. Scheduling: Theory, Algorithms and Systems. 1st Edn. Prentice Hall, New Jersey .

Rajendran, C. and H. Ziegler, 2003. Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. Eur. J. Operat. Res., 149: 513-522.

Ruiz, R., C. Maroto and J. Alcaraz, 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur. J. Operat. Res., 165: 34-54.

Saygin, C. and S.E. Kilic, 1999. Integrating flexible process plans with scheduling in flexible manufacturing systems. Int. J. Adv. Manuf. Tech., 15: 268-280.

Schaller, J.E., J.N.D. Gupta and A.J. Vakharia, 2000. Scheduling a flowline manufacturing cell with sequence dependent family setup times. Eur. J. Operat. Res., 125: 324-339.

Shanker, K. and B.K. Modi, 1999. A branch and bound based heuristic for multi-product resource constrained scheduling problem in FMS environment. Eur. J. Operat. Res., 113: 80-90.