行政院國家科學委員會補助專題研究計畫成果報告

開發求解最小化最大流程時間於具等效平行機之流程式生產的啟發式排程方法

計畫類別：X 個別型計畫　　☐ 整合型計畫

計畫編號：NSC 96－2221－E－004－004－

執行期間：　2007 年 8 月 1 日　至　2008 年 7 月 31 日

計畫主持人：陳春龍

共同主持人：

計畫參與人員：　陳俊龍

成果報告類型(依經費核定清單規定繳交)：☐精簡報告　X 完整報告

本成果報告包括以下應繳交之附件：

☐赴國外出差或研習心得報告一份

☐赴大陸地區出差或研習心得報告一份

☐出席國際學術會議心得報告及發表之論文各一份

☐國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢

☐涉及專利或其他智慧財產權，☐一年☐二年後可公開查詢

執行單位：

中　華　民　國　2008　年　10　月 3 日

A Bottleneck-based Heuristic for Minimizing Makespan in a Flexible Flow Line with Unrelated Parallel Machines

Abstract

This study proposes the use of a bottleneck-based heuristic for a flexible flow line (BBFFL) to address a flexible flow line problem with a bottleneck stage, where unrelated parallel machines exist in all the stages, with the objective of minimizing the makespan. The essential idea of BBFFL is that scheduling jobs at the bottleneck stage may affect the performance of a heuristic for scheduling jobs in all the stages. Therefore, in BBFFL, a variant of Johnson's rule was used to develop a bottleneck-based initial sequence generator (BBISG). Then, a bottleneck-based multiple insertion procedure (BBMIP) was applied to the initial sequence to control the order by which jobs enter the bottleneck stage to be the same as that at the first stage. Five experimental factors were used to design 243 different production scenarios and ten test problems were randomly generated in each scenario. These test problems were used to compare the performance of BBFFL with several well-known heuristics. Computational results show that the BBFFL significantly outperforms all the well-known heuristics.

KEY WORDS: Bottleneck-based multiple insertion heuristic; Flexible flow line; Bottleneck, Unrelated parallel machines; Makespan

1. Introduction

This study considers the scheduling problems in a flexible flow line with a bottleneck stage and with unrelated parallel machines in all the stages. A flexible flow line (FFL) is also called a flexible flow shop (FFS), a hybrid flow shop (HFS), or a flow shop with multiple processors (FSMP). A typical FFL problem can be defined as follows: there are *N* jobs passing through a *J*-stage flow line with one or more parallel machines at each stage. There are unlimited buffers between two successive stages. The flow of jobs through the shop is unidirectional and moves from the first stage to the last stage in order. Flexible flow lines occur in many different manufacturing environments, including electronics manufacturing (Wittrock, 1988), the packaging industry (Adler et al., 1993), the pharmaceutical sector (Guinet and Solomon, 1996), glass container fabrication (Leon and Ramamoorthy, 1997), automobile assembly (Agnetis et al., 1997), printed circuit board (PCB) assembly (Jin et al., 2002; Sawik, 2002), PCB fabrication (Alisantoso et al., 2003; Lee et al., 2003; Choi et al., 2005), multilayer ceramic capacitor (MLCC) manufacturing (Yang et al., 2004), leadframe manufacturing (Lee et al., 2004), ceramic tile manufacturing (Ruiz and Maroto, 2006), and metal forming, plastic injection, weaving, and assemblies (Jenabi et al., 2007).

The bottleneck phenomena occur frequently in many manufacturing systems. Goldratt and Cox (1992) stated the idea that bottleneck resources govern overall system performance. Bottleneck management is a very important task on the shop floor and is extremely effective in production scheduling. Scheduling approaches for flow shop and job shop problems with bottleneck stages usually include three steps (Adler et al., 1993; Pinedo, 2002): (1) identify bottleneck stage, (2) schedule bottleneck stage, and (3) schedule non-bottleneck stages. The drum-buffer-rope (DBR) scheduling method proposed by Goldratt and Fox (1986), derived from the theory of constraint (TOC), is a renowned approach. This method concentrates on scheduling constraint resources (bottleneck resources) and deals with other resources (non-bottleneck resources) casually. However, Conway (1997) stated that it is often important to schedule subordinate resources carefully to ensure timely support of constraint resources. Therefore, several researchers considered full resource and presented bottleneck-based methods to solve FFL and flow shop problems. Chen and Lee (1998) suggested a bottleneck-based group scheduling procedure to solve flow line cell scheduling problems. The procedure was based on the bottleneck machine and attempted to fully utilize the bottleneck machine and minimize makespan. Lee et al. (2004) developed a bottleneck-based heuristic to solve a multi-stage hybrid flow shop problem with identical parallel machines at each stage and with minimum total tardiness as the objective. The heuristic first focuses on the bottleneck stage, constructs the schedule of the bottleneck stage, and then constructs schedules for other stages based on the schedule of the bottleneck stage. The heuristic uses the sum of processing times of a job at the upstream stages to be the arrival time of the job at the bottleneck stage. If the procedure results in an infeasible schedule, then the arrival times of the jobs at the bottleneck stages will be iteratively modified until a feasible schedule is obtained. They compared the performance of eight well-known dispatching rules with the bottleneck-based heuristic, and the computational results showed that the heuristic dominated all the dispatching rules. Rajendran and Alicke (2007) considered the problem of dispatching in flowshop with bottleneck machines; their objective was to develop dispatching rules to solve flowshop scheduling problems. They measured performance by considering three factors: the minimization of total flowtime, the minimization of the sum of earliness and tardiness, and the minimization of total tardiness, each considered separately.

Unrelated parallel machine problems considered in this paper refer to a job that may have different processing times at a stage in a flexible flow line. Unrelated parallel machine manufacturing systems are common in real-world factories. This may be because factories want to extend the capacity of their production by adding supplementary parallel machines at certain stages. Load imbalance is another factor leading to the supplementing of machines at certain stages. In addition, unrelated parallel machines may exist in stages due to the coexistence of new and old machines. Unrelated parallel machine scheduling can be found in real-world manufacturing environments, such as the drilling operations of PCB fabrication (Yu et al., 2002; Hsieh et al., 2003) and semiconductor wafer manufacturing (Kim et al., 2003). Since these manufacturing systems usually include a large number of stages, they can be classified as flexible flow lines with unrelated parallel machine problems. For example, Ruiz and Maroto (2006) presented a ceramic tile manufacturing system as a flexible flow line with unrelated parallel machine problems.

For the last few decades, many researchers have studied the FFL problem. However, most of them have dealt with multiple stages involving identical parallel machines. To the best of our knowledge, the FFL problem with unrelated parallel machines is rarely solved. The FFL problem is a NP-hard problem; it requires much computational time to find the optimal solution. A heuristic is an acceptable practice to find a good solution. Therefore, in this study, we develop a bottleneck-based heuristic to solve the FFL problem with a bottleneck stage and with unrelated parallel machines in all the stages. The objective of the candidate problem is to minimize makespan. The remainder of this study is organized as follows. In section 2 we provide a literature review. Section 3 presents the proposed heuristics. Section 4 describes and analyzes computational experiments. Finally, section 5 summarizes the major findings of this research and proposes some further research.

2. Literature review

Heuristics can be classified into three types: index-development, solution-construction, and solution-improvement (Framinan et al., 2004). However, some heuristics may consist of one or more of these types. Dispatching rule belongs to the type of index-development, and multiple-insertion heuristic is a solution-construction type. Meta-heuristic such as tabu search algorithm and simulated annealing algorithm can be regarded as a solution-improvement type. Obviously, solution-improvement type heuristics take the longest computation time to find a solution, but index-development type heuristics find a solution fast. A brief review of literature on the work in FFL problems using these three types of heuristics is presented below.

Over the past several decades, many heuristics have been developed for FFL problems with identical parallel machines. Dispatching rules are commonly used heuristics for FFL problems. Hunsucker and Shah (1994) evaluated six priority rules under congestion levels in a constrained flow shop with multiple processors environments for makespan, mean flowtime and maximum flowtime. The results of the simulation study indicated that the shortest processing time rule (SPT) yielded superior performance for makespan and mean flow time criteria. However, for the maximum flow time criterion, clear superiority of a particular dispatching procedure was not established. Kadipasaoglu et al. (1997) have also found the SPT rule to be the best rule on flow time and makespan. Brah and Wheeler (1998) investigated the effect on mean flow time and makespan of scheduling rules with dynamically established priorities in a FFL. Nine rules were tested and the SPT rule was consistently superior. Jayamohan and Rajendran (2000) studied many dispatching rules in flexible flow shops. They reported that PT +WINQ+AT (process time + work in next queue + arrival time) rule could offer good performance for the makespan.

In addition, many heuristics have been developed for pure flow shop problems. Palmer (1965), CDS (Campbell et al., 1970), Gupta (1971), DAN (Dannenbring, 1977), NEH (Nawaz et al., 1983), and CDSD (Park et al., 1984) are well-known ones. These heuristics have recently been applied to solve FFL problems. Santos et al. (1996) evaluated the performance of the heuristics of Palmer, CDS, Gupta, and DAN on FFL problems. Their results showed that the heuristics of CDS and DAN outperformed the other two heuristics. Brah and Loo (1999) investigated the performance of the heuristics of CDS, Gupta, DAN, NEH and CDSD. Their results showed that NEH dominated all other heuristics. Note that NEH is a solution-construction type heuristic.

Local searches or meta-heuristics such as tabu search, genetic algorithm, and simulated annealing algorithm are common approaches used to solve FFL problems. Chen et al. (1998) proposed a tabu search heuristic to solve flexible flow line scheduling problems with minimizing makespan. Wardono and Fathi (2004) considered FFL problems with limited buffer capacities between stages. Their primary objective was to find a schedule that would minimize makespan. They developed a tabu search algorithm to solve problems. Bertel and Billaut (2004) developed a genetic algorithm to solve a multi-processor flow shop problem involving recirculation and considered the objective of minimizing the total number of weighted tardy jobs. Sivrikaya Serifoglu and Ulusoy (2004) also proposed a genetic algorithm to solve FFL problems with minimizing makespan. Jin et al. (2006) proposed the simulated annealing algorithm to solve FFL problems with minimizing makespan.

As surveyed in Linn and Zhang (1999), most heuristics study identical parallel machines. To our best knowledge, only a few studies consider FFL problems with unrelated parallel machines. Adler et al. (1993) considered a very specific problem and proposed a bottleneck-based five-step method to solve the problem. Low (2005) developed a simulated annealing heuristic to solve flexible flow lines with unrelated parallel-machine problems in a flow shop. They considered the objective of minimizing the total flow time. Ruiz and Maroto (2006) proposed a genetic algorithm to solve a hybrid flow shop with sequence dependent setup times and with minimum makespan as the objective. Jenabi et al. (2007) considered the economic lot sizing and scheduling problem in FFL with unrelated parallel machines and considered the objective of minimizing the sum of setup and inventory holding costs per unit time without any stock-out. They proposed two hybrid meta-heuristics, a hybrid genetic algorithm, and a simulated annealing to solve the problems.

In summary, FFL problems with unrelated parallel machines are seldom solved and meta-heuristics are the most common approaches to solve the problems. Although meta-heuristics are usually effective and reasonably efficient, it may not be appropriate when efficiency is especially critical, such as a decision-making on job scheduling on the shop floor. Therefore, other types of heuristics such as index-development and solution-construction are needed to be developed to solve the problems considered in the study.

3. A bottleneck-based heuristic for the flexible flow line scheduling problem

A bottleneck-based heuristic is proposed to solve the candidate FFL problem. Because unrelated parallel machines are considered at each stage, three machine selection rules will be used to determine the schedule of the jobs at each stage in this study. Given a job at a stage, the first machine selection rule, EAAM, is to select the machine with the earliest available time among the available machines. The second selection rule, ECAM, is to select the machine with the earliest completion time when the job is assigned to the available machines. The third selection rule, ECALLM, is to select the machine with the earliest completion time when the job is assigned to all the machines at the stage, including available and unavailable machines. This rule may cause an idle period of the job. However, since unrelated parallel machines are considered at the stage, an unavailable but more efficient machine may produce an earlier completion time for the job. The three machine selection rules will be employed in all algorithms.

The notations, assumptions, and details of the proposed heuristics will be presented in the following subsections.

3.1 Notations

To describe the proposed method, we use the following notations:

i = job index, $i = 1,2,3,\ldots,n$

j = stage index, $j = 1,2,3,\ldots,J$

b = bottleneck stage index, $b \in [1,2,3,\ldots,J]$

s = machine index at stage j, $s = 1,2,3,\ldots,m_j$

m_j = number of unrelated parallel machines at stage j

$\overline{p}_{ij}$ = average processing time of job i at stage j

P_{ibs} = processing time of job *i* on machine *s* at bottleneck stage *b*

R_j= the workload of stage *j*

C_{iJ} = completion time of job *i* at last stage *J*

$fP_i^{\min}$ = total minimum processing time required for job *i* before the bottleneck stage *b*

$lP_i^{\min}$ = total minimum processing time required for job *i* after the bottleneck stage *b*

3.2 Assumptions

The FFL problem considered in this paper assumes that there are *J* stages and include a bottleneck stage *b*. There are m_j unrelated parallel machines in stage *j* and the number m_j may vary from stage to stage. There are *n* jobs to be processed and each job has the same routing and must visit all the stages consecutively. All jobs are available at time zero. The processing time of an operation of a job at a stage depends on the machine assigned at the stage, and it is known in advance. A machine can process only one job at a time, and jobs cannot be preempted. There are unlimited buffers between stages. There is no machine breakdown and no set-up time required before jobs are processed on any machine.

The workload is used as an indicator to identify the bottleneck stage. Since the machines are unrelated at each stage, the processing time of an operation at a stage is dependent upon the machine assigned to the operation. The workload of stage *j* is computed by the sum of the average processing times of all the operations processed at the stage divided by its number of machines, denoted as $R_j = (\sum_{i=1}^{n} \bar{p}_{ij})/m_j$. The stage with the largest $R_{j,}$ is defined as the bottleneck stage.

3.3 The Bottleneck-Based Heuristic for the Candidate FFL problem (BBFFL)

Garey et al. (1976) proved that minimizing makespan for the flow shop problem with three machines is NP-hard in the strong sense. Hoogeveen et al. (1996) proved that minimizing makespan for two-stage FFL problems with stage 1 having one machine and stage 2 having two machines or with stage 1 having two machines and stage 2 having one machine is NP-hard. Therefore, the candidate FFL problem considered is at least NP-hard.

A bottleneck-based heuristic (BBFFL) is proposed to solve the candidate FFL problem. The proposed heuristic belongs to the classification of solution construction. Its main idea is that scheduling the jobs at the bottleneck stage may affect the performance of a heuristic for scheduling the jobs in all the stages. Therefore, BBFFL generates schedules on the basis of the schedule produced at the bottleneck stage. This heuristic consists of three steps: (1) Identifying the bottleneck stage, (2) generating an initial sequence of the jobs by a bottleneck-based initial sequence generator (BBISG), and (3) applying a bottleneck-based multiple insertion procedure (BBMIP) to the initial sequence to generate the final schedule.

Applying Johnson's rule to solve the two-machine flow shop problem with makespan as the objective can obtain the optimal schedule. There have been many studies extending the idea of Johnson's rule to construct several variant heuristics (Gupta, 1997; Chen and Lee, 1998; Kurz and Askin, 2003). The bottleneck-based initial sequence generator (BBISG) is also a variant of Johnson's rule (SPT-LPT rule). The FFL can be divided into three subsystems: the upstream subsystem (the stages ahead of the bottleneck stage), the bottleneck subsystem (the bottleneck stage), and the downstream subsystem (the stages after the bottleneck stage). The BBISG heuristic generates job sequences mainly based on the total processing times in the upstream subsystem and the downstream subsystem of the jobs. Since unrelated parallel machines are considered in the stages, for each job, we choose the minimum processing time in each stage and sum up the minimum processing times in the stages (denoted as total minimum processing time) in the upstream subsystem and in the downstream subsystem respectively. The bottleneck-based initial sequence generator (BBISG) is described as follows:

Step 1. Set Ω to ϕ.

Step 2. Divide the system into upstream subsystem, bottleneck subsystem and downstream subsystem. Compute the total minimum processing times of the upstream subsystem ($fP_i^{\min}$) and the downstream subsystem ($lP_i^{\min}$) for each job.

Step 3. Assign jobs to set U if the jobs satisfy the following condition: $fP_i^{\min} \le lP_i^{\min}$; assign jobs to set L if the jobs satisfy the following condition: $fP_i^{\min} > lP_i^{\min}$.

Step 4. If $U = \phi$, go to Step 5. Select the job with the smallest value of $fP_i^{\min}$ for $i \in U$. If there is more than one job having the same smallest value of $fP_i^{\min}$, select the job with the maximum average processing time at the bottleneck stage ($\bar{p}_{ib}$). If the figures are, once again, the same, break the tie arbitrarily. Append the selected job to Ω and remove the job from set U; redo Step 4.

Step 5. If $L = \phi$, go to Step 6. Select the job with maximum value of $lP_i^{\min}$ for $i \in L$. If there is more than one job having the same maximum value of $lP_i^{\min}$, select the job with the maximum average processing time at the bottleneck stage ($\bar{p}_{ib}$). If the figures are, once again, the same, break the tie arbitrarily. Append the selected job to Ω and remove the job from set L; redo Step 5.

Step 6. Obtain an initial sequence of the jobs in Ω.

Step 7. Stop.

The BBMIP applies a multiple insertion procedure to the initial sequence generated by BBISG. Given a job sequence with n jobs, n-1 iterations will be performed in a general multiple insertion procedure. In each iteration, say iteration k, k partial sequences are generated by inserting the k-th job in the initial sequence into the positions before, between, and after every two consecutive jobs in the best partial sequence found in the k-1-th iteration, which includes the first k-1 job in the initial sequence. The partial sequence with the smallest makespan is the best partial sequence in the k-th iteration. Note that to calculate the makespan of a partial sequence, since the FFL problem is a flow shop problem, we can schedule the jobs in the partial sequence stage by stage. In the first stage, the jobs enter the stage according to the partial sequence, and a machine in the stage is chosen for a job based on a specified machine selection rule. The completion time of each job at the first stage becomes the arrival time of the job at the second stage. This determines the jobs' entering sequence at the second stage, which may not be the same as that at the first stage. The same machine selection rule is used to assign the machines in the second stage to the jobs. This procedure continues until the last stage and the completion of the last job at the stage determines the makespan of the partial sequence. In BBMIP, we modify the calculation of makespan for a partial sequence by adjusting the jobs' entering sequence at the bottleneck stage to be the same as that at the first stage. The procedure of BBMIP is presented as follows.

Step 1. Select the first job in the initial sequence generated by BBISG and let it be the current partial sequence.

Step 2. Select the next job in the initial sequence and insert the job into the positions before, between and after every two consecutive jobs of the current partial sequence.

Step 3. Calculate makespan for each partial sequence produced in Step 2 while adjusting jobs' entering sequence at the bottleneck stage to be the same as that at the first stage.

Step 4. Select the partial sequence with minimum makespan and let the partial sequence be the current partial sequence.

Step 5. If the current partial schedule includes all the *n* jobs, then stop; otherwise go to Step 2.

We use a simple example to illustrate the procedure of BBFFL. This example includes three jobs and three stages; each stage has two machines. The processing times of the jobs on each machine at each of the stages are shown in Figure 1. The workloads of the stages are R_1 = 66.25, R_2 = 135, R_3 = 72.5; Stage 2 is the bottleneck stage. ECALLM is the machine selection rule used in this example to select the machines at all the stages. Figure 2 displays the procedure to generate the initial sequence by using BBISG for the example, and Figure 3 displays the procedure to generate the best schedule by using BBMIP for the example. A four-field notation is used to express the schedule of a job at a stage in Figure 3. The first field describes the job number and the second field describes the selected machine for the job at the stage. The third and the fourth fields provide the start time and completion time of the job on the selected machine, respectively. Note that, in Figure 3, while calculating the makespan for all the partial sequences, the entering sequence at the bottleneck stage, Stage 2, is adjusted to be the same as that at Stage 1. However, as mentioned previously, jobs' entering sequence at a stage is dependent on the jobs' completion times at its previous stage, so jobs' entering sequence at Stage 2 may not be the same as that at Stage 1 if BBMIP is not applied. Figure 4 presents the two cases in Figure 3 whose jobs' entering sequence at Stage 2 is not the same as that at Stage 1 if BBMIP is not applied. For instance, in calculating the makespan for job sequence 3-1, the entering sequence of the two jobs at Stage 2 should be 1-3 if BBMIP is not applied because the completion time of job 1 at Stage 1 is 40, 1/M_{12}/0/40, and the completion time of job 3 at Stage 1 is 45, 3/M_{11}/0/45. With this entering sequence at Stage 2, if machine 1 is assigned to job 1, the completion time of job 1 is 130, and if machine 2 is assigned to job 1, the completion time of job 1 is 150. Therefore, according to ECALLM, machine 1 is selected for job 1 at Stage 2 and denoted as 1/M_{21}/40/130. The first case in Figure 4 presents the result for calculating the makespan for partial sequence 3-1 without the application of BBMIP; the makespan is 215, which is worse than that of the same sequence with the application of BBMIP. The second case in Figure 4 presents the result for calculating makespan for partial sequence 3-1-2 without the application of BBMIP; the makespan is 240, which is worse than that for the same sequence with the application of BBMIP.

4. Computational experiments

A series of computational experiments have been conducted to evaluate the performance of BBFFL under different production scenarios. Table 1 summarizes the experimental factors used to define the production scenarios: the number of jobs, the number of stages, the variation of job processing times, the position of the bottleneck stage in the flow line, and the workload difference between bottleneck and non-bottleneck stages. The number of jobs has three levels with values set at 30, 50, and 100 (low, medium, and high). The number of stages also has three levels with values set at 5, 10, and 20 (low, medium, and high), where the number of machines at each stage is generated from discrete uniform distribution in the range of [1, 10]. The processing times of an operation on different machines at stage j are generated from a discrete uniform distribution multiplied by the number of machines at stage j, m_j. The purpose of the term, m_j, is to balance the workload of the stages (Lee et al., 2004). Three ranges are considered for uniform distribution to define the variation of job processing times. They are the following ranges: [10, 50], [10, 100], and [10, 200] (low, medium, and high). The position of the bottleneck stage in the system has three levels: the first quarter, the second quarter, and the third quarter of the flow line. The exact position of the bottleneck stage is randomly selected from the first quarter, the second quarter, or the third quarter of the flow line. The workload difference between the bottleneck stage and the highest workload non-bottleneck stage is set at three ratios of 1.1, 1.5 and 2.0 (low, medium, and high). The workload of a specified bottleneck stage is created as follows: (1) with a given combination of the number of jobs and the number of stages, randomly generate the processing times of the operations of every job on every stage; (2) calculate workload R_j = $(\sum \overline{p}_{ij})/m_j$ for every stage $1 \leqq j \leqq J$, and choose the stage, say stage j', with the largest R value; (3) randomly select a bottleneck stage b, $b \neq j'$, from a predetermined quarter on the flow line; (4) with a specified workload difference (wd), modify the processing time of job i on machine m at bottleneck stage b, p_{ibs}, where new p_{ibs} = (old p_{ibs})×($R_{j'}$/old R_b)×(wd). This procedure will guarantee that the ((new R_b)/$R_{j'}$) equals the specified wd.

"Please Insert Table 1 about here"

With the five three-level factors considered in Table 1, there are a total of 243 production scenarios and ten test problems are generated for each scenario in the experiment. These problems are conducted to compare the performance of BBFFL and four well-known heuristics: CDS, DAN, NEH, and CDSD. The relative percentage deviation (RPD) from the best solution and the number of best solutions produced (NBS) are chosen to evaluate the performance of the heuristics. The RPD is defined as:

$$RPD = \frac{S_a - S_b}{S_b} \times 100.$$

S_a is the solution value obtained by method a, and S_b is respectively the best solution value among those obtained by the algorithms included in the comparison.

Table 2 presents the computational results of the heuristics and shows several noteworthy points. First, the machine selection rules significantly affect the performance of the well-known heuristics and BBFFL. For the well-known heuristics, when EAAM is used, the average RPDs are all around 79.34 to 91.17. They decrease to around 31.08 to 45.58 when ECAM is used and further decrease to around 3.28 to 10.78 when ECALLM is used. Also, when EAAM is used, the NBS are all 0, and they increase to several hundred when ECAM and ECALLM are used. As for BBFFL, when EAAM is used, the average RPD is around 70.16. It significantly decreases to about 22.07 when ECAM is used and further decreases to about 0.70 when ECALLM is used. The NBS shows the same tendency as that of the well-known heuristics. When EAAM is used, the NBS is 0. It increases to over 50 when ECAM is used and further increases to over 1,600 when ECALLM is used. These results demonstrate the strong effect of the machine selection rule, ECALLM, on the performance of all the heuristics. This may be due to the fact that in a stage with unrelated parallel machines, an unavailable but more efficient machine may produce earlier completion time for a job. Second, within each of the machine selection rules, BBFFL outperforms the well-known heuristics. When EAAM is used, the RPD produced by BBFFL is 70.16, and the RPD produced by the best well-known heuristic, CDS, is 79.34. When ECAM is used, the RPD produced by BBFFL is 22.07, and the RPD produced by the best well-known heuristic, NEH, is 31.08. When ECALLM is used, the RPD produced by BBFFL is 0.70, while the RPD produced by the best well-known heuristic, NEH, is 3.28. The NBS shows the same tendency; when ECALLM is used, the NBS produced by BBFFL is 1,639, and the NBS produced by the best well-known heuristic, NEH, is 470. Note that although BBFFL using ECALLM is not able to produce best solutions for all the test examples, it deviates from the best solutions, on average, by only 0.7%.

“Please Insert Table 2 about here”

We further applied the analysis of variance (ANOVA) and the least significance difference method (LSD) to test the RPD of the 2,430 test problems to confirm the previous conclusions. The four well-known heuristics and BBFFL are grouped into a factor and denoted as the algorithms. Table 3, the ANOVA table, shows that machine selection rules significantly affect the performance of the well-known heuristics and BBFFL, and Table 4, the result of the LSD test, shows that ECALLM significantly dominates ECAM and EAAM. Note that in Table 4, the machine selection rules are sequenced in descending order in terms of their average RPD, and the rules with the same letter represent that the performance of the rules are not significantly different. Then, within each of the machine selection rules, we applied the analysis of variance and the LSD test to test if BBFFL significantly outperforms the well-known heuristics. Since the test results for the three machine selection rules are about the same, we present only the results for ECALLM here. Table 5, the ANOVA table, shows that the performance of BBFFL and the four well-known heuristics is significantly different, and Table 6, the result of the LSD test, shows that BBFFL significantly dominates all the well-known heuristics.

“Please Insert Table 3 about here”
“Please Insert Table 4 about here”
“Please Insert Table 5 about here”
“Please Insert Table 6 about here”

According to the previous analyses, BBFFL working with ECALLM is the best heuristic for the candidate problem, and it is denoted as BBFFL/ECALLM. We further studied the performance of BBFFL/ECALLM under different production scenarios. We applied the LSD test to study the relative performance of BBFFL/ECALLM and the best well-know heuristic, NEH/ECALLM, at different levels of the experimental factors. Table 7 summarizes the average RPD produced by BBFFL/ECALLM and NEH/ECALLM at each level of all the factors, and Table 8 summarizes of the results of the LSD tests. The results show that BBFFL/ECALLM significantly dominates NEH/ECALLM at each level of all the factors. These results show the consistent performance of BBFFL/ECALLM at different production scenarios.

“Please Insert Table 7 about here”
“Please Insert Table 8 about here”

Finally, we discuss the efficiency of BBFFL/ECALLM. All the heuristics are coded using the C++ language, and all experiments are performed on a PC with Intel Xeon, 3.2 GHz CPU and 4 GB RAM. Table 9 displays the average computation time (in seconds) required for BBFFL and the four well-known heuristics to solve a problem. The results show that the average CPU time for BBFFL to solve a 100-job problem is around 6.88 seconds, which has no significant different from that of NEH. This concludes that BBFFL/ECALLM is efficient enough for real world problems.

“Please Insert Table 9 about here”

5. Conclusions

This research has concluded that machine selection rule is a key factor affecting the performance of the heuristics for the FFL problem with unrelated parallel machines, and the machine selection rule, ECALLM, significantly dominates the other two machine selection rules, EAAM and ECAM. Also, the proposed heuristic, BBFFL significantly outperforms the four well-known heuristics, CDS, CDSD, DAN and NEH. The heuristic, BBFFL/ECALLM, produced best solutions for most of the test problems (1639 out of 2430), and it deviates from the best solutions, on average, by only 0.7%. Furthermore, the experimental design has shown that the performance of BBFFL is fairly stable; it consistently and significantly dominates all the well-known heuristics in different levels of all the experimental factors.

As mentioned in the literature review, there were only a few research papers published on the FFL problem with unrelated parallel machines. These papers dealt with FFL problems considering different characteristics such as sequence dependent setup times and solved the problems using local search methods such as simulated annealing and genetic algorithms. Although local search methods were usually effective for the problems, they were computationally burdensome for large size problems. Therefore, the proposed bottleneck-based heuristic, BBFFL/ECALLM, can be further applied to FFL problems considering some characteristics such as sequence dependent setup operations and reentrant processing; it can also be applied to solve other scheduling problems such as job shop problems with bottleneck stages.

REFERENCES

Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, J. C. and Wu, T. P., 1993. BPSS: A scheduling support system for the packaging industry. Operations Research 41, 641–648.

Agnetis, A., Pacifici, A., Rossi, F., Lucertini, M., Nicoletti, S., Nicolo, F., Oriolo, G., Pacciarelli, D. and Pesaro, E., 1997. Scheduling of flexible flow shop in an automobile assembly plant. European Journal of Operational Research 97, 348–362.

Alisantoso, D., Khoo, L. P. and Jiang, P. Y., 2003. An immune algorithm approach to the scheduling of a flexible PCB flow shop. International Journal of Advanced Manufacturing Technology 22, 819–827.

Bertel, S. and Billaut, J. C., 2004. A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. European Journal of Operational Research 159, 651–662.

Brah, S. A. and Loo, L. L., 1999. Heuristics for scheduling in a flow shop with multiple processors. European Journal of Operational Research 113, 113–122.

Brah, S. A. and Wheeler, G. E., 1998. Comparison of scheduling rules in a flow shop with multiple processors: A simulation, Simulation 71, 302–311.

Campbell, H. G., Dudek, R. A. and Smith, M. L., 1970. A heuristic algorithm for the n-job, m-machine sequencing problem. Management Science 16, 630–637.

Chen, C. L., Usher, J. M. and Palanimuthu, N., 1998. A tabu search based heuristic for a flexible flow line with minimum flow time criterion. International Journal of Industrial Engineering 5, 157–168.

Chen, Y. C. and Lee, C. E., 1998. Bottleneck-based group scheduling in a flow line cell. International Journal of Industrial Engineering-Applications and Practice 5, 288–300.

Choi, S. W., Kim, Y. D. and Lee, G. C., 2005. Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. International Journal of Production Research 43, 2149–2167.

Conway, R., 1997. Comments on an exposition of multiple constraint scheduling. Production and Operations Management 6, 23–24.

Dannenbring, D. G., 1977. An evaluation of flow shop sequencing heuristics. Management Science 23, 1174–1182.

Garey, M. R., Johnson, D. S. and Sahni, S., 1976. Fowshop and jobshop schedules: complexity and approximation. Mathematics of Operations Research 1, 117–127.

Goldratt, E. and Fox, R., 1986. The Race. (North River Press: New York).

Goldratt, E. and Cox, J., 1992. The Goal: A Process of Ongoing Improvement. (North River Press: New York).

Guinet, A. G. P. and Solomon, M., 1996. Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. International Journal of Production Research 34, 1643–1654.

Gupta, J. N. D., 1971. A functional heuristic algorithm for the flow-shop scheduling problem. Operations Research Quarterly 22, 39–47.

Gupta, J. N. D., 1997. A flowshop scheduling problem with two operations per job. International Journal of Production Research 35, 2309–2325.

Hoogeveen, J. A., Lenstra, J. K. and Veltman, B., 1996. Preemption scheduling in a two-stage multiprocessor flowshop is NPhard. European Journal of Operational Research 89, 172–175.

Hsieh, J. C., Chang, P. C. and Hsu, L. C., 2003. Scheduling of drilling operations in printed circuit board factory. Computers and Industrial Engineering 44, 461–473.

Hunsucker, J. L. and Shah, J. R., 1994. Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. European Journal of Operational Research 72, 102–114.

Jayamohan, M. S. and Rajendran, C., 2000. A comparative analysis of two different approaches to scheduling in flexibile flow shops. Production Planning and Control 11, 572–580.

Jenabi, M., Fatemi Ghomi, S. M. T., Torabi, S. A. and Karimi, B., 2007. Two hybrid meta-heuristics for the finite horizon ELSP in flexible flow lines with unrelated parallel machines. Applied Mathematics and Computation 186, 230–245.

Jin, Z. H., Ohno, K., Ito, T. and Elmaghraby, S. E., 2002. Scheduling hybrid flowshops in printed circuit board assembly lines. Production and Operations Management 11, 216–230.

Jin, Z. H., Yang, Z. and Ito, T., 2006. Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. International Journal of Production Economics 100, 322–334.

Johnson, S. M., 1954. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–67.

Kadipasaoglu, S. N., Xiang, W. and Khumawala, B. M., 1997. A comparison of sequencing rules in static and dynamic hybrid flow systems. International Journal of Production Research 35, 1359–1384.

Kim, D. W., Na, D. G. and Chen, F. F., 2003. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. Robotics and Computer Integrated Manufacturing 19, 173–181.

Kurz, M. E. and Askin. R. G., 2003. Comparing scheduling rules for flexible flow lines. International Journal of Production Economics 85, 371–388.

Lee, G. C., Kim, Y. D., Kim, J. G. and Choi, S. H., 2003. A dispatching rule-based approach to production scheduling in a printed circuit board manufacturing system. Journal of the Operational Research Society 54, 1038–1049.

Lee, G. C., Kim, Y. D. and Choi, S. W., 2004. Bottleneck-focused scheduling for a hybrid flowshop. International Journal of Production Research 42, 165–181.

Leon, V.J. and Ramamoorthy, B., 1997. An adaptable problemspace-based search method for flexible flow line scheduling. IIE Transactions 29, 115–125.

Linn, R. and Zhang, W., 1999. Hybrid flow shop scheduling: A survey. Computer & Industrial Engineering 37, 57–61.

Low, C. Y., 2005. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Computers and Operations Research 32, 2013–2025.

Nawaz, M., Enscore, E. E. and Ham, I., 1983. A heuristic algorithm for the m-Machine, n-Job flow-shop sequencing problem. OMEGA 11, 91–95.

Palmer, D. S., 1965. Sequencing jobs through a multi-stage process in the minimum total time-A quick method of obtaining a near optimum. Operations Research Quarterly 16, 101–107.

Park, Y. B., Pegden, C. D. and Enscore, E. E., 1984. A survey and evaluation of static flow shop scheduling heuristics. International Journal of Production Research 22, 127–141.

Pinedo, M., 2002. Scheduling: Theory, Algorithms, and Systems. (Prentice-Hall, Upper

Saddle River, New Jersey).

Rajendran, C. and Alicke, K., 2007. Dispatching in flowshops with bottleneck machines. Computers and Industrial Engineering 52, 89–106.

Ruiz, R. and Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169, 781–800.

Santos, D. L., Hunsucker, J. L. and Deal, D.E., 1996. An evaluation of sequencing heuristics in flow shops with multiple processors. Computers and Industrial Engineering 30, 681–692.

Sawik, T., 2002. An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers. Mathematical and Computer Modeling 36, 461–471.

Sivrikaya Serifoglu, F. and Ulusoy, G., 2004. Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach. Journal of the Operational Research Society 55, 504–512.

Wardono, B. and Fathi, Y., 2004. A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities. European Journal of Operational Research 155, 380–401.

Wittrock, R.J., 1988. An adaptable scheduling algorithm for flexible flow lines. Operations Research 36, 445–453.

Yang, T., Kuo, Y. and Chang, I., 2004. Tabu-search simulation optimization approach for flow-shop scheduling with multiple processors – a case study. International Journal of Production Research 42, 4015–4030.

Yu, L., Shih, H. M., Pfund, M., Carlyle, W.M. and Fowler, J.W., 2002. Scheduling of unrelated parallel machines: an application to PWB manufacturing. IIE Transactions 34, 921–931.

Table 1. Experimental design for generating test problems

Experimental Factors	Levels	
Number of jobs	30	L
	50	M
	100	H
Number of stages	5	L
	10	M
	20	H
Variation of processing times	U[10, 50]	L
	U[10, 100]	M
	U[10, 200]	H
Position of bottleneck stage	The first quarter of the flow line	L
	The second quarter of the flow line	M
	The third quarter of the flow line	H
Workload difference	1.1	L
	1.5	M
	2.0	H

Table 2. Performance comparisons of the proposed heuristics and the dispatching rules (in terms of average RPD and number of best solutions (NBS))

Algorithms	Average RPD			NBS		
	Machine selection rules			Machine selection rules		
	EAAM	ECAM	ECALLM	EAAM	ECAM	ECALLM
CDS	79.3433	36.6330	5.2855	0	0	173
DAN	91.1659	45.5789	10.7767	0	0	9
CDSD	79.5552	36.6229	5.2743	0	0	173
NEH	79.8047	31.0792	3.2767	0	7	470
BBFFL	70.1643	22.0729	**0.6988**	0	54	**1639**

Table 3. Analysis of Variance to test the significance of the machine selection rules

Source of variation	Degree of freedom	Sum of squares	Mean squares	F^{**}
Algorithms	4	1236277.50	309069.37	495.51**
Machine selection rules	2	34657428.41	17328714.20	27781.95**
Error	36443	22730955.43	623.74	
Total	36449	58624661.33		

**Difference in the effects at significance level 0.01.

Table 4. Results of LSD Test for the machine selection rules

Machine selection rules	RPD	Results* (groups)
ECALLM	5.0624	A
ECAM	34.3974	B
EAAM	80.0067	C

*Statistically significant difference (at significance level of 0.01) between machine selection rules with different alphabet letters.

Table 5. Analysis of Variance to test the significance of the algorithms when ECALLM is used

Source of variation	Degree of freedom	Sum of squared error	Mean squared error	F^{**}
Algorithms	4	133595.82	33398.96	1571.89**
Error	12145	258052.00	21.25	
Total	12149	391647.82		

**Difference in the effects at significance level 0.01.

Table 6. Results of LSD Test for the algorithms when ECALLM is used

Algorithms	RPD	Results* (groups)
BBFFL	**0.6988**	**A**
NEH	3.2767	B
CDSD	5.2743	C
CDS	5.2855	C
DAN	10.7767	D

*Statistically significant difference (at significance level of 0.01) between machine selection rules with different alphabet letters.

Table 7. Average RPD produced by BBFFL/ECALLM and NEH/ECALLM at each level of the experimental factors

Algorithms	Number of Jobs			Number of Stages			Position of the bottleneck stage			Workload differences			Range of processing times		
	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H
BBFFL/ECALLM	1.1226	0.604	0.37	0.6472	0.6956	0.7537	0.6525	0.6727	0.7713	0.7013	0.6836	0.7116	0.5055	0.6489	0.9421
NEH/ECALLM	3.0135	3.333	3.4836	3.2236	3.2624	3.3441	2.9636	3.3524	3.514	3.0501	3.3054	3.4747	3.3838	3.3199	3.1264

Table 8. Results of LSD Test for BBFFL/ECALLM and NEH/ECALLM at each level of the experimental factors

Algorithms	Number of Jobs			Number of Stages			Position of the bottleneck stage			Workload differences			Range of processing times		
	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H
BBFFL/ECALLM	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
NEH/ECALLM	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B

Table 9. Average computation time required for the proposed heuristic and the dispatching rules

Heuristics	CPU times (s)						
	Number of jobs			Number of stages			Overall
	Low	Medium	High	Low	Medium	High	
CDS	0.0071	0.0140	0.0421	0.0035	0.0124	0.0473	0.0211
DAN	0.0008	0.0012	0.0030	0.0009	0.0014	0.0027	0.0017
CDSD	0.0073	0.0141	0.0423	0.0037	0.0126	0.0473	0.0212
NEH	0.1244	0.6372	7.0868	1.1457	2.2474	4.4553	2.6161
BBFFL	0.1238	0.6292	6.8824	1.0566	2.1725	4.4063	2.5451