Division of Research
School of Business Administration
The University of Michigan

December 1992

SCHEDULING PARTS THROUGH A TWO-STAGE TANDEM FLEXIBLE FLOW SHOP

J. Blazewicz
Technical University of Poznan
Moshe Dror
The University of Arizona
G. Pawlak
Technical University of Poznan
K.E. Stecke
The University of Michigan

Working Paper No. 699

ABSTRACT

With the increasing installation of flexible manufacturing systems, a need exists to investigate methods for their analysis and modeling. The type of FMS considered here is a tandem flexible flow system (FFS), in which at each stage there may be more than one machine executing operations. Here, we investigate input sequencing heuristics into a tandem two-stage FFS for the performance measure of minimizing makespan. As this scheduling problem with the schedule length criterion is NP-hard, one should investigate heuristic algorithms. For these, we estimate their worst-case, as well as mean, behavior. In this paper, an open NP case of the flexible flow shop problem is studied for which four heuristic algorithms are proposed and analyzed.

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 [1985]). As the machines become more complicated, the FMSs are versatile, so that most of the operations on a part can be accomplished by just one or two machine types (Jaikumar [1986]). One type of an FMS used in practice is a flexible flow system (FFS), where the routing of operations is unidirectional.

As opposed to the classical flow shop, an FFS may have more than one machine at each stage, processing operations of different parts in parallel. Recently, Hoogeveen and Lenstra [1992] prove 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, not even a pseudo-polynomial-time optimization algorithm is likely to be able to solve the problem. This indicates the usefulness of the development of some polynomial-time heuristic algorithms along with the evaluation of their accuracy. The first work in this direction has been done by Sriskandarajah and Sethi [1989], who analyze flow shop problems with two stages. The first stage contains either one machine or the same number of machines as the second stage. They analyze the worst case behavior of several heuristic algorithms.

Different input sequencing problems of FFSs are addressed by Kubiak and Sethi [1992] and Smith and Stecke [1991]. The former finds JIT sequences to balance workloads. The latter determines cyclic input sequences that balance workloads.

In this paper, we investigate heuristic algorithms for the special open case of the flow shop scheduling problem with parallel machines at the first stage. Here, the flexible flow shop consists of 2 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. One real-life FMS like this is described in Błażewicz et al. [1991].

We begin in Section 2 by analyzing a scheduling algorithm based on Johnson's procedure [1954], which minimizes schedule length for a two-machine problem. The heuristic's worst case behavior is proven to be equal to two. Then three additional simple heuristics are presented. These algorithms are evaluated experimentally. Next we define the problem formally.

There are n parts, T_j, j=1,...,n, to be processed in up to two operations in the FFS. For each part, there is specified a processing time vector $\bar{p}_j = [p_{j1}, p_{j2}]$, where p_{j1} and p_{j2} denote the operation processing times of part T_j at S_1 and S_2, respectively. The parts are independent and the operations are nonpreemptable. The optimality criterion here is *minimizing the schedule length*, $C_{max} = \max_j\{c_j\}$, where c_j is a completion time of part T_j. According to notation proposed by Graham et al. [1979], with extensions given by Sriskandarajah and Sethi [1989], this problem is denoted as follows:

$$\mathbf{F2} \;\; \Big| \;\; \mathbf{m_1 = m,\; m_2 = 1} \;\; \Big| \;\; \mathbf{C_{max}}.$$

In this note, buffers at the machine stages have sufficient capacity to hold all parts waiting for processing.

The outline of this paper is as follows. In Section 2, an algorithm based on Johnson's algorithm is presented, and its worst case performance is analyzed. In Section 3, three other algorithms are presented and the average performances of all of the algorithms are tested. A summary is provided in Section 4.

2. ALGORITHM 1

The following heuristic Algorithm 1, which is based on Johnson's algorithm [1954], finds a schedule of possibly minimal length.

<u>Algorithm 1</u>

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

STEP 2. The remaining parts are scheduled 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.

We see that the algorithm uses Johnson's strategy to sort parts at the first stage. However, since there are more than one machine at this stage, it may fail to find an optimal schedule. The worst case behavior of Algorithm 1 is analyzed in the following Theorem 1.

Theorem 1. For the problem, $\mathbf{F2} \mid \mathbf{m_1 = m \geq 2,\ m_2 = 1} \mid \mathbf{C_{max}}$, let p_{j1} and p_{j2}, j = 1...n, be the part processing times on machines at stages S_1 and S_2, respectively. Let C_{max} be the schedule length after applying Algorithm 1 and C^*_{max} be the optimal schedule length. Then

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

this bound is the best possible.

Proof: A *lower bound* on the optimal schedule length cannot be less than the following:

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

where $C^{*'}_{1max}$ is a *lower bound* on an optimal schedule length *for stage S_1*:

$$C^{*'}_{1max} = \max\{\frac{1}{m} \sum_{j=1}^{n} p_{j1},\ \max_j\{p_{j1}\}\}.$$

Condition (1) describes the only two possible cases for the worst schedule at either the first or the second stage. Of course, we can define other terms to describe the optimal schedule length (for example, $\max_j\{p_{j1} + p_{j2}\}$). Then, for some instances we could get better results than using condition (1) only. However, all other cases reduce to condition (1) for Algorithm 1 within the bound of value 2.

If we can find any parts that satisfy the inequality from STEP 1 of Algorithm 1, then the schedule produced by Algorithm 1 would not generate the worst case. The worst case occurs when we cannot find parts which satisfy the inequality from STEP 1. Because parts are assigned in non-increasing order of p_{j2}, in the worst case we can first schedule the longest operation time parts (p_{j1}) at stage S_1, which would give the maximum idle time at the beginning of the second stage. This situation is described by equation (3) and is illustrated by Example 1.

On the other hand, if the longest operation time part can be scheduled at the end of stage S_1 and if the processing times, p_{j2}, of the parts are small, then this provides the second case of the worst schedule. This situation is described by equation (9) and is illustrated by Example 2. All other cases can be reduced to these two cases.

Now we consider the two possible cases that can follow from equation (1).

Case 1:

Assume first that the lower bound is

$$C^{*'}_{\max} = \min_j \{p_{j1}\} + \sum_{j=1}^{n} p_{j2}. \tag{2}$$

We see that it is the best lower bound of the schedule length. From Algorithm 1, the worst case occurs when the schedule length $C_{\max}$ fulfills the following equation:

$$C_{\max} = \max_j \{p_{j1}\} + \sum_{j=1}^{n} p_{j2}. \tag{3}$$

This is the upper bound of the schedule length for Case 1. This is because if the $\max_j \{p_{j1}\}$ is scheduled at the beginning of the schedule, then this gives the maximum idle time at the beginning at stage S_2.

Hence

$$C_{\max} = C^{*'}_{\max} + \max_j \{p_{j1}\} - \min_j \{p_{j1}\}.$$

Thus

$$\frac{C_{\max}}{C^{*'}_{\max}} \leq 1 + \frac{\max_j \{p_{j1}\} - \min_j \{p_{j1}\}}{\min_j \{p_{j1}\} + \sum_{j=1}^{n} p_{j2}}. \tag{4}$$

Let us denote:

$$\max_j \{p_{j1}\} = B \text{ and } \min_j \{p_{j1}\} = \varepsilon.$$

We can assume without loss of generality for n>m that

$$\sum_{j=1}^{n} p_{j2} \geq B. \tag{5}$$

If condition (5) were not true, then Case 1 would reduce to Case 2.

We now have

$$\frac{C_{max}}{C^{*'}_{max}} = 1 + \frac{B-\varepsilon}{B+\varepsilon} < 2.$$

Thus

$$\frac{C_{max}}{C^{*}_{max}} < 2, \text{ and this is the worst bound in this case.}$$

Now we show an example for which $\frac{C_{max}}{C^{*}_{max}}$ approaches 2.

Example 1:

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

$$\bar{p}'_j = \left[m-1, \frac{1}{m-1}\right], \qquad j = 1 \ldots m;$$

$$\bar{p}''_j = \left[\frac{1}{m}, \frac{1}{m} - \varepsilon\right], \qquad j = 1 \ldots m^2;$$

and $\varepsilon < \frac{1}{m^3}$.

A schedule constructed from this data by Algorithm 1 is shown in Figure 1.

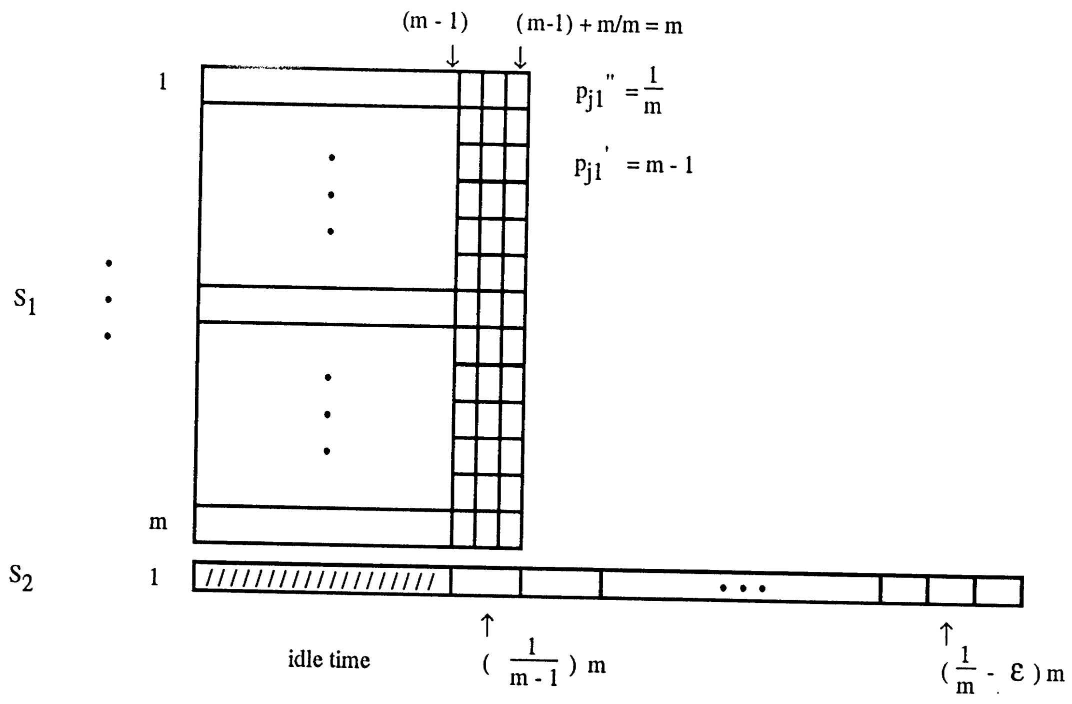

Figure 1. Schedule Constructed by Algorithm 1, for Example 1.

We see that for this schedule,

$$C_{\max} = m - 1 + m\left(\frac{1}{m-1}\right) + \left(\frac{1}{m} - \varepsilon\right)m^2$$

$$= 2m + \frac{m}{m-1} - 1 - m^2\varepsilon. \tag{6}$$

On the other hand, an optimal schedule for this data is shown in Figure 2.

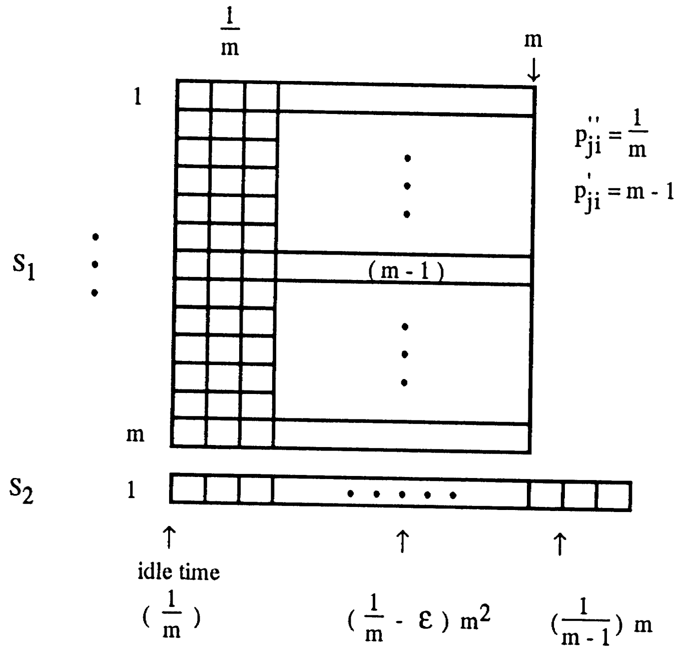

Figure 2. The Optimal Schedule for Example 1.

Thus, the optimal schedule length is

$$C^*_{max} = \frac{1}{m} + \left(\frac{1}{m} - \varepsilon\right)m^2 + \left(\frac{1}{m-1}\right)m$$

$$= \frac{1}{m} + m - m^2\varepsilon + \frac{m}{m-1}. \tag{7}$$

Dividing equation (6) by equation (7), we have:

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

As $\varepsilon \to 0$, then

$$\frac{C_{max}}{C^*_{max}} = \frac{2m + \frac{m}{m-1} - 1}{m + \frac{1}{m} + \frac{m}{m-1}} = \frac{2m + \frac{1}{m-1}}{m + \frac{m}{m-1} + \frac{1}{m}} \to 2, \text{ as } m \to \infty.$$

Case 2:

Case 2 occurs when the second part of equation (1) is greater than the first part. We then have:

$$C^{*'}_{max} = C^{*'}_{1max} + \min_j \{p_{j2}\}. \tag{8}$$

In the worst case, a schedule length, C^*_{1max}, at stage S_1 (a list schedule) is related to an optimal schedule as follows (see Graham [1966]):

$$C_{1max} = \left(2 - \frac{1}{m}\right)C^*_{1max}.$$

In this worst case, from Algorithm 1, we have:

$$C_{max} = \left(2 - \frac{1}{m}\right)C^*_{1max} + \min_j \{p_{j2}\}. \tag{9}$$

This is the worst case of Algorithm 1 for Case 2, where the largest part ($\max_j \{p_{j1}\}$) is scheduled at the end of Stage 1.

On the other hand, we have:

$$C^{*'}_{1\max} \le C^{*}_{1\max} .$$

Dividing equation (9) by equation (8), we have:

$$\frac{C_{\max}}{C^{*'}_{\max}} \le \frac{\left(2-\frac{1}{m}\right)C^{*'}_{1\max} + \min_j \{p_{j2}\}}{C^{*'}_{1\max} + \min_j \{p_{j2}\}} .$$

Let us denote $\min_j \left\{p_{j1}\right\} = \varepsilon$. If $\varepsilon \to 0$, then

$$\frac{C_{\max}}{C^{*'}_{\max}} \le 2 - \frac{1}{m}.$$

From equation (1), we know that $C^{*'}_{\max} \le C^{*}_{\max}$, and thus

$$\frac{C_{\max}}{C^{*}_{\max}} \le 2 - \frac{1}{m}.$$

Example 2:

We define two types of parts whose processing times are:

$p'_j = [1, 2\varepsilon], \quad j = 1 \ldots m(m - 1);$

$p''_j = [m , \varepsilon], \quad j = 1;$

and $\varepsilon \le \frac{1}{bm}$, where $b > 2$.

In Figure 3, the worst case behavior of Algorithm 1 is shown.

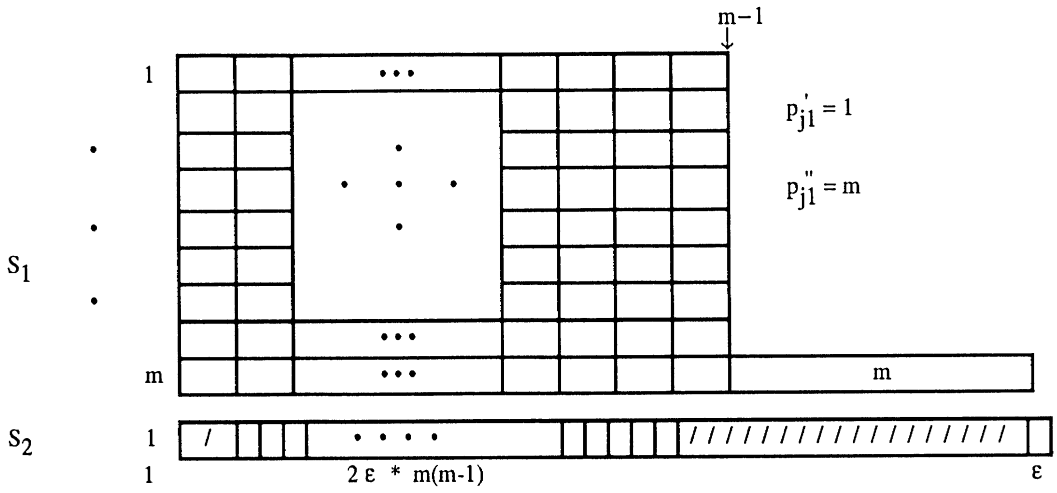

Figure 3. Schedule for Example 2 Developed from Algorithm 1.

If $\varepsilon = \frac{1}{bm}$, then $2\varepsilon * m(m - 1) < m$, and from Figure 3, we have

$$C_{max} = m - 1 + m + \varepsilon = 2m - 1 + \varepsilon. \tag{10}$$

An optimal schedule is shown in Figure 4.

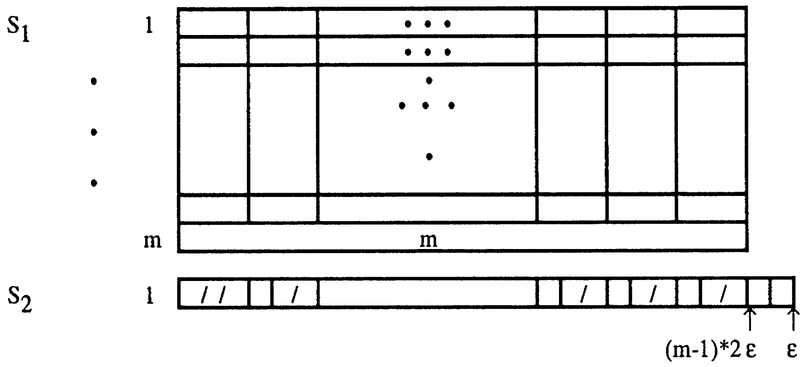

Figure 4. Optimal Schedule for Example 2.

The optimal schedule length is:

$$C^*_{max} = m + 2\varepsilon\,(m - 1) + \varepsilon.$$

Dividing equation (10) by equation (11), we obtain

$$\frac{C_{\max}}{C^*_{\max}} = \frac{2m-1+\varepsilon}{m+2\varepsilon(m-1)+\varepsilon} = \frac{2m-1+\frac{1}{bm}}{m+\frac{2}{b}-\frac{2}{bm}+\frac{1}{bm}} .$$

As $b \to \infty$, then $\varepsilon \to 0$. Then for $m \to \infty$,

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

Hence from Case 1 and Case 2, we conclude that the worst case performance for the considered Algorithm 1 is:

$\boldsymbol{\frac{C_{\max}}{C^*_{\max}} \leq 2}$, and this bound is the best possible. ■

3. COMPUTATIONAL EXPERIMENTS

In addition to Algorithm 1, three other algorithms are tested. They have a similar framework, but a different ordering of the operations. These algorithms are now described:

Algorithms 2, 3, and 4

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

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

These four algorithms have been tested experimentally. These computational experiments are now described as follows.

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

1. The number of operations, n, is from 1 to 1000.
2. The number of machines at the first stage, m, is from 1 to 50.
3. Two cases of processing times are distinguished:

a) Processing times are generated randomly according to p_{j1} from 1 to 1000 and p_{j2} from 1 to 1000; or

b) The processing times at stage S_2 depend on the number of machines at stage S_1 as follows: p_{j1} from 1 to 1000 and p_{j2} from 1 to 1000/m.

Other data of the computational experiments are as follows:

- The total number of problems generated is 50,000.
- The computer installation used is a SUN station IPC.
- The ratio, $C^A_{max}/C^{*'}_{max}$, is evaluated, where $C^{*'}_{max}$ is a lower bound on the optimal schedule length, C^*_{max} and C^A_{max} is the performance of Algorithm A, where A is 1, 2, 3, or 4.

The results are provided in Tables 1 and 2. In these tables, the numbers of cases are shown in which the ratio, $C^A_{max}/C^{*'}_{max}$, has been reached for each of the four algorithms. The best results are highlighted in boldface.

Table 1.

The Ratio $C^A_{max}/C^{*'}_{max}$ for the Four Algorithms and Arbitrary Processing Times of Parts at Both Stages.

	Ratio: Heuristic/Best Lower Bound			
	Algorithm 1	Alg. 2 (SPT)	Alg. 3 (LPT)	Alg. 4 (RND)
Mean	**1.008**	**1.006**	1.0979	1.0880
Std. Dev.	0.0104	0.0096	0.0153	0.0330
Best[1]	**49,647**	**49,726**	1,139	6,051

1. This is the number of times that each Algorithm obtained the best lower bound out of all 50,000 problems tested.

Table 2.

The Ratio $C^A_{max}/C^{*'}_{max}$ for the Four Algorithms and Processing Times of Parts at Stage S2 Depending on the Number of Machines at Stage S1.

	Ratio: Heuristic/Best Lower Bound			
	Algorithm 1	Alg. 2 (SPT)	Alg. 3 (LPT)	Alg. 4 (RND)
Mean	**1.1112**	1.1129	1.337	1.1330
Std. Dev.	0.0950	0.0950	0.0797	0.0833
Best[1]	**9,392**	9,168	111	115

1. This is the number of times that each Algorithm obtained the best lower bound out of all 50,000 problems tested.

We see from Table 1 that for an arbitrary generation of processing times of operations of parts, the machine at the second stage becomes a bottleneck. All of the algorithms become nearly optimal and the ratio nearly never exceeds 1.3. Algorithm 1 and the modified SPT heuristic are both by far the best.

However, when the processing times at the second stage depend on the number of machines at the first stage, then the algorithms behave quite differently. The best percentage of solutions close to optimum are still Algorithm 1 and a close second, Algorithm 2 (modified SPT). The standard deviation for none of the algorithms is high. The worst algorithm in both situations is Algorithm 3 (LPT). LPT is even a bit worse than a random assignment of operations to the machines at stage S_1. This could be expected, because an LPT assignment at the first stage can increase the idle time at the second stage.

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 machine. An algorithm based on Johnson's strategy has been proposed and its worst case behavior is shown to be 2. Then, three other simple heuristics have been proposed and all four 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.

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

ACKNOWLEDGMENTS

The authors gratefully acknowledge helpful comments made by Ervin 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

Błażewicz, J., H. A. Eiselt, G. Finke, G. Laporte, and J. Weglarz, "Scheduling Tasks and Vehicles in a Flexible Manufacturing System," International Journal of Flexible Manufacturing Systems, Vol. 4, No. 1, pp. 5-16 (August 1991).

Hoogeveen, J. A., J. K. Lenstra, and B. Veltman, "Minimizing Makespan in a Multiprocessor Flowshop is Strongly NP-hard," Working Paper, Eindhoven University of Technology, The Netherlands (1992).

Jaikumar, R., "Postindustrial Manufacturing," Harvard Business Review, Vol. 64, No. 6 (November-December 1986).

Johnson, S. M., "Optimal Two- and Three-stage Production Schedules with Set-up Times Included," Naval Research Logistics Quarterly, Vol. 1, pp. 61-68 (1954).

Kubiak, W. and S. P. Sethi, "Optimal Just-In-Time Schedules for Flexible Transfer Lines," Working Paper, Memorial University of Newfoundland, Faculty of Business Administration, St. John's, Canada, A1B 3X5 (July 1992).

Graham, R. J., "Bounds for Certain Multiprocessing Anomalies," Bell Systems Technical Journal, Vol. 45, No. 9 (November 1966).

Graham, R. J., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnoy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey," Annals of Discrete Mathematics, Vol. 5, pp. 287-294 (1979).

Smith, T. M. and K. E. Stecke, "On the Robustness of Using Balanced Part Mix Ratios to Determine Cyclic Part Input Sequences Into Flexible Flow Systems," Working Paper, No. 658, Division of Research, The University of Michigan, School of Business, Ann Arbor, MI (May 1991).

Sriskandarajah, C. and S. P. Sethi, "Scheduling Algorithms for Flexible Flowshops: Worst and Average Case Performance," European Journal of Operational Research, Vol. 43, pp. 143-160 (1989).

Stecke, K. E., "Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems," Annals of Operations Research, Vol. 3, pp. 3-12 (1985).