A Heuristic to Minimize Makespan in Proportional Parallel Flow Shops
Ameer Al-Salem
Pages 98-107

A Heuristic to Minimize Makespan in Proportional Parallel Flow Shops

Ameer Al-Salem
Mechanical Engineering Department, University of Qatar, Doha, P.O. Box 2713

Abstract: *This paper addresses the problem of minimizing the makespan on a two-stage parallel flow shops with proportional processing times. The problem of scheduling n independent jobs on m proportional parallel flow shops to minimize makespan is known to be NP-hard and hence too difficult to solve in polynomial time. A multi-phase heuristic algorithm to minimize the makespan on two proportional parallel flow shops is proposed. A simulation study is conducted to examine the effectiveness of the proposed heuristic algorithm for small and large problems. The results and analysis of quite extensive computational experiments for small size problems show that the proposed algorithm can yield solutions within a few percent of optimal solutions. The computational results for large size problems show that the performance of the proposed algorithm is relatively more effective than the existing SKV algorithm.*

Keywords: *Scheduling, Sequencing, Heuristics, Parallel Flow Shops.*

Received*: January 25, 2004* | **Revised***: February 15, 2005* | **Accepted***: March 11, 2005*

1. Introduction

Scheduling is one of the most important decisions in production control systems. In recent years extensive research work has been undertaken in this area, especially in developing various optimization and approximation algorithms. Techniques used for scheduling include linear and integer programming, branch and bound, tabu search, genetic algorithms, etc. The problem of scheduling jobs on parallel machines has received a good deal of attention in the research literature. However, the problem of scheduling jobs on parallel processing systems where the systems are flow shops with two or more machines has received less attention. In this paper, we consider a two-stage parallel flow shops scheduling problem having the following characteristics:

1. All jobs are available for machine processing simultaneously at time zero.
2. Two parallel flow shops are available, each of which consists of two stages.
3. Two proportional machines are available at each stage.
4. Each job has to be processed in exactly one machine in each stage starting with stage 1 and ending with stage 2.
5. Switching jobs between flow shops is not allowed.
6. The jobs can be processed in either the slower flow shop (flow shop 1) or in the faster flow shop (flow shop 2).
7. The processing time in flow shop 1 is a multiple of the processing time in flow shop 2.

Problems such as the one described above can be formulated as a two-stage parallel flow shops with proportional processing times. Figure 1 illustrates a schematic diagram of this scheduling situation with two stages.

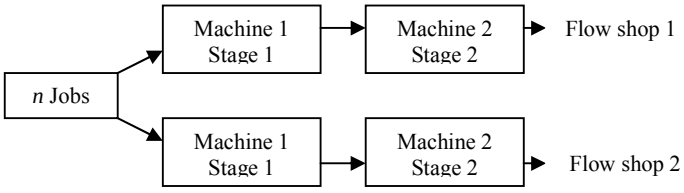

Figure 1. Two-stage parallel flow shops with proportional processing times

At each stage *j*, for a job, there are preferred machines and non-preferred machines. A preferred machine processes a job faster than a non-preferred machine. The objective is to minimize the makespan. An industrial situation wherein new technology/ equipment operates in parallel with older technology/equipment fit the scheduling situation described above.

This paper proposes a new polynomial-time heuristic to solve the above NP-hard scheduling problem. The primary objective is to allocate and

sequence the jobs on flow shops 1 and 2 so that the makespan for completing all the jobs is minimized. Minimizing makespan aims at achieving high utilization of equipment and resources by getting all jobs out quickly. This is an important scheduling criterion especially for automated systems, because of the high investment cost.

Most flow shop problems have been shown to be NP-hard problems. Thus, efficient algorithms for obtaining an optimal solution only exist for very special cases of simplified flow shops (e.g., Johnson's algorithm for the two-stage flow shop problem, [17]).

Single-stage scheduling problems with identical machines have been studied intensively in the literature. Garey and Johnson [10] showed that minimizing the makespan on *m* identical machines is NP-hard and hence the existence of a polynomial time algorithm is highly unlikely. Consequently, for single-stage scheduling problems of this type, most researchers have studied heuristic methods that provide an approximate solution. An elementary result for this class of scheduling problems has been introduced by McNaughton [20]. He introduced an algorithm called the wrap-around algorithm to minimize makespan on *m* identical parallel machines. For more approximation algorithms for single-stage scheduling problems with identical machines see Rothkopf [24], Elmaghraby and Park [9], Barnes and Berennan [4], and Dogramaci and Surkis [8].

For single-stage problems with proportional machines, Vel and Shijie [28] proposed a non-polynomial algorithm that used the bin-packing techniques to minimize the makespan on a set of proportional machines. This class of scheduling problems has been studied extensively in Gonzalez and Sahni [12], Gonzalez, Ibarra, and Sahni [11], and Liu and Liu [18].

For single-stage problems with unrelated machines, Horowitz and Sahni [15] developed an exact and approximation algorithm that is similar to Sahni's [25] for identical machines. The Earliest Completion Time heuristic (ECT) was presented by Ibarra and Kim [16] who also developed another four heuristics that are based on ECT. The Earliest Completion Time heuristic is similar to the Longest Processing Time first (LPT) rule to minimize the makespan on identical parallel machines. The LPT rule orders the jobs by decreasing order of processing time and assigns them to machine that results in lowest partial makespan. The ECT heuristic, however, assigns at $t = 0$ the m largest average processing time (over the machine) jobs to m machines. After that, whenever a machine is free, the unscheduled largest average processing time job is put on that machine. This heuristic tries to place the shortest average processing time jobs toward the end of the schedule where they can be used for balancing the loads.

For more approximation algorithms for single-stage problems with unrelated machines see Davis and Jaffe [6], De and Morton [7], Potts [22], Hariri and Potts [14], Van de Velde [27], and Martello, Soumis and Toth [19]. This class of unrelated parallel machine scheduling problems is the simplest relaxation of the problem discussed in this paper.

For the two-stage case where there are two machines at the first stage and two machines at the second stage, the heuristic algorithms developed by Sundaraghavan, Kunnathur, and Viswanathan [26] and Al-Kuwari, Al-Zara, Ashknani, and Salem [1] may be used to find near approximate solutions.

For the two-stage case where there are two machines at the first stage and a single machine at the second stage, Gupta [13] developed an approximation algorithm to find near optimal solutions while Arthanary and Ramaswamy [2] developed a branch and bound algorithm to solve the problem optimally.

For the two-stage case where there is a single machine at the first stage and two machines at the second stage, Rao [23] developed a heuristic algorithm to find good approximate solutions. For the two-stage case where there is a single machine at the first stage and multiple machines at the second stage, Narasimhan and Panwalkar [21] developed an approximation heuristic algorithm. They assumed that a job is processed by exactly one machine in the first stage, but by all machines in the second stage and a job could start on any machine in the second stage, as soon as it has been finished in the first stage. Past work on single-stage and two-stage scheduling problems is summarized in Table 1.

The roots of the present paper lie in six main sections. We start in Section 2 by introducing the notation sued in the paper. In Section 3 we introduce the scheduling problem and its formulation as a mathematical program. In Section 4 we include a brief discussion of existing scheduling approaches that bear on the problem of minimizing makespan on proportional parallel flow shops. In Section 5 we include a description of the new multi-phase heuristic algorithm to find near optimal solutions to the formulated problem. Section 6 illustrates the computational studies to evaluate the performance of the new heuristic algorithm. Finally, the paper is concluded in Section 7.

Table 1. Brief summary of past work on single-stage and two-stage scheduling problems

Authors (year)	Problem Type	Solution Methodology
Garey and Johnson [10]	Single-stage with identical machines	Complexity analysis
McNaughton [20]	Single-stage with identical machines	Wrap-around algorithm
Rothkopf [24]	Single-stage with identical machines	Approximation algorithm
Elmaghraby and Park [9]	Single-stage with identical machines	Approximation algorithm
Barnes and Berennan [4]	Single-stage with identical machines	Approximation algorithm
Dogramaci and Surkis	Single-stage with	Approximation algorithm

[8]	identical machines	
Vel and Shijie [28]	Single-stage with uniform machines	Bin-packing technique
Gonzalez and Sahni [12]	Single-stage with uniform machines	Approximation algorithm
Gonzalez, Ibarra, and Sahni [11]	Single-stage with uniform machines	Bounds for LPT
Horowitz and Sahni [15]	Single-stage with uniform machines	Exact and approximation algorithms
Liu and Liu [18]	Single-stage with uniform machines	Bounds on scheduling algorithms
Horowitz and Sahni [15]	Single-stage with unrelated machines	Exact and approximation algorithms
Ibarra and Kim [16]	Single-stage with unrelated machines	ECT algorithm
Davis and Jaffe [6]	Single-stage with unrelated machines	Approximation algorithm
De and Morton [7]	Single-stage with unrelated machines	Approximation algorithm
Potts [22]	Single-stage with unrelated machines	Approximation algorithm
Hariri and Potts [14]	Single-stage with unrelated machines	Approximation algorithm
Martello, Soumis and Toth [19]	Single-stage with unrelated machines	Exact and approximation algorithms
Sundaraghavan, Kunnathur, and Viswanathan [26]	Two-stage with two machines at each stage	Approximation algorithms
Al-Kuwari, Al-Zara, Ashknani, and Salem [1]	Two-stage with two machines at each stage	Approximation algorithm
Gupta [13]	Two-stage with two machines at the first stage and a single machine at the second stage	Approximation algorithm
Arthanary and Ramaswamy [2]	Two-stage with two machines at the first stage and a single machine at the second stage	Branch and bound exact algorithm
Rao [23]	Two-stage with a single machine at the first stage and two machines at the second stage	Approximation algorithm
Narasimhan and Panwalkar [21]	Two-stage with one machine at the first stage and multi machines at the second stage	Approximation algorithm

2. Notation

The following notation is used throughout the paper with additional notation introduced when necessary:

n number of jobs;

N the set of jobs. $N = \{1,\ldots,n\}$;

F_1 the set of jobs processed in flow shop 1;

F_2 the set of jobs processed in flow shop 2;

P_{i1} the processing time of job i on machine 1 in flow shop 1;

P_{i2} the processing time of job i on machine 2 in flow shop 1;

T_i the sum of the processing times of job i on machines 1 and 2 in flow shop 1. $T_i = P_{i1} + P_{i2}$;

α processing time multiplier for job processing times on flow shop 2 relative to flow shop 1;

C_1 the partial makespan to date on flow shop 1;

C_2 the partial makespan to date on flow shop 2;

C_{max} the maximum partial makespan and equal to $\max(C_1, C_2)$;

I_j idle time in machine 2 of flow shop 1 preceding job j;

K_j idle time in machine 2 of flow shop 2 preceding job j;

X_i a binary variable equal to 1 if job i is assigned to flow shop 1 and 0 otherwise.

3. The Scheduling Model

This paper considers the scheduling problem under the following conditions. A set of n jobs is processed on two proportional parallel flow shops, where the processing times P_{i1} and P_{i2} are given. The processing time multiplier α for job processing times on flow shop 2 relative to flow shop 1 is prespecified. All jobs are available to be scheduled at the beginning of the planning period.

The makespan minimization on a two-machine flow shop can be formulated as a linear program [3]. Makespan minimization on a proportional flow shop problem can also be formulated as a zero-one mixed integer programming (MIP) by extending that formulation as follows [26]:

$$\min C_{\max} \qquad (a)$$

subject to :

$$C_{\max} \geq \sum_{j=1}^{n} I_j + \sum_{j=1}^{n} X_j P_{j2} \qquad (1.a)$$

$$C_{\max} \geq \sum_{j=1}^{n} K_j + \sum_{j=1}^{n} (1 - X_j)\alpha P_{j2} \qquad (2.a)$$

$$I_j \geq \sum_{i=1}^{j} X_i P_{i1} - \sum_{i=1}^{j-1} X_i P_{i2} - \sum_{i=1}^{j-1} I_i \ \text{for } j \in N \qquad (3.a)$$

$$K_j \geq \sum_{i=1}^{j} (1 - X_i)\alpha P_{i1} - \sum_{i=1}^{j-1} (1 - X_i)\alpha P_{i2} - \sum_{i=1}^{j-1} K_i \ \text{for } j \in N \qquad (4.a)$$

$$I_j \geq 0, \ j \in N \qquad (5.a)$$

$$K_j \geq 0, j \in N \qquad (6.a)$$

$$X_j = 1 \text{or } 0, j \in N \qquad (7.a)$$

The objective of problem (a) is to minimize the makespan. Constraints (1.a) and (2.a) relate to the computation of the completion time of last job processed in flow shops 1 and 2. Constraints (3.a) and (4.a) refer to the calculation of idle times in machine 2 preceding job j in flow shops 1 and 2. The objective guarantees that the overall makespan will be minimized. It is worth noting that a job j that is

assigned to flow shop 1 (i.e., X_j =1) is automatically not assigned to flow shop 2 (i.e., $(1-X_j)=0$). Hence, in this formulation a job will always be assigned to either flow shop 1 or flow shop 2 and the number of jobs assigned to flow shops 1 and 2 taken together will always be equalled to the sum of *n* jobs.

The MIP can be used to optimally solve the problem addressed in this paper. However, when the number of machines and/or the number of jobs becomes large, which is the case in many real world problems, the MIP becomes too large to be solved in a reasonable amount of time. In this paper, a multi-phase algorithm that uses a constructive heuristic to assign jobs to flow shops and an improvement heuristic to improve the solution is developed to obtain near optimal solution values to problem (a).

4. Review of Sundararaghavan-Kunnathur-Viswanthan Heuristic

Before trying to develop a new and better heuristic algorithm that minimizes the makespan on proportional parallel flowshops, the previous ones should be analyzed in order to locate and avoid the problems inherent in them. In this context, we consider the algorithm developed by Sundararaghavan, Kunnathur, and Viswanathan (SKV) [26] because it is the most recent and most efficient of the algorithms available. Sundararaghavan, Kunnathur, and Viswanathan developed two approximation algorithms to minimize the makespan on proportional parallel flow shops. To measure the effectiveness of the two algorithms, they compared the algorithms' solution values with optimal solutions for problems of small size (n = 15). For problems of large size (n = 30 and n = 50), they compared the algorithms' solution values with lower bounds on the optimal objective function value. Their computational results showed that both algorithms could yield good solutions with one algorithm, the SKV algorithm, outperformed the other one in most cases.

The SKV algorithm starts by allocating the job with the largest combined processing time (the processing time on the first stage and the processing time on the second stage) to the fast flow shop. It then continues to assign jobs one at a time in a decreasing order of combined processing times to either flow shop in a myopic makespan-reducing manner until all jobs are assigned. The S-K-V algorithm proceeds as follows:

Phase 1 (Ordering the jobs)

1. Let $C_1 = C_2 = C_{max} = 0$
2. For i = 1 to n, Calculate: T_i ($T_i = P_{i1} + P_{i2}$)
3. Order the jobs by decreasing values of T_i

Phase 2 (Assigning jobs to flow shops using Johnson's algorithm)

4. For i = 1 to n in the ordered set in step 3, Do:
   - Using Johnson's rule sequencing, assign job i to F_1 if $C_1 < C_2$ otherwise assign job i to F_2. Update $C_{max} = \max(C_1, C_2)$.

End.

The Johnson algorithm used in step 4 always generates an optimal schedule for the two-machine flow shop problem (Daniel, Robert, and Bulfin [5]). The Johnson algorithm proceeds as follows:

0. Let $N = \{1,\ldots,n\}$, $K = 1$, $L = n$, and position $(i) = 0$ for $i = 1,\ldots,n$.
1. While $N \neq \varnothing$, Do:
   (a) Let $P_{i^* j^*} = \min\{\min_{i=1,n} P_{i1}, \min_{i=1,n} P_{i2}\}$. If $j^* = 1$, go to (b). If $j^* = 2$, go to (c)
   (b) Schedule job i^* in the earliest available position (K) of the sequence. Set position $(K) = i^*$, $K = K + 1$ and $N = N - \{i^*\}$. Go to 1.
   (c) Schedule job i^* in the latest available position (L) of the sequence. Set position $(L) = i^*$, $L = L - 1$ and $N = N - \{i^*\}$. Go to 1.

The above algorithm determines the optimal sequence that minimizes the makespan on a single flow shop with two machines. The sequence of the jobs is determined by (a) and (b) where each job is allocated to a position i ($i = 1,\ldots,n$) in the sequence.

The proposed heuristic algorithm, in the following section, uses different way of allocating n jobs to flow shops to improve the assignment obtained in Phase 2 of the SKV heuristic algorithm. Specifically, the proposed algorithms use a constructive heuristic to assign jobs to flow shops followed by an improvement heuristic to further improve the solution obtained by the constructive heuristic.

5. Proposed Heuristic Algorithm

Minimizing the makespan on two-stage parallel flow shops with proportional processing times can be characterized as a multi-level optimization problem. At the first level, jobs are assigned to flow shops and at the second level, the solution is modified by balancing the load between the flow shops.

According to De and Morton [7], there are three principal objectives that need to be achieved in order to solve the makespan problem on unrelated parallel machines:

1. To make the load on different machines as equal as possible.

2. To avoid "lumpiness" in the load near the end, i.e., to avoid late processing of an excessively long job.
3. To process each job on the machine on which it has a "comparative advantage."

In general, it is not possible to simultaneously satisfy all three objectives mentioned above. The best that can be done is to look for a good compromise among them. The new algorithm is built on this idea. While the concentration on Objective 2 is always maintained, at the initial stage the algorithm mainly emphasizes 3; but as the flow shops get "close" to capacity the emphasis is shifted towards 1. Since "close" is not well defined, several values for closeness are tried and the best result is selected.

A new polynomial-time heuristic, called the partitioning flow shop heuristic (PFH), is developed to allocate and sequence the jobs on the flow shops so that the makespan for completing all the jobs is minimized. The proposed PFH algorithm incorporate the concept used by De and Morton [7] to minimize the makespan on unequal parallel machines by making an initial assignment of selected jobs and then assigning the remaining pending jobs. Specifically, the new PFH algorithm uses a constructive heuristic to assign jobs to flow shops followed by an improvement heuristic to further improve the solution obtained by the constructive heuristic. Here, a "solution" is a satisfactory feasible assigned sequence of jobs to flow shops. A basic layout of the proposed partitioning flow shop heuristic is shown in Figure 2.

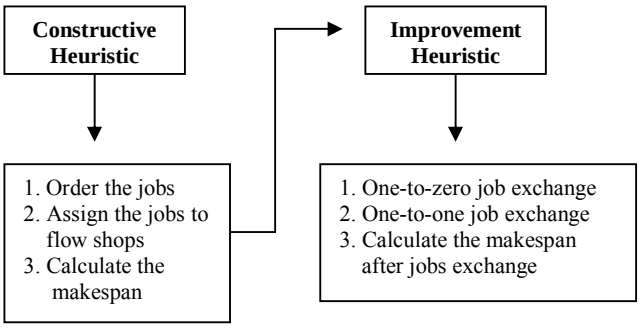

Figure 2. Basic layout of the partitioning flow shop heuristic (PFH).

5.1 Phase 1—Constructive Heuristic

The constructive assignment heuristic proceeds in two stages. The first stage starts by ordering the jobs in a decreasing order of combined processing times (the processing time on the first stage and the processing time on the second stage). The rationale behind ordering the jobs with the largest combined processing time first is to avoid lumpiness (Objective 2) caused by longer jobs near the end. Following this order, the jobs are assigned one at a time to either flow shop if the partial makespan does not exceed the value of βC_{SKV}, where C_{SKV} is the makespan of the SKV algorithm and β is a number between 0 and 1. If the partial makespan exceeds βC_{SKV}, i.e., a flow shop is regarded to be loaded close to capacity; the job is considered to be pending and is assigned to a flow shop in stage 2.

In stage 2, pending jobs are assigned one at a time in decreasing order of combined processing times to either flow shop in myopic makespan-reducing manner until all jobs are assigned. The reason for assigning the remaining jobs (pending jobs) in decreasing order of their combined processing times is again to avoid lumpiness. This approach follows proven concepts of scheduling the most restrictive jobs first (e.g., LPT).

Let PJ represent the set of jobs that have been considered for assignment and not yet assigned (pending) and N_{PJ} represent the number of jobs belong to PJ. The phase 1-constructive heuristic proceeds as follows:

0. Repeat Steps 1 through 5 for different values of β and select the best solution.

Stage 1 (Initial job assignment to flow shops)

1. Let PJ = $\varnothing$; $N_{PJ} = 0$; $C_1 = C_2 = C_{max} = 0$
2. For $i = 1$ to n, Calculate: T_i
3. Order the jobs by decreasing values of T_i
4. For $i = 1$ to n in the ordered set in step 3, Do:
   (a) Using Johnson's rule, assign job i to F_1 if $C_1 < C_2$ and $C_1 \leq \beta C_{SKV}$ otherwise assign job i to F_2 if $C_2 \leq C_1$ and $C_2 \leq \beta C_{SKV}$. Update $C_{max} = \max(C_1, C_2)$.
   (b) If step 4(a) is not satisfied, job i is considered pending. Update PJ = PJ $\cup$ {i}; $N_{PJ} = N_{PJ} + 1$.

Stage 2 (Assignment of pending jobs to flow shops)

5. $\forall_i \in$ PJ and For $i = 1$ to N_{PJ} Do:
   - Using Johnson's rule, assign job i to F_1 if $C_1 < C_2$ otherwise assign job i to F_2. Update $C_{max} = \max(C_1, C_2)$

End.

5.2 Phase 2—Improvement Heuristic

The result of phase 1 is an assignment of all jobs to flow shops. The makespan is the largest of the partial makespans developed in Phase 1. Specifically, $C_{max} = \max(C_1, C_2)$.

In order to reduce the makespan, an improvement heuristic can be applied to Phase 1 schedule. In this application, the composite exchange heuristic

developed by Harriri and Potts [14] for scheduling unrelated parallel machines is modified and applied to Phase 1 schedule. The modified composite exchange heuristic consists of two stages. In the first stage, a job is removed from the flow shop that produces the largest makespan and is assigned to a flow shop that produces the lowest makespan. All jobs and all possible assignments are considered. This is called a *one to zero exchange*. In the second stage, two jobs are exchanged, one from the flow shop that produces the largest makespan and one from the flow shop that produces the lowest makespan. This is called a *one to one exchange* and all jobs are considered for the two flow shops. The first stage is applied first using the constructive heuristic schedule as input. Then using the resulting improved schedule as input to the second stage, a further reduction in makespan is attempted. The procedure continues by repeatedly applying the first stage and the second stage until no further reduction in makespan is possible. Stages 1 and 2 of the improvement heuristic proceed as follows:

5.2.1 Stage 1—One-to-zero exchange

1. Let flow shop A be the one that has the maximum makespan, i.e., $C_A = C_{max}$ and flow shop B be the one that has the minimum makespan.
2. Search for job $j \in F_A$, such that when assigning job j to F_B, $C_B < C_{max}$ and $C_A < C_{max}$
3. If no such job j is found, then C_{max} is the final solution.
4. If job j is found, then job j is assigned to flowshop B. Update $F_B = F_B \cup \{j\}$ and $F_A = F_A - \{j\}$.

The entire procedure is repeated searching for all one to zero exchanges until no further reduction in the maximum completion time is possible.

5.2.2 Stage 2—One-to-one exchange

1. Let flow shop A be the one that has the maximum makespan and the other flow shop B be the one that has the minimum makespan.
2. Search for jobs $j \in F_A$ and $i \in F_B$, such that when assigning job j to F_B and job i to F_A, $C_B < C_{max}$ and $C_A < C_{max}$
3. If jobs j and i cannot be found, then C_{max} is the final solution.
4. If jobs j and i are found, then j and i are interchanged. Update $F_B = F_B \cup \{j\} - \{i\}$ and $F_A = F_A \cup \{i\} - \{j\}$.

The entire procedure is repeated searching for all possible one to one exchanges until no further reduction in the maximum completion time is possible. If any exchanges are made in stage 2, stage 1 is repeated. If additional exchanges are made in stage 1, stage 2 is repeated. The process continues until no further exchanges are realized in one stage.

6. A Simulation Experiment

A simulation experiment is performed to evaluate the effectiveness of the proposed algorithm. The simulation consists of two stages. Stage 1 consists of 300 small size problems, with jobs ranging from five to fifteen, while stage 2 consists of 500 large size problems, with jobs ranging from twenty to one hundred. The proposed algorithm is compared with optimal solution values for small size problems and with the existing SKV algorithm for large size problems. Optimal solutions are obtained using AMPLE software as a modelling language and CPLEX 7.0 as a solver. All heuristics are programmed in Borland C++ and implemented on a Pentium III 650 personal computer.

6.1 Generation of Test Problems

The processing time is assumed to follow a discrete uniform distribution between 5 and 35, i.e., DU(5, 35). Two factors are considered: the number of jobs, n; and the processing time multiplier, α. For stage 1, the number of jobs are set at three levels, 5, 10, and 15; and the processing time multiplier is specified by α, and set at five levels, 1, 1.25, 1.5, 1.75, and 2. For stage 2, the number of jobs is set at five levels, 20, 40, 60, 80, and 100. The processing time multiplier is set in the same manner as in stage 1. Hence, 15 and 25 sets of problems are run in stages 1 and 2, respectively. For each set of problems in stages 1 and 2, 20 replications are considered. Thus a total of $3 \times 5 \times 20 = 300$, and $5 \times 5 \times 20 = 500$ problem instances were examined for stages 1 and 2, respectively.

6.2 Computational Results for Small Size Problems

Several comparisons for small size problems (stage 1) are performed to examine the overall performance of the partitioning flow shop heuristic. Specifically, the percentage deviations above optimal solution values before and after implementing Phase 2-improvement heuristic and the ability to obtain optimal solutions are examined.

Let Z_{dev} denote the percent deviation of the heuristic solution value above the optimal solution value and is used as a measure of performance for each problem instance where:

$$Z_{dev} = \frac{Heuristic\ \ Solution\ \ Makespan - Optimal\ \ Solution\ \ Makespan}{Optimal\ \ Solution\ \ Makespan} \times 100$$

The proposed partitioning flow shop heuristic is executed for 3 values of β starting from β = 0.5 with an increment of 0.2, and the least makespan is chosen for each replication. The results included in Table 2 are the mean percentage deviations above optimum over the 20 replications. Notice that for each replication the best solution (the minimum percentage deviation above optimum) among the three values of β (β = 0.5, 0.7, 0.9) is selected and the mean best-deviations over the 20 replications is calculated and presented in last column of Table 2. The summary results for the experiment included in Table 2 show that the overall mean percent deviations above optimum for the partitioning flow shop heuristic before implementing phase 2-improvement heuristic is equal to 1.870 %. This compares with 2.089 % for the existing SKV assignment comparison. The results in Table 2 also suggest that the average performance (over the 25 condition combinations) improves as the value of β increases.

This comparison clearly indicates that before implementing phase 2-improvement heuristic the new partitioning flow shop heuristic provides solutions that are relatively more effective than the existing SKV algorithm and are close to the optimal value.

Table 3 shows a clear superiority of the proposed PFH algorithm over the existing SKV algorithm when phase 2-improvement heuristic was implemented. Note that the value of the overall mean percent deviations above optimum for the proposed PFH algorithm decreases from 1.870% to 1.843% after implementing the improvement heuristic. It is obvious from Tables 2 and 3 that encouraging results can always be obtained when phase 2-improvement heuristic is applied after phase 1-constructive heuristic.

Table 2. Computational results: Mean percentage deviations above optimum before implementing the phase 2-improvement heuristic

#	N	m	SKV	PFH β=0.5	PFH β=0.7	PFH β=0.9	PFH Best β
1		1	1.85	1.91	2.01	1.86	1.79
2		1.25	1.87	1.81	1.97	1.88	1.80
3	5	1.5	1.91	1.97	1.89	1.62	1.61
4		1.75	1.88	1.89	1.99	1.51	1.46
5		2	2.01	2.56	3.02	1.88	1.79
	Average		1.904	2.028	2.176	1.75	1.69
6		1	1.94	1.99	1.89	1.88	1.83
7		1.25	1.95	1.89	1.86	1.83	1.80
8	10	1.5	1.89	1.97	2.31	1.52	1.48
9		1.75	2.11	3.21	3.21	1.99	1.92
10		2	2.21	3.41	3.12	2.01	1.99
	Average		2.02	2.494	2.478	1.846	1.804
21		1	2.12	2.11	2.13	2.13	2.04
22		1.25	1.98	2.32	1.89	1.79	1.71
23	15	1.5	1.99	1.95	1.78	1.96	1.70
24		1.75	2.42	3.15	3.48	2.43	2.42
25		2	3.21	4.01	3.44	2.96	2.72
	Average		2.344	2.708	2.544	2.254	2.118
	Overall Average		2.089	2.410	2.399	1.950	1.870

Table 3. Computational results: Mean percentage deviations above optimum after implementing the phase 2-improvement heuristic

#	N	m	SKV	PFH β=0.5	PFH β=0.7	PFH β=0.9	PFH Best β
1		1	1.85	1.85	1.83	1.82	1.75
2		1.25	1.87	1.80	1.88	1.85	1.77
3	5	1.5	1.91	1.89	1.84	1.61	1.58
4		1.75	1.88	1.85	1.83	1.50	1.42
5		2	2.01	1.99	2.00	1.87	1.75
	Average		1.904	1.876	1.876	1.73	1.654
6		1	1.94	1.92	1.85	1.81	1.78
7		1.25	1.95	1.87	1.84	1.81	1.79
8	10	1.5	1.89	1.88	1.88	1.51	1.46
9		1.75	2.11	2.10	2.10	1.95	1.91
10		2	2.21	2.18	2.19	1.99	1.97
	Average		2.02	1.99	1.972	1.814	1.782
21		1	2.12	2.10	2.11	2.11	2.02
22		1.25	1.98	1.97	1.85	1.72	1.69
23	15	1.5	1.99	1.92	1.74	1.94	1.67
24		1.75	2.42	2.41	2.41	2.40	2.39
25		2	3.21	3.20	2.99	2.94	2.70
	Average		2.344	2.32	2.22	2.222	2.094
	Overall Average		2.089	2.062	2.022	1.922	1.843

Figure 1 illustrates the relationship between the number of jobs and the overall average percentage deviation from optimal solution for the existing SKV algorithm and the proposed PFH algorithm before and after implementing phase 2-improvement heuristic.

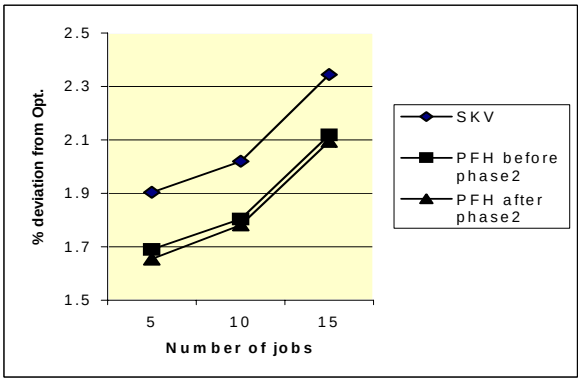

Figure 1. Comparative performance-% deviation above optimal

It is clear from Figure 1 that the proposed algorithm outperformed the existing SKV algorithm in all cases. Note that the CPU times are not reported since it took much less than a second for each problem instance to be solved.

In this computational study, the new heuristic algorithm obtained optimal solution values 106 times out of 300 generated test problems, 35.3 percent of the cases (see Table 4). Table 4 shows that the overall best percent deviation is 0% and the overall worst percent deviation is 3.01% among the 300 problem instances.

Table 4. Heuristic algorithm performance

Test #	N	m	# problem instances	# optimal solutions	Best deviation	Worst deviation
1		1	20	16	0	1.92
2		1.25	20	14	0	1.79
3	5	1.5	20	15	0	1.99
4		1.75	20	13	0	1.78
5		2	20	13	0	1.89
6		1	20	7	0	1.98
7		1.25	20	6	0	1.85
8	10	1.5	20	8	0	1.76
9		1.75	20	4	0	2.00
10		2	20	4	0	2.01
21		1	20	3	0	3.01
22		1.25	20	2	0	1.99
23	15	1.5	20	0	0.02	1.89
24		1.75	20	0	0.01	2.89
25		2	20	1	0	2.96

6.3 Computational Results for large Size Problems

For large size problems (stage 2), the proposed algorithm is only compared with the existing SKV algorithm. Tables 5 and 6 give stage 2 simulation results before and after implementing phase 2-improvement heuristic respectively.

Table 5. Computational results: Average makespan of the heuristic algorithms before implementing phase 2-improvement heuristic

#	N	m	SKV	PFH β=0.5	PFH β=0.7	PFH β=0.9	PFH Best β
1		1	219.7	217.2	216.7	216.6	216.6
2		1.25	246.1	243.55	243.7	243.425	242.95
3	20	1.5	262.35	258.95	257.7	258.2	257.05
4		1.75	277.475	272.35	271.825	273.725	270.7
5		2	289.8	286.2	285.9	285.6	283.6
	Avg.		259.085	255.65	255.165	255.51	254.18
6		1	420.7	418.7	418.9	419.1	418.7
7		1.25	467.25	465.675	465.75	465.4	465.15
8	40	1.5	518.6	515.65	515.6	516.3	514.95
9		1.75	531.1	526.35	526.675	526.375	525.325
10		2	561.8	557.9	557.1	557.3	556.2
	Avg.		499.89	496.855	496.805	496.895	496.065
11		1	657.8	654.7	654.6	654.6	654.6
12		1.25	684.2	682	681.875	681.875	681.975
13	60	1.5	742.55	740.35	740.7	740.75	739.9
14		1.75	802.45	797.9	797.825	797.9	796.525
15		2	841.3	836.1	835.7	835.9	834.5
	Avg.		745.66	742.21	742.14	742.205	741.5
16		1	837.8	836.7	836.7	836.7	836.7
17		1.25	927.775	925.825	925.525	925.925	925.325
18	80	1.5	1004.1	999.65	999.7	999.75	998.8
19		1.75	1041.1	1037.3	1037.88	1037.75	1036.57
20		2	1115	1111.3	1109.9	1110.4	1109.3
	Avg.		985.155	982.155	981.941	982.105	981.339
21		1	1016.5	1015.5	1015.4	1015.4	1015.4
22		1.25	1123.72	1121.4	1121.43	1121.18	1121.1
23	100	1.5	1237.3	1234.55	1234.35	1234.85	1234.1
24		1.75	1308.05	1304.18	1304.53	1304.68	1303.65
25		2	1387.1	1381.4	1382	1382.2	1380.4
	Avg.		1214.534	1211.406	1211.542	1211.662	1210.93
	Overall Average		740.8648	741.1754	741.25	741.0628	739.8658

Table 6. Computational results: Average makespan of the heuristic algorithms after implementing the phase 2-improvement heuristic

#	N	M	SKV	PFH β=0.5	PFH β=0.7	PFH β=0.9	PFH Best β
1		1	219.7	219.4	219.8	219.8	217.7
2		1.25	246.1	246.775	246.65	246	244.8
3	20	1.5	262.35	262.5	263.85	262.35	260.95
4		1.75	277.475	277.55	276.8	278.35	276
5		2	289.8	292.5	291.2	290.6	289.2
	Avg.		259.085	259.745	259.66	259.42	257.73
6		1	420.7	420.6	421.5	421.2	420
7		1.25	467.25	467.275	467.825	468.075	466.9
8	40	1.5	518.6	518.65	518.95	518.55	517.55
9		1.75	531.1	530.675	530.7	531.05	529.25
10		2	561.8	563	563	562.7	561.8
	Avg.		499.89	500.04	500.395	500.315	499.1
11		1	657.8	657.8	657.7	657.2	657.1
12		1.25	684.2	683.775	684.125	684.3	683.1
13	60	1.5	742.55	743.6	743.8	743.3	742.35
14		1.75	802.45	803.05	801.75	801.3	800.525
15		2	841.3	839.8	840.7	839.8	838.9
	Avg.		745.66	745.605	745.615	745.18	744.395
16		1	837.8	837.7	837.9	837.9	837
17		1.25	927.775	928.075	929.8	928.425	927.2
18	80	1.5	1004.1	1004.75	1004.4	1003.75	1002.45
19		1.75	1041.1	1042.18	1041.05	1042.3	1040.5
20		2	1115	1115.7	1116.5	1116.5	1114.9
	Avg.		985.155	985.681	985.93	985.775	984.41
21		1	1016.5	1016.2	1016.3	1017.2	1016.1
22		1.25	1123.72	1123.38	1122.95	1123.8	1122.32
23	100	1.5	1237.3	1238.25	1237.65	1237.6	1236.8
24		1.75	1308.05	1308.5	1308.45	1308.12	1307.55
25		2	1387.1	1387.7	1387.9	1386.4	1385.7
	Avg.		1214.534	1214.806	1214.65	1214.624	1213.694
	Overall Average		740.8648	737.6552	737.5186	737.6754	736.8028

From Tables 5 and 6, it is clear that the performance of the proposed algorithm compared well with the existing SKV algorithm before and after implementing phase 2-improvement heuristic. The overall average results of partitioning flow shop heuristic are superior compared to the exciting SKV heuristic specially when phase 2-improvement heuristic was applied. Table 7 illustrates the change in the objective function value between the existing SKV algorithm and the proposed PFH algorithm after implementing phase 2-improvement heuristic. On average, for all problem scenarios, the partitioning flow shop heuristic reduced the objective function value in all cases.

Table 7. SKV objective function performance comparison after implementing phase 2-improvement heuristic

	SKV Performance Comparison- % Deviation 100(SKV-PFH)/PFH			
Test #	PFH β=0.5	PFH β=0.7	PFH β=0.9	PFH Best β
1	1.151	1.384	1.431	1.431
2	1.047	0.984	1.098	1.296
3	1.312	1.804	1.607	2.061
4	1.881	2.078	1.369	2.502
5	1.257	1.364	1.470	2.186
6	0.477	0.429	0.381	0.477
7	0.338	0.322	0.397	0.451

8	0.572	0.581	0.445	0.708
9	0.902	0.840	0.897	1.099
10	0.699	0.843	0.807	1.006
11	0.473	0.488	0.488	0.488
12	0.322	0.340	0.340	0.326
13	0.297	0.249	0.242	0.358
14	0.570	0.579	0.570	0.743
15	0.621	0.670	0.646	0.814
16	0.131	0.131	0.131	0.131
17	0.210	0.243	0.199	0.264
18	0.445	0.440	0.435	0.530
19	0.366	0.310	0.322	0.437
20	0.332	0.459	0.414	0.513
21	0.098	0.108	0.108	0.108
22	0.206	0.204	0.226	0.233
23	0.222	0.238	0.198	0.259
24	0.296	0.269	0.258	0.337
25	0.412	0.369	0.354	0.485
Average	0.585	0.629	0.593	0.770

Finally, from a statistical perspective, the paired or matched samples test is used to test the significance of the difference between (SKV and PFH-β=0.5), (SKV and PFH-β=0.7), (SKV and PFH-β=0.9) and (SKV and PFH-Best) before and after implementing phase 2-improvement heuristic. The results of the matched samples test before and after implementing phase-2 are tabulated in Tables 8 and 9 respectively. The results of the matched sample test indicate that the matched samples test between (SKV and PFH-Best) is statistically significant at a significant level of 0.005 for the one-tail-test before and after implementing the improvement heuristic. Note that the matched sample tests between (SKV and PFH-β=0.5), (SKV and PFH-β=0.7), and (SKV and PFH-β=0.9) are also statistically significant after implementing the improvement heuristic. Thus, the proposed algorithm appears to perform better on average than the existing SKV algorithm.

Table 8. Results of matched samples test before implementing the phase 2-improvement heuristic

Paired tested	**Mean of paired differences**	**Standard deviation of paired differences**	**Z-*test* value**	**Test**
SKV vs. PFH (β=0.5)	-0.3106	0.786032	-6.844	Not significant
SKV vs. PFH (β=0.7)	-0.3852	0.774554	-8.636	Not significant
SKV vs. PFH (β=0.9)	-0.198	0.707768	-4.845	Not significant
SKV vs. PFH-Best	0.999	0.652618	25.512	Significant

Table 9. Results of matched samples test after implementing the phase 2-improvement heuristic

Paired tested	**Mean of paired differences**	**Standard deviation of paired differences**	**Z-*test* value**	**Test**
SKV vs. PFH (β=0.5)	3.2096	1.296546	42.875	Significant
SKV vs. PFH (β=0.7)	3.3462	1.401427	41.355	Significant
SKV vs. PFH (β=0.9)	3.1894	1.273914	43.363	Significant
SKV vs. PFH-Best	4.062	1.829694	38.451	Significant

7. Conclusions

A new polynomial-time algorithm called the partitioning flow shop heuristic (PFH) is developed to minimize the makespan on a two-stage parallel flow shops with proportional processing times. The portioning flow shop heuristic followed three steps to minimize the makespan. In the first step, a job selection criterion was used to partition the unassigned jobs into two sets, those that are assigned to specific flow shops, and those that are assigned as pending. In the second step, a suitable assignment rule was used to assign the pending jobs. In the third step, a job exchange rule was applied to improve the solution.

The proposed algorithm was tested on small size problems, up to 15 jobs, and on large size problems, up to 100 jobs. The performance of the heuristic algorithm, measured as the percentage deviation between heuristic solution values and optimal solution values for small size problems, was very satisfactory, yielding solutions within few percentage points of the optimal solution values. For large size problems, the proposed algorithm was compared with one of the best previously known algorithm called the SKV algorithm. The computational results showed that the proposed algorithm performed better than the existing SKV algorithm before and after implementing phase 2-improvement heuristic.

References

[1] Al-Kuwari, A., Al-Zara, I., Ashknani, H., and Salem, A, "Minimizing makespan in proportional parallel flowshops", the first international conference on information and systems, Cairo University, Egypt. 17-20 June, 2002.

[2] Arthanary, TS., and Ramaswamy, KG, "An extension of two machine sequencing problem", Journal of the Operational Research Society of India., Vol. 8. pp. 10-22, 1971.

[3] Baker, KR. Introduction of sequencing and scheduling. Wiley: New York, 1974.

[4] Barnes, JW., and Brennan, JJ, "An improved algorithm for scheduling jobs on identical machines", AIIE Transactions, Vol. 9. pp. 25-31, 1977.

[5] Daniel, S., Robert, L., and Bulfin, Jr. Production: planning, control, and integration. The McGraw-Hill Companies, Inc. New York. 1997.

[6] Davis, E., and Jaffe, J.M, "Algorithms for scheduling tasks on unrelated Processors", Journal of the Association for Computing Machinery., Vol. 28. pp. 721-736, 1981.

[7] De, P., and Morton, T, "Scheduling to minimize makespan on unrelated processor",

Decision Sciences., Vol. 11. pp. 586-602, 1980.

[8] Dogramaci, A., and Surkis, "Evaluation of a heuristic for scheduling independent jobs on parallel identical processors", Management Science., Vol. 25. pp. 1208-1216, 1979.

[9] Elmaghraby, SS., and Park, "Scheduling jobs on a number of identical machines", AIIE Transactions., Vol. 5. pp. 1-13, 1974.

[10] Garey, M. R., and Johnson. Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman and Company. 1979.

[11] Gonzales, T., Ibarra, OH., and Sahni, S, "Bounds for LPT schedules on uniform processors", Computer Science Tech. Rep. No. 75-1. University of Minnesota: Minneapolis, MN. 1974.

[12] Gonzales, T., and Sahni, S, "Preemptive scheduling of uniform processor systems", Journal of Association for Computing Machinery., Vol. 25, pp. 92-101, 1978.

[13] Gupta, JND, "Two-stage hybrid flow-shop scheduling problem", Journal of Operational Research Society., Vol. 39. pp. 359-364, 1988.

[14] Hariri, A., and Potts, C, "Heuristics for scheduling unrelated parallel machines", Computers and Operations Research., Vol. 18. pp. 323-331, 1991.

[15] Horowitz, E., and Sahni, S, "Exact and approximation algorithms for non-identical processors", Journal of Association for Computing Machinery., Vol. 23. pp. 317-327, 1976.

[16] Ibarra, O., and Kim, C, "Heuristic algorithms for scheduling independent tasks on nonidentical processors", Journal of the Association for Computing Machinery., Vol. 24, pp. 280-289, 1977.

[17] Johnson, SM, "Optimal 2- and 3-stage production schedules with setup times included", Naval Research Logistics., Q1, pp. 61-68, 1954.

[18] Liu, JWS., and Liu, CL, "Bounds on scheduling algorithms for heterogeneous computing systems", In Proceedings, IFIPS Congress 74, North Holland, Amsterdam. pp. 349-353, 1974.

[19] Martello, S., Soumis, F., and Toth, P, "Exact and approximation algorithms for makespan minimization on unrelated parallel machines", Discrete applied mathematics., Vol. 75. pp. 169-188, 1997.

[20] McNaughton, R, "Scheduling with deadlines and loss functions", Management Science., Vol. 6, pp. 1-12, 1959.

[21] Narasimhan, SL., and Panwalkar, SS, "Scheduling in a two-stage manufacturing process", Int J Prod Res., Vol. 22, pp. 555-564, 1984.

[22] Potts, C.N, "Analysis of a linear programming heuristic for scheduling unrelated parallel Machines", Discrete Applied Mathematics., Vol. 10, pp. 155-164, 1985.

[23] Rao, TBK, "Sequencing in the order A, B with multiplicity of machines for a single operation", Journal of the Operational Research Society of India., Vol. 7, pp. 135-144, 1970.

[24] Rothkopf, MH, "Scheduling independent tasks on parallel processors", Management Science., Vol. 12, pp. 437-447, 1966.

[25] Sahni, S, "Algorithms for scheduling with independent tasks", Journal of Association for Computing Machinery., Vol. 23, pp. 116-127, 1976.

[26] Sundaraghavan, PS., Kunnahur, AS., and Viswanathan, I, "Minimizing makespan in parallel flowshops", Operational Research Society., Vol. 48, pp. 834-842, 1997.

[27] Van de Velde S.L, "Duality-based algorithms for scheduling unrelated machines", ORSA Journal of Computing., Vol. 5, pp. 192-205, 1993.

[28] Vel, H. V. D., and Shijie, S, "An application of the bin-packing technique to job scheduling on uniform processors", Operational Research Society., Vol. 42, pp. 169-172, 1991.