Research Support
School of Business Administration

June 1994

A NOTE ON FLEXIBLE FLOWSHOP SCHEDULING WITH TWO STAGES

Working Paper #9409-15

Jacek Blazewicz and G. Pawlak
Technical University of Poznan
and
Moshe Dror
University of Arizona
and
Kathryn E. Stecke
University of Michigan

A Note on Flexible Flowshop Scheduling with Two-Stages

J.Błażewicz[*] M.Dror[†] G.Pawlak[*] K.E.Stecke[‡]

Abstract

With the increasing emphasis on flexible manufacturing systems, there is a need to examine methods for their analysis and modeling. In this note we consider tandem two-stage flexible flow shop systems (FFS), where at each stage there may be more than one machine executing operations. We focus on the makespan objective for which this scheduling problem is NP-hard in the strong sense. We propose a heuristic based on Johnson's algorithm, and prove that its worst-case behavior is at most 2. We also compare the mean performance of this heuristic with a number of other simple scheduling algorithms, which proves a superiority of the proposed algorithm.

[*]Technical University of Poznań, Institute of Computing Science, Poznań, Poland

[†]The University of Arizona, MIS Department, Tucson,USA

[‡]The University of Michigan, School of Business Administration, Ann Arbor, USA

1 Introduction

An important problem that needs to be addressed in the operation of a flexible manufacturing system (FMS) is the scheduling of parts on machines (Stecke [11] and Kusiak [9]). As the machines become more versatile, most of the operations on a part can be accomplished by just one or two machine types (Jaikumar [7]). One type of an FMS used in practice is a flexible flow shop system (FFS - Sriskandarajah, Sethi [10]), where the routing of operations is unidirectional (Błażewicz at al. [1]). As opposed to the classical flow shop, an FFS may have more than one machine for at least one stage, processing operations of different parts in parallel. One FMS like this is described in Błażewicz at al. [2]. In this last paper, a model of an FMS has been introduced which enabled one to analyze a factory producing parts for helicopters. In this model, the first stage corresponded to versatile (identical in the sense of processing capabilities) machines and the second stage corresponded to an inspection stage (which can be modeled by one machine). The algorithms presented dealt with part scheduling and vehicle routing assuming the second stage can be postponed. In the present paper, we remove this assumption and study scheduling algorithms for a two stage system.

Recently, Hoogeveen at al. [6] proved that the problem of scheduling parts in such a system in order to minimize schedule length is NP-hard in the strong sense. This means that in general, even a pseudo-polynomial-time optimization algorithm is not likely to solve the problem. (For FFS like above a branch & bound algorithm minimizing the maximum completion time was developed by Brah and Hunsucker [3]). This indicates the usefulness of the development of polynomial-time heuristic algorithms along with the evaluation of their accuracy. The first work in this direction has been done by Sriskandarajah and Sethi [10], who analyzed flow shop problems with two stages. The first stage contained either one machine or the same number of machines as the second stage. They analyzed the worst case behavior of several heuristic algorithms.

In this paper, we investigate heuristic algorithms for the special open case of the above model with parallel machines at the first stage, thus complementing the above results. Here, the flexible flow shop consists of two machine stages $[S_1, S_2]$, with stage S_1 having $m_1 \geq 2$ parallel machines and S_2 having $m_2 = 1$ machine. This problem setting examination is scheduling algorithm based on Johnson's procedure [8]. This procedure's worst case behavior is proven to equal two. Subsequently, three additional simple heuristics are evaluated experimentally. These tests prove the superiority of the first algorithm.

We proceed with a formal problem description. There are n parts, T_j, to be processed in up to two operations in the FFS. For each part, there is specified a processing time vector $\overline{p_j} = [p_{j1}, p_{j2}]$, where p_{j1} and p_{j2} denote the operation processing times of part T_j in stages S_1 and S_2, respectively. The operations are nonpreemptable and it is assumed that the buffers at each of the machine stages have sufficient capacity to hold all parts waiting for processing. The optimality criterion here is *schedule length* (makespan), $C_{max} = \max_j\{c_j\}$, where c_j is a completion time of part T_j . According to the notation given by Graham at al. [5] and modifications made by Sriskandarajah and Sethi [10], this problem is denoted as follows:

$$F2(m_1 = m, m_2 = 1) \parallel C_{max}.$$

The outline of this paper is as follows. In Section 2, the first algorithm based on Johnson's procedure is presented, and its worst case performance is analyzed. In Section 3, other algorithms are presented and the average performance of all the algorithms is examined empirically. Conclusions are provided in Section 4.

2 Algorithm 1

For the scheduling problem in question the following algorithm based on Johnson's approach [8], is proposed.

Algorithm 1

STEP 1. Choose those parts for which the operation times $p_{j1} \leq p_{j2}$. Schedule these parts on the m_1 machines at stage S_1 in non-decreasing order of their processing times, p_{j1} .

STEP 2. The remaining parts schedule on the machines at stage S_1 in non-increasing order of their processing times, p_{j2} .

STEP 3. Schedule all of these parts on the single machine at stage S_2 in order of their completion at stage S_1 .

since there are more machines then one at this stage, it may fail to find an optimal schedule. The worst case behavior of Algorithm 1 is stated in Theorem 1.

Theorem 1.

For the problem, $F2(m_1 = m, m_2 = 1) \parallel C_{max}$ $(m \geq 2)$. Let C_{max} be the schedule length after applying Algorithm 1 and C^*_{max} be the optimal schedule length. Then

$$\frac{C_{max}}{C^*_{max}} \leq 2$$

and this bound is the best possible.

Proof: It is rather obvious that an optimal schedule length cannot be less than the following bounds:

$$C^*_{max} \geq \min_j\{p_{j1}\} + \sum_{j=1}^{n} p_{j2}. \tag{1}$$

$$C^*_{max} \geq C^*_1 + \min_j\{p_{j2}\} \tag{2}$$

where C^*_1 is an optimal schedule length at stage S_1 .

We consider two main cases, depending on the value of the schedule length produced by Algorithm 1. Each of these cases contains two subcases.

Let $\min_j\{p_{j1}\} + \sum_{j=1}^{n} p_{j2} = x_1$; $\max_j\{p_{j1}\} = x_2$; $\min_j\{p_{j1}\} = \delta$.

From (1) and (2) we see that:

$$C^*_{max} \geq x_1 \tag{3}$$

$$C^*_{max} \geq x_2. \tag{4}$$

Case 1. The schedule length C_{max} satisfies the following relation:

$$C_{max} \leq \max_j\{p_{j1}\} + \sum_{j=1}^{n} p_{j2}. \tag{5}$$

Case 2. An inverse inequality holds

$$C_{max} > \max_j\{p_{j1}\} + \sum_{j=1}^{n} p_{j2}. \tag{6}$$

Now, we will analyze the two cases separately.

Case 1.

Define two subcases

Case 1.1: $x_1 \geq x_2$

Case 1.2: $x_1 < x_2$

Case 1.1. From (5) we have:

$$C_{max} \leq x_2 + (x_1 - \delta).$$

From (3) we have

$$C^*_{max} \geq x_1.$$

Combining these two inequalities we get

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{x_2 - \delta}{x_1} \leq 2,$$

since δ can be made arbitrarily small.

Case 1.2. From (5) we have:

$$C_{max} \leq x_2 + (x_1 - \delta).$$

From (4) we have

$$C^*_{max} \geq x_2.$$

Thus,

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{x_1 - \delta}{x_2} \leq 2.$$

□

Case 2.

Consider now the second case, i.e., inequality (6) is fulfilled. Denoting $x_3 = \sum_{j=1}^{n} p_{j2}$, we have

$$C_{max} > x_2 + x_3 \tag{7}$$

Again two subcases may occur.

Case 2.1. We have

$$x_3 \geq x_2. \tag{8}$$

We will prove first that in this case schedule length is bounded from above by $C_1^* + x_3$.

Proposition 1

If $C_{max} > x_2 + x_3$ and $x_3 \geq x_2$, then

$$C_{max} \leq C_1^* + x_3. \tag{9}$$

Proof: Two extreme subcases depend on the schedule obtained at the first stage (S_1) of the flowshop system.

a) The schedule at S_1 is an optimal one (cf Fig.1).

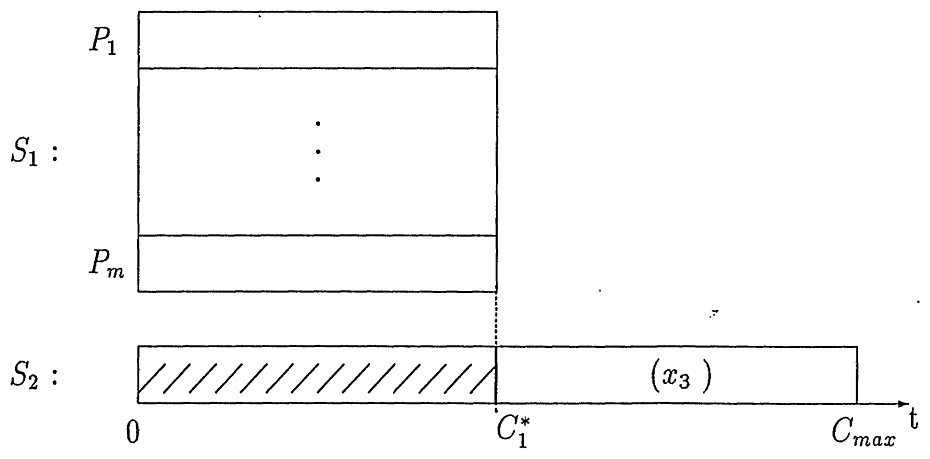

Figure 1.

Clearly, even in the worst case at the second stage (as depicted in Fig. 1) inequality (9) must be fulfilled.

b) The schedule at S_1 is the worst possible, i.e., C_1 is the longest.
Taking into account the worst possible behavior of a schedule for parallel processors, as analyzed by Graham [4], the longest operation (of length x_2) will be processed at the end of stage S_1 . (If the longest operation at S_1 belonged to a part fulfilling a condition of Step 1

of Algorithm 1, then it could not be scheduled at the end of stage S_1 according to Algorithm 1. Therefore the schedule at S_1 would not be the longest possible.) Following Algorithm 1 and its second step, a corresponding operation of the same part is the shortest (of length p_{q2}) from among those scheduled in this step at stage S_2 (cf. Fig. 2), i.e., $p_{q2} = \min_j\{p_{j2} : p_{j2} < p_{j1}\}$.

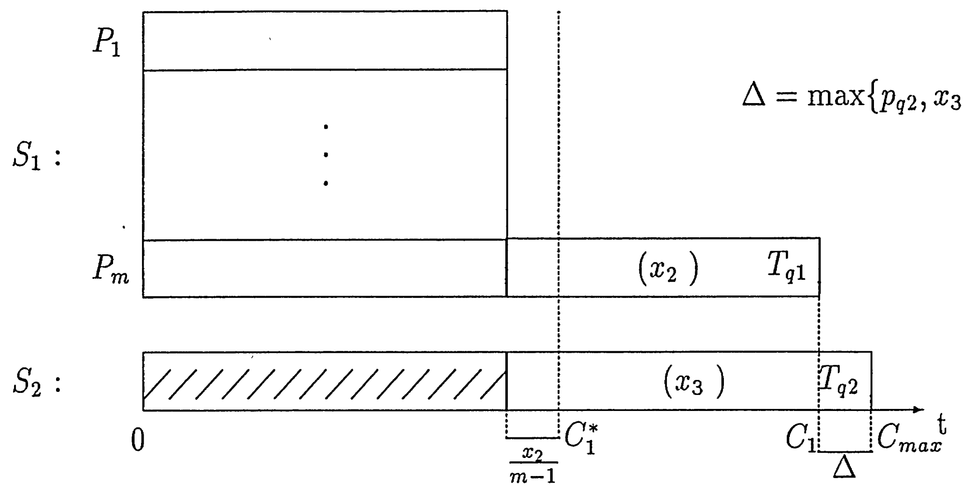

Figure 2.

Since the mean value by which an optimal schedule length C_1^* is decreased on $m-1$ processors at S_1 is $\frac{x_2}{m-1}$, we can assume that the processing of operations starts at stage S_2 at moment $C_1^* - \frac{x_2}{m-1}$ at the latest.

If $x_3 - x_2 \geq p_{q2}$, then no idle time at S_2 appears beyond $C_1^* - \frac{x_2}{m-1}$ and we have:

$$C_{max} \leq C_1^* - \frac{x_2}{m-1} + x_2 + x_3 - x_2 < C_1^* + x_3. \tag{10}$$

Otherwise (if $x_3 - x_2 < p_{q2}$), an idle time at S_2 appears and

$$C_{max} \leq C_1^* - \frac{x_2}{m-1} + x_2 + p_{q2}. \tag{11}$$

Then again, if $p_{q2} < \frac{x_2}{m-1}$, (11) may be written as

$$C_{max} \leq C_1^* + x_3$$

since $x_3 \geq x_2$.

On the other hand, i.e., if $p_{q2} \geq \frac{x_2}{m-1}$, operations T_{j2} assigned to the machine at stage S_2 according to Step 2 must be longer than T_{q2} (or at least equal to). Thus, $p_{j2} \geq \frac{x_2}{m-1}$ for these operations and the sum of their processing requirements is equal at least to x_2 , since there are at least $m-1$ such operations. (If there were less then $m-1$ operations of this type then inequality (7) could not be satisfied and we would have Case 1.) Hence, the idle time at S_2 doesn't appear beyond moment $C_1^* - \frac{x_2}{m-1}$, $p_{q2} \leq x_3 - x_2$ and schedule length C_{max} is constrained as in inequality (10). This completes the proof of case b.

Any intermediate case (between extreme cases a and b) will not lengthen the schedule beyond bound $C_1^* + x_3$. For any other assignment of tasks to processors at stage S_1 will shorten C_1 and thus C_{max} . (If another long task is assigned to P_{m-1} , we can repeat the above reasoning for the set of processors $P_1, ..., P_{m-1}$.) This completes the proof of Proposition 1.

□

Following Proposition 1, we can now estimate the ratio $\frac{C_{max}}{C_{max}^*}$. For $x_3 \leq C_1^*$, we have

$$\frac{C_{max}}{C_{max}^*} \leq \frac{C_1^*+x_3}{C_1^*+\min_j\{p_{j2}\}} \leq 2,$$

where the bound C_{max}^* was given in formula (2).

On the other hand, if $x_3 > C_1^*$ then

$$\frac{C_{max}}{C_{max}^*} \leq \frac{C_1^*+x_3}{x_3+\min_j\{p_{j1}\}} \leq 2$$

and the bound for C_{max}^* was given in (1).

This completes the proof for Case 2.1.

Case 2.2. We have

$$x_3 < x_2 \tag{12}$$

In this case we will prove that the schedule length is bounded from above by $C_1^* + x_2$.

Proposition 2.2

If $C_{max} \leq x_2 + x_3$ and $x_3 < x_2$, then

$$C_{max} \leq C_1^* + x_2. \tag{13}$$

Proof: Again, as in Proposition 1 we will distinguish two extreme cases depending on the schedule at stage S_1 .

a) The schedule at S_1 in an optimal one (cf Fig. 1).
It is obvious that in this case

$$C_{max} \leq C_1^* + x_3 < C_1^* + x_2.$$

b) The schedule at S_1 is the worst possible and we have the situation depicted in Fig.2, but now $\Delta = p_{q2}$.

We see that

$$C_{max} \leq C_1^* - \frac{x_2}{m-1} + x_2 + p_{q2}. \tag{14}$$

Assuming now that $p_{q2} \geq \frac{x_2}{m-1}$, Algorithm 1 would assign (at Step 2) operations in nonincreasing order of their processing times p_{j2} , which therefore must be greater than or equal to p_{q2}. The total processing requirement of these operations is at least

$\frac{x_2}{m-1}(m-1) = x_2.$

This contradicts (12) and thus

$$p_{q2} < \frac{x_2}{m-1}. \tag{15}$$

Using (15) in evaluating C_{max} (14), we get

$$C_{max} \leq C_1^* + x_2.$$

This completes the proof of case b. Any intermediate case can be handled analogously as in Proposition 1.

□

Now, the estimation of the worst case bound $\frac{C_{max}}{C^*_{max}}$ is as follows. If $x_2 \leq C_1^*$ we have

$$\frac{C_{max}}{C^*_{max}} \leq \frac{C_1^*+x_2}{C_1^*+\min_j\{p_{j2}\}} \leq 2;$$

otherwise

$$\frac{C_{max}}{C^*_{max}} \leq \frac{C_1^*+x_2}{x_2+\min_j\{p_{j2}\}} \leq 2,$$

because $C^*_{max} \geq x_2 + \min_j\{p_{j2}\}$.

This completes the proof of Case 2.2.

Now we will show that the above bound is achievable.

Example:

We define two types of parts with processing times, respectively:

1) $\overline{p_a} = [m, \frac{1}{m}]$, $\qquad a = 1, ..., m;$

2) $\overline{p_b} = [\frac{1}{m}, \frac{1}{m} - \varepsilon]$, $\qquad b = 1, ..., mm;$

A schedule constructed by Algorithm 1 is shown in Figure 3.

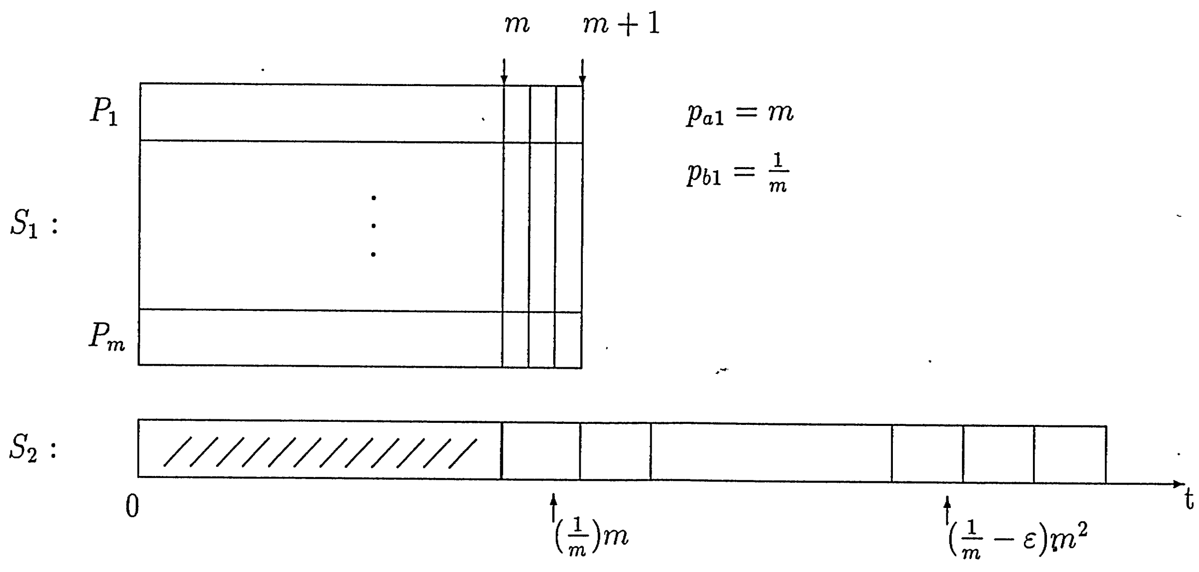

Figure 3. A schedule constructed by Algorithm 1 for the Example.

We see that for this schedule,

$$C_{max} = 2m + 1 - m^2\varepsilon. \tag{16}$$

On the other hand, an optimal schedule is shown in Fig. 4.

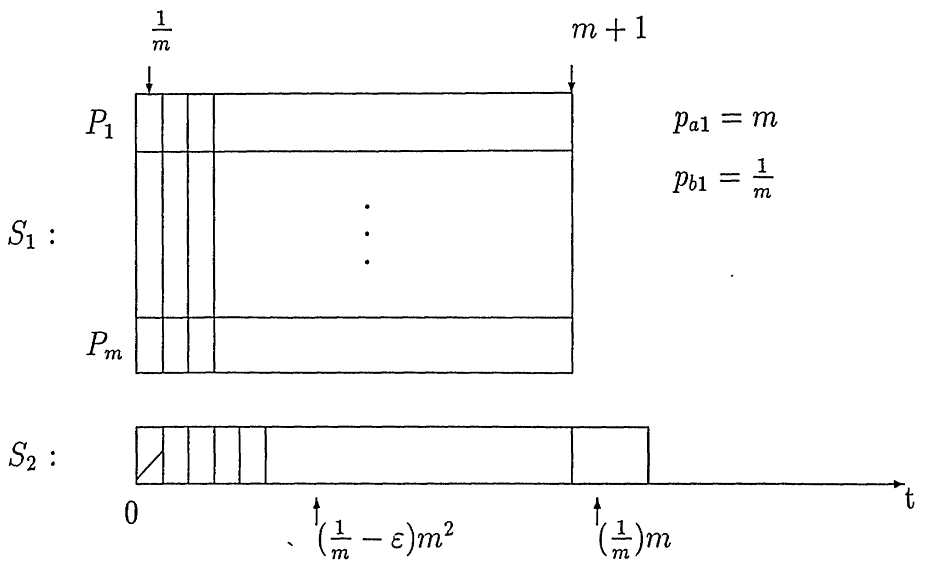

Figure 4. An optimal schedule for the Example.

Thus, the optimal schedule length is

$$C^*_{max} = \frac{1}{m} + m + 1 - m^2\varepsilon. \tag{17}$$

Hence, dividing equation (16) by equation (17), we have:

$$\frac{C_{max}}{C^*_{max}} = 1 + \frac{1-\frac{1}{m^2}}{1+\frac{1}{m^2}+\frac{1}{m}-m\varepsilon} \leq 2 \qquad \text{for } m \to \infty$$

and $\varepsilon \leq \frac{1}{m^2} + \frac{2}{m^3}$

This completes the proof of Theorem 1.

□

As will be shown in the next Section, mean behavior of Algorithm 1 is better then 2. Its superiority over some other simple heuristics will be also shown.

3 Computational Experiments

To test the mean behavior of Algorithm 1, an extensive experiment has been carried out. Besides Algorithm 1, three other algorithms have been tested. They have the same framework, but a different ordering of tasks at stage S_1 . Their action is described below.

Algorithms 2,3,4

STEP 1. Schedule tasks on machines at stage S_1 in SPT (respectively, LPT, random) order of their processing times, p_{j1} .

STEP 2. Schedule all of these tasks on the single machine at stage S_2 in order of their completion at stage S_1 .

These four algorithms have been tested experimentally. The computational data are described as follows:

- All of the parameters are randomly generated using a uniform distribution with the following ranges:

  1. number of tasks n (ranges are given in Tables 1 and 2)
  2. processing times. Two subcases have been distinguished:
     - arbitrary generation of processing times (see Table 1) p_{j1} and p_{j2}, in the range of 1 to 10 and 1 to 100, respectively;
     - processing times at stage S_2 depended on the number of machines at stage S_1 (see Table 2), p_{j1}, in the range of 1 to 10 and 1 to 100; and $p_{j2} = \frac{p_{j1}}{m_1}$.

- The number of problem instances generated for each parameter range was 1000.

- The computer installation was a Hewlett-Packard 9000/750.

The results are gathered in Tables 1 and 2. In these tables, the numbers of instances generated for each range of parameters are equal to 1000. For each algorithm and each parameter range, a number of cases is shown for which a solution generated is equal to the optimum (the parameter best). Moreover, mean values over 1000 instances of the ratio $C^{A}_{max}/C^{LB^*}_{max}$ are reported (parameter mean), where $C^{LB^*}_{max}$ is a lower bound for the optimal schedule length (obtained as a maximum from equations (1) and (2), where $C^*_1 \geq \max\{\max_j\{p_{j1}\}, \frac{1}{m}\sum_{j=1}^{n} p_{j1}\}$). We see that for an arbitrary generation of processing times of parts, the machine at the second stage becomes a bottleneck and all of the algorithms become almost optimal and the

Table 1: The ratio $C^{A}_{max}/C^{LB^*}_{max}$ for four algorithms and arbitrary processing times of parts at both stages.

No	m_1	n	$p_{j1,2}$	Alg. 1.		A. 2(SPT)		A. 3(LPT)		A. 4(RND)	
				best	mean	best	mean	best	mean	best	mean
1	2	2-20	1-10	960	1.005	963	1.005	9	1.122	55	1.101
2	2	2-20	1-100	949	1.006	959	1.005	7	1.120	56	1.098
3	2	2-50	1-10	962	1.006	968	1.005	15	1.132	205	1.087
4	2	2-50	1-100	911	1.012	933	1.007	16	1.143	89	1.101
5	2	2-100	1-10	975	1.002	977	1.004	7	1.117	135	1.088
6	2	2-100	1-100	956	1.005	960	1.005	5	1.120	51	1.100

* best - the number of optimal instances for which an algorithm produces an optimal schedule.

ratio hardly exceeds 1.3, even for $m_1 = 2$. However, if the processing times at the second stage depend on a number of machnes at the first stage (a more realistic case), then the algorithms behave quite differently. The best percentage of solutions close to optimum have Algorithms 1 and 2 (SPT). Algorithm 2(SPT), however, generates worse schedules (as in the case of LPT) for the case in which the first operations are equal or almost equal. The worst is Algorithm 3 (LPT); it is even worse than a random (arbitrary) assignment of parts to machines at stage S_1. This could be expected, because LPT assignment at the first stage increases idle time at the second stage.

Table 2: The ratio $C^{A}_{max}/C^{LB^*}_{max}$ for four algorithms and processing times of parts at stage S_2 depending on the number of machines at stage S_1 .

No	m_1	n	p_{j1}	Alg. 1.		A. 2(SPT)		A. 3(LPT)		A. 4(RND)	
				best	mean	best	mean	best	mean	best	mean
1	2	2-20	1-10	458	1.060	378	1.079	7	1.302	12	1.132
2	2	2-20	1-100	456	1.058	376	1.076	8	1.302	17	1.130
3	2	2-50	1-10	685	1.040	566	1.059	20	1.271	645	1.136
4	2	2-50	1-100	437	1.070	348	1.092	15	1.314	35	1.158
5	2	2-100	1-10	778	1:025	680	1.039	8	1.256	50	1.116
6	2	2-100	1-100	465	1.059	380	1.075	6	1.301	14	1.130
7	5	5-20	1-10	493	1.087	569	1.082	47	1.291	218	1.141
8	5	5-20	1-100	112	1.203	84	1.267	12	1.388	28	1.268
9	5	5-50	1-10	759	1.038	814	1.034	16	1.252	174	1.112
10	5	5-50	1-100	458	1.060	378	1.079	7	1.302	12	1.132
11	5	5-100	1-10	456	1.058	376	1.076	8	1.302	17	1.130
12	5	5-100	1-100	685	1.040	566	1.058	20	1.271	64	1.136

* p_{j2} - is randomly generated from 1 to p_{j1}/m_1 for all instances.

4 Summary

In this paper, a special case of the flow shop scheduling problem has been analyzed in which the first stage contains parallel identical machines, while the second stage contains only one. An algorithm based on Johnson's strategy has been proposed and its worst case behavior proved to be 2. Then, three other very simple heuristics have been proposed and all of the algorithms have been tested experimentally. The mean behaviors of Algorithm 1 (based on Johnson's approach) and of Algorithm 2 (based on the SPT rule) were the best and close to optimal, with a superiority of Algorithm 1 for the case of a non-bottleneck second stage.

Further investigation should include different criteria (for example, mean weighted completion time or tardiness). Other details that could be included are the finite buffers between stages or automated guided vehicle routing.

ACKNOWLEDGMENTS

The authors gratefully acknowledge helpful comments made by Erwin Pesch on an earlier draft of this paper. Kathy Stecke would like to acknowledge a summer research grant from the Business School of The University of Michigan.

References

[1] Błażewicz, J., Ecker, K., Schmidt, G., and Węglarz, J., *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag, New York, Berlin, 1993.

[2] Błażewicz, J., Eiselt, H., Finke, G., Laporte, G., and Węglarz, J., "Scheduling tasks and vehicles in a flexible manufacturing systems", *International Journal of Flexible Manufacturing Systems* 4 (1991), pp. 5-16.

[3] Brah, S.A., and Hunsucker, J.L., "Branch and bound algorithm for the flow shop with multiple processors", *European Journal of Operational Research* 51 (1991), pp. 88-99.

[4] Graham, R.L., "Bounds for certain multiprocessors anomalies", *Bell Systems Technical Journal* 45/9 (1966), pp. 1563-1581.

[5] Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G., "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics* 5 (1979), pp. 287-294.

[6] Hoogeveen, J.A., Lenstra, J.K., and Veltman, B., "Minimizing makespan in a multiprocessor flowshop is strongly NP-hard", Working Paper, Centre for Mathematics and Computer Science, Amsterdam, 1992.

[7] Jaikumar, R., "Postindustrial manufacturing", *Harvard Business Review* 64 (1986), 6.

[8] Johnson, S.M., "Optional two- and three-stage production schedules", *Naval Research Logistics Quarterly* 1 (1954), pp. 61-68.

[9] Kusiak, A., and Wilhelm, E., "Analysis modeling and design of modern production system", *Annals of Operation Research* 17 (1989), J.C.Baltzer, Basel.

[10] Sriskandarajah, C., and Sethi, S.P., "Scheduling algorithms for flexible flowshops: worst and average case performance", *European Journal of Operational Research* 43 (1989), pp. 143-160.

[11] Stecke, K.E., "Design, planing, scheduling and control problems of flexible manufacturing systems", *Annals of Operations Research* 3 (1985), pp. 3-12.