Scheduling in Network Flow Shops

Reza H. Ahmadi

Revised September 1995

Anderson Graduate School of Management at UCLA
405 Hilgard Avenue
Los Angeles, CA 90024-1481
Email: rahmadi@agsm.ucla.edu

ABSTRACT

We consider the general problem of static scheduling of a set of jobs in a network flow shop. In network flow shops, the scheduler not only has to sequence and schedule but also must concurrently determine the process routing of the jobs through the shop. In this paper, we establish the computational complexity of this new class of scheduling problem and propose a *general purpose heuristic procedure*. The performance of the heuristic is analyzed when makespan, cycle time and average flow time are the desired objectives.

Key Words: Scheduling, Routing, Network, Flow Shops, Heuristics, Error Bounds

* This research has been supported by the UCLA Academic Senate Grant #95.

I. Introduction

The general network flow shop problem can be stated as follows. Given a directed acyclic graph G=(M, E), where M identifies the set of processors (machines) and E denotes the set of links that connect the processors, the problem is to decide on process routing for the jobs and construct a schedule such that a desired regular performance measure is optimized. There are n jobs available at time zero. Each job $i \in N$ (i=1, ..., n) would require p_{ij} unit of processing time on machine j=1, . . ., m=|M|, if the processing path (routing) designated for job i requires machine j. The processing paths to complete the job requirements are assumed to be simple paths. We assume the execution times are nonnegative integers. The n jobs may be processed in any order through the network on the machines, but no two of them may be executed at the same time on any machine. A job consists of a sequence of operations given by a path in the network G. All jobs are assumed to be processed without preemption. A schematic representation of a network flow shop along with a sample processing path, indicated by double lines, is shown in Figure 1.1.

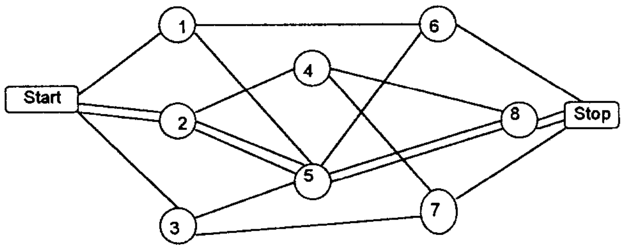


Figure 1.1: Schematic Representation of a Network Flow Shop

Traditionally the scheduling literature has assumed a fixed process routing for the jobs. In other words the set of machines to be visited by each job is assumed to be known in advance and prior to the onset of operations, although the processing sequence of jobs could be similar (in case of flow shops), or different (job shops), or arbitrary (open shops). Recent growth in development and adoption of Flexible Manufacturing Systems (FMS's) has introduced many challenging dimensions to the classical questions asked in the scheduling problems. An important feature of FMS's is in their flexibility, not only in manufacturing different parts and products with diverse features, but also processing the jobs in different processing sequences and at different workstations. This aspect of a job's process flexibility, where choice of the job's routing could affect the total processing time of the job, has not been addressed in the deterministic scheduling literature.

The problem of sequencing and scheduling in a *generalized flow shop* has been the subject of considerable study by many researchers in recent years. Graves et al. [4] addressed the problem of scheduling a set of jobs in re-entrant flow shops, where a job may return one or more times to any processor for additional processing. Examples of such shops are found in flexible machining systems and integrated circuit fabrication processes. Wittrock [20] and Sriskandarajah and Sethi [16] looked at the problem of scheduling parts in flexible flow lines, where several identical machines process parts at each processing stage. Lawler et. al [7], as well as many other researchers, have looked at the problem of scheduling jobs in shops with parallel machines. In these systems, the scheduler needs to decide on the allocation of the jobs to the machines, but each job requires only one operation. The network flow shop problem considered in this paper generalizes these instances of classical scheduling problems.

The remainder of this paper is organized as follows. In Section 2 we propose a general purpose heuristic for the static network flow shop problem. Section 3 establishes the computational complexity of the network flow shop with the objective of minimizing the makespan, and provides several tight worst case relative error bounds. In Section 4 we analyze the complexity and the worst case error bound of the proposed heuristic for minimizing the cycle time in a repetitive manufacturing environment. Section 5 concentrates on the analysis of the average flow time. In Section 6, we develop a polylogarithmic algorithm for the makespan problem. Section 7 reports the result of our computational experiments. In Section 8 we summarize our results and conclude the paper.

2. Myopic Shortest Path Heuristic

In this section we present a general heuristic, referred to as Myopic Shortest Path (MSP), to find the near optimal solution for the network flow shop problem under various objectives. Unlike the classical approaches in scheduling problems, we do not attempt to design a procedure suited solely for a given objective function; rather, we develop a heuristic procedure which is quite robust under various completion time based performance measures.

The MSP heuristic consists of two parts. In the first part, the procedure defines the order for releasing jobs to the shop. In the second part, the procedure concurrently defines the routing and scheduling of the jobs while in the shop. The heuristic constructs a partial schedule, $PS(\sigma_K)_j$ on each machine j, at each iteration. $PS(\sigma_K)_j$ defines the schedule of job set $K \subset N$ on machine j and $(\sigma_K)_j$ denotes the sequence and the set of jobs that visit machine j. The heuristic may be stated as follows:

Myopic Shortest Path Procedure:

Step 1. Compute the shortest processing route for each job i=1, ..., n, assuming complete machine availability.

Step 2. Reindex the jobs such that $C_1^* \leq C_2^* \leq ... \leq C_n^*$, where C_i^* denotes the length of the shortest processing path for job i.

Step 3. Set $K=\{1\}$ for machines visited by job 1, for all other machines $K=\{\emptyset\}$, define $(\sigma_K)_j$ and $PS(\sigma_K)_j$ for $j=1,...,m$, and set $k=2$.

Step 4. (i) Find the shortest route for job k given the partial schedule $PS(\sigma_K)_j$, $j=1,2, ..., m$.

(ii) Set $K=K+\{k\}$ for machines that are visited by job k. Update $(\sigma_K)_j$ and $PS(\sigma_K)_j$ to incorporate the schedule of job k on each machine it will visit.

(iii) Let C_k denote the completion time of job k on the last machine on its route.

Step 5. Set $k=k+1$ and go to Step 4 if $k \leq n$, else stop.

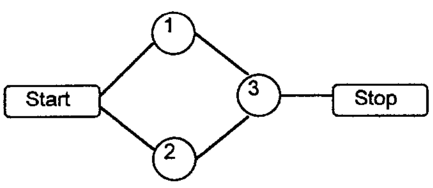
Step 4 of the procedure requires evaluation of the shortest path through a network given the partial schedule $PS(\sigma_K)_j$ of jobs on machine j. The partial schedule defines the available and occupied time slots on each machine. To perform this step, the dynamic program for shortest path can simply be revised to find the earliest time that a job can be processed on machine j without interrupting the processing of the other jobs in the partial schedule. The MSP heuristic requires $O(n^3)$ steps of computation.

3. The Makespan Problem

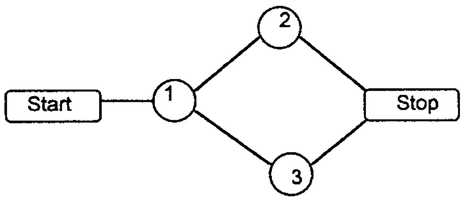
In this section we establish the computational complexity of the network flow shop problem when the objective is to minimize the makespan. Moreover, we provide several worst case relative error bounds for the MSP heuristic under various network configurations.

Proposition 3.1: Minimizing makespan in the network flow shop is unary NP-hard.

Proof: We complete the proof by establishing the computational complexity of two basic network structures. The first one is the *rooted tree* and the second one is an *assembly tree*.



An Assembly Tree



A Rooted Tree

<u>*Case I:*</u> In this case we consider a rooted tree with m=3. We transform a general instance of the 3-Partition problem into an instance of the network flow shop problem. The 3-Partition problem is

defined as follows. A finite set A of 3n elements, a bound $B \in Z^+$, and a "size" $a_i \in Z^+$ for each i=1, ..., 3n, such that $B/4 < a_i < B/2$ and $\sum_{i=1}^{3n} a_i = nB$. The instance of rooted tree flow shop is defined as:

$p_{i1} = a_i$, $p_{i2} = 2nB + B$, $p_{i3} = 2a_i$, for $i = 1, \dots, 3n$.
$p_{i1} = B$, $p_{i2} = 2B$, $p_{i3} = 2nB$ for $i = 3n+1, \dots, 4n-1$.
$p_{4n,1} = B$, $p_{4n,2} = B$, $p_{4n,3} = 2nB$.

$p_{4n+1,1} = B$, $p_{4n+1,2} = 0$, $p_{4n+1,3} = 0$.
$p_{4n+2,1} = 0$, $p_{4n+2,2} = 0$, $p_{4n+2,3} = B$.
$p_{4n+3,1} = 0$, $p_{4n+3,2} = 2B$, $p_{4n+3,3} = 0$, and $Y = B(2n+1)$.

There exists an schedule with makespan of B(2n+1) if and only if the set A can be partitioned into n disjoint sets A_j such that $\sum_{a_i \in A_j} a_i = B$ for each $1 \le j \le n$. Given Figure 3.1, if the 3-Partition has a solution then network flow shop has a makespan of B(2n+1). Note that the routing choices are easily

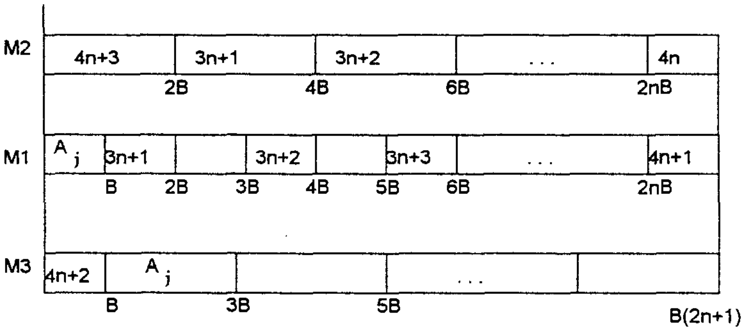


Figure 3.1

defined, due to considerable processing time differences. Conversely, if the makespan of any schedule is less than Y=B(2n+1), we claim that: (i) job 4n+3 is scheduled first on machine 2; (ii) job 4n+2 is scheduled first on machine 3; (iii) job 4n+1 is scheduled last on machine 1; (iv) the schedule of jobs on machine 1 should alternate between a subset of jobs from job 1 through 3n and a job from 3n+1 through 4n, starting with jobs in the first class, similar to the schedule in Figure 3.1.

If 3-Partition does not exists, therefore there are at least two subsets of jobs 1 through 3n such that $\sum_{a_i \in A_r} a_i < B$ and $\sum_{a_i \in A_j} a_i > B$. The schedule of these jobs would create idle time on machines 1 or 3, resulting in a contradiction. We now turn to the proof of (i), (ii), (iii), and (iv).

(i), [ii] If job 4n+3 (4n+2) is not scheduled first, then idle time would be created on machine 2 (3), causing a contradiction.

(iii) Similarly, if job 4n+1 is not scheduled last on machine 1, idle time would be created on machine 2 and 3.

(iv) Note that if a job from 3n+1 to 4n is not scheduled in every 2B time interval on machine 1, there would be idle time on machine 2. Conversely, if subsets of jobs from set A are not scheduled in the same time interval, a delay is caused in machine 3, resulting in a contradiction. Furthermore, it is easy to see that the schedule on machine 1 should start with the subset of jobs from set A.

Case II: In this case we establish the computational complexity of an assembly tree with m=3. The proof is based on the following lemma, which generalizes a well-known property in the flow shop scheduling problem.

Lemma 3.2: The problem of minimizing makespan in a rooted tree is equivalent to minimizing makespan in an assembly tree.

Proof: Given any rooted tree problem with m=3, a mirror image assembly tree can be constructed by changing the direction of flows in the graph. For every path in the rooted tree we have a corresponding path in the assembly tree. It is well known that minimizing the makespan in a flow shop is equivalent to minimizing the makespan in its mirror image problem. Since this result holds for every path, therefore we conclude the lemma.

From Case I and Lemma 3.2 it is evident that minimizing makespan in an assembly tree is equivalent to minimizing makespan in a rooted tree. This completes the proof of the proposition. □

The network flow shop problem remains NP-complete even when the processing times of jobs are *proportionate*. In an exact proportionate network flow shop, a constant of proportionality may be associated with each machine (Ow [12]). If the first machine has a constant of proportionality equal to one, and the jth machine has a constant of proportionality equal to α_j, then a job of size p will have the processing times p, $\alpha_2 p$, ..., $\alpha_m p$ on machine 1 through machine m. Consequently, the processing times of each job operation are proportionate. The following proposition establishes the complexity of the proportionate network flow shops.

Proposition 3.3: Minimizing makespan in the proportionate network flow shop is NP-hard.

Proof: We show the proof for the assembly tree case. The rooted tree case can be shown from Lemma 3.2. Partition can be reduced in polynomial time to an instance of a proportionate assembly tree flow shop. Partition may be described as follows. Given a finite set A of size n, with elements $a_i \in Z^+$ for each i=1, ..., n, is there a subset $B \subseteq A$ such that $\sum_{a_i \in B} a_i = \sum_{a_i \in A-B} a_i$? The instance of proportionate network flow shops is defined such that: $p_{i1} = p_{i2} = a_i$, $p_{i3} = \frac{a_i}{M} = \varepsilon_i$ for i = 1,...,n.

The elements have been reindexed such that $\varepsilon_1 \le \varepsilon_2 \le \ldots \le \varepsilon_n$. ε_i is a sufficiently small positive number and M is a very large number. Note that machine 3 is dominated, and if Partition has a solution then the proportionate network flow shop has makespan of smaller than or equal to $\sum_{a_i \in B} a_i + 2\varepsilon_n$, (see Figure 3.2). Conversely if the makespan of any schedule is less than or equal to $\sum_{a_i \in B} a_i + 2\varepsilon_n$, then clearly partition should exist. □

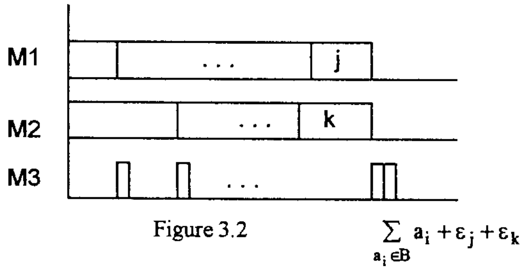


Figure 3.2

Next we address the computational complexity of minimizing makespan in a network flow shop with a *no-wait* restriction.

Proposition 3.4: Minimizing makespan in the network flow shop with a no-wait restriction is unary NP-hard.

Proof: 3-Partition reduces to an instance of a no-wait rooted tree flow shop problem. We define an instance of a no-wait rooted tree flow shop as follows:

$p_{i1} = a_i$,	$p_{i2} = 0$,	$p_{i3} = 0$,	for $i = 1, \ldots, 3n$.
$p_{i1} = B$,	$p_{i2} = 2B+1$,	$p_{i3} = H$,	for $i = 3n+1, \ldots, 4n-1$.
$p_{i1} = 0$,	$p_{i2} = 2B+1$,	$p_{i3} = 0$,	$i = 4n$.
$p_{i1} = 1$,	$p_{i2} = H$,	$p_{i3} = 2B+1$,	for $i = 4n+1, \ldots, 5n-1$.
$p_{i1} = 1$,	$p_{i2} = H$,	$p_{i3} = B$,	$i = 5n$.
$p_{i1} = B$,	$p_{i2} = 0$,	$p_{i3} = 0$,	$i = 5n+1$.
$p_{i1} = 0$,	$p_{i2} = 0$,	$p_{i3} = B+1$,	$i = 5n+2$. And $Y = n(2B+1)$.

If 3-Partition has a solution, then there exists a schedule with makespan of Y=n(2B+1). Figure 3.3 shows the corresponding schedule. We note that the selection of route is easily identifiable because of the significant processing time differences in the routes, since H is a very large number.

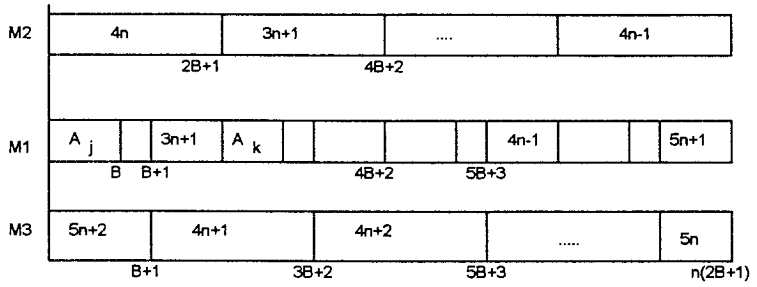


Figure 3.3

Conversely, if there is a schedule with makespan of less than or equal to n(2B+1), we claim that (i) job 4n (5n+2) should be scheduled first on machine 2 (3); (ii) job 5n+1 should be scheduled last on machine 1; (iii) the job's schedule on machine 1 consists of cycles of a subset of jobs from 3-Partition elements, a job from 3n+1 to 4n-1 and a job from 4n+1 to 5n, similar to the schedule given in Figure 3.3. It follows that if 3-Partition does not exist then the schedule on machine 1 would not be packed, and some idle time would be created, resulting in a contradiction. Now we consider the proof of (i), (ii), and (iii).

(i) These jobs only require processing on machines 1 and 3, respectively. Delaying their processing will result in idle time on the corresponding machines, therefore causing a contradiction.

(ii) Scheduling this job earlier would cause idle time on machines 1 and 3.

(iii) It is easy to see that other forms of schedules on machine 1 would create unnecessary idle time on machine 2 or 3. □

In real-life applications, it is often possible to control the processing time of jobs by managing the resources required to execute the tasks. However, shorter durations can only be achieved at some additional costs. This phenomenon is commonly recognized in the area of project scheduling, where the time-cost trade-offs are considered. This notion adds an additional dimension to the scheduling problem in which the processing times are also decision variables. The problem of minimizing makespan with controllable processing time in network flow shops can be stated as:

$$Z(x,\pi) = \text{makespan} + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

where $x \varepsilon X$ is the set of all feasible compressions, $\pi \varepsilon \Pi$ is the set of all schedules on the machines, c_{ij} is the cost of compressing job i at machine j, and x_{ij} is the amount of compression. Compressed processing times are $p_{ij} \Leftarrow p_{ij} - x_{ij}$.

Proposition 3.5: Minimizing makespan in the network flow shop with controllable processing time is NP-complete.

Proof: The network flow shop is a generalization of the two machine flow shop. Nowicki and Zdrazlka [10] show that the decision form of the problem is NP-complete. The problem remains NP-complete, even when the processing times on the machines are fixed and all the processing cost units are similar.

Now, we focus our attention on analyzing the error bound of the MSP heuristic. In the following, we let Z^{MSP} and Z^* be the makespan obtained by the MSP heuristic and the optimal schedule, respectively.

Proposition 3.6: ***The relative error bound of the MSP heuristic is*** $\frac{Z^{MSP}-Z^*}{Z^*} \leq m-1$. ***There are network flow shops for which this is the best possible bound.***

Proof: If all the processing times are equally divided among the m machines in the network flow shop, we have $Z^* \geq \frac{\sum_{i=1}^{n} C_i^o}{m}$ and if jobs are processed one at a time in the shop, then we get $Z^{MSP} \leq \sum_{i=1}^{n} C_i^o$;

therefore: $\frac{Z^{MSP}-Z^*}{Z^*} \leq \frac{\sum_{i=1}^{n} C_i^o - \frac{1}{m}\sum_{i=1}^{n} C_i^o}{\frac{1}{m}\sum_{i=1}^{n} C_i^o} = m-1.$

Recall that C_i^o is the shortest path through the network for job i, given that all the machines are free. We use the following example to show the tightness of the bound. In the example the number of machines is equal to the number of jobs and the graph is a directed linear graph.

$m = n$, $p_{ii} = 1$ for $1 \leq i \leq n-1$ and $p_{ij} = \varepsilon$ for $1 \leq i \leq n$ and $1 \leq j \leq n-1$ and $i \neq j$.

$p_{in} = i\varepsilon$ for $1 \leq i \leq n-1$ and $p_{nn} = 1 + n\varepsilon$. Therefore:

$C_i^o = 1+(n-2)\varepsilon + i\varepsilon$ for $1 \leq i \leq n-1$ and $C_n^o = 1+(2n-1)\varepsilon$. Consequently the MSP heuristic orders the jobs as (1, 2, . . ., n). However, the optimal order is the inverse of the order given by the heuristic and is (n, n-1, . . ., 1). Therefore, we have:

$C_i^{MSP} = i+(n+i-2)\varepsilon$ for $1 \leq i \leq n-1$ and $C_n^{MSP} = n+3(n-1)\varepsilon$. And

$C_n^* = 1+(n-1)\varepsilon + n\varepsilon$ and $C_1^* = 1+(n-1)\varepsilon + \frac{n(n+1)}{2}\varepsilon$. Therefore:

$$\frac{Z^{MSP}-Z^*}{Z^*} = \frac{n+3(n-1)\varepsilon - 1 - (n-1)\varepsilon - \frac{n(n+1)}{2}\varepsilon}{1+(n-1)\varepsilon + \frac{n(n+1)}{2}\varepsilon} \text{ and } \lim_{\varepsilon \to 0} \frac{Z^{MSP}-Z^*}{Z^*} = n-1 = m-1. \;\square$$

Corollary 3.7: The relative error bound of the MSP heuristic for minimizing makespan with a no-wait restriction is $\frac{Z^{MSP} - Z^*}{Z^*} \leq m - 1$. ***This bound is obtainable.***

Proof: The Myopic Shortest Path heuristic can be modified to obtain a solution under a no-wait restriction. Step 4 of the heuristic needs to be adjusted to incorporate this restriction. The proof is similar to the proof of Proposition 3.6. The example given in Proposition 3.6 and the schedules given in Figure 3.4 establish that the bound is tight. □

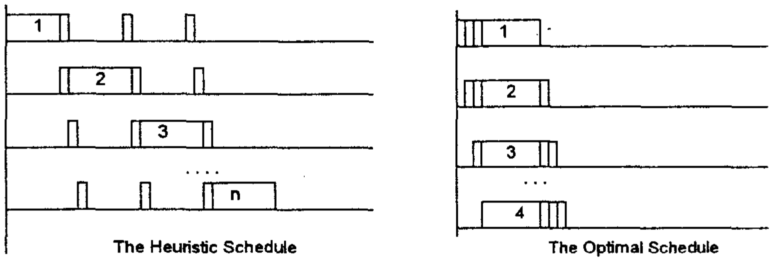


Figure 3.4

Next we look at special cases of the network flow shops, when the graph G is a *layered network*. Moreover, we assume that each layer (stage) l contains K_l identical machines and there are L layers.

Proposition 3.8: The relative error bound of the MSP heuristic is $\frac{Z^{MSP} - Z^*}{Z^*} \leq \sum_{l=1}^{L}(2 - \frac{1}{K_l}) - 1$. ***There exists a layered network for which this is the best bound.***

Proof: First we show that the MSP heuristic constructs a "busy" schedule for a layered network, then we establish the relative performance of the "busy" schedule. In the "busy" schedule, jobs are routed to the earliest available machine at each stage. Since the machines at each layer are identical, all the routes in the network are identical in their processing requirements. Consequently, Step 4 of the heuristic, which finds the shortest path given the machine availabilities, will route the jobs to the first available machine at each stage. Next, suppose all the n jobs are completely processed on the l-th stage before their processing begins on the (l+1)-th stage, for $1 \leq l \leq L-1$. Now let F_l denote the completion time of the jobs at stage l, if they were processed in isolation. We then have $Z^{MSP} \leq \sum_{l=1}^{L} F_l$. For each stage l, with K_l identical machines, from Graham [5], we get $F_l \leq (2 - \frac{1}{K_l})F_l^*$ for the bound on each stage of the parallel machines. Where F_l^* is the optimal processing for the jobs on stage l, if they were done in isolation. It is easy to see that $F_l^* \le Z^*$ for all l and consequently:

$$\frac{Z^{MSP}-Z^*}{Z^*} \le \frac{\sum_{l=1}^{L} F_l - Z^*}{Z^*} \le \frac{\sum_{l=1}^{L}(2-\frac{1}{K_l})Z^* - Z^*}{Z^*} = \sum_{l=1}^{L}(2-\frac{1}{K_l})-1.$$

The following example shows that the bound is the best possible for an instance of a network flow shop. $L = 1$, $K_1 = K$, $p_i = 1$ for $i = 1,\ldots,2K(K-1)$ and $p_i = 2K$ for $i = 2K(K-1)+1$. The makespan provided by the MSP heuristic is given by the schedule given in Figure 3.5.

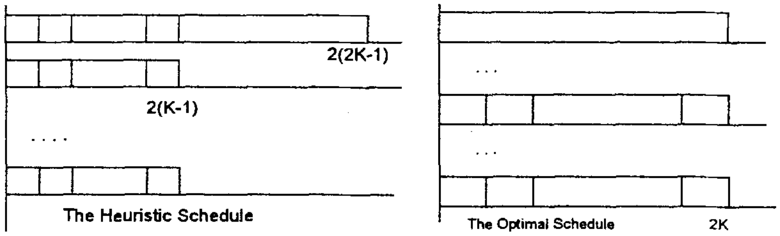


Figure 3.5

The corresponding optimal makespan is 2K and we have: $\frac{Z^{MSP}-Z^*}{Z^*} = \frac{2(2K-1)-2K}{2K} = 1-\frac{1}{K}$. □

In the next proposition, we derive the relative error bound for the *bottleneck network flow shop*. In a bottleneck network flow shop, we are assuming that there exists at least one machine, on which every job has to be processed. Moreover, the total processing time of all the jobs at the bottleneck machine is larger than the total processing time of jobs on any other machine.

Proposition 3.9: The relative error bound of the MSP heuristic is $\frac{Z^{MSP}-Z^*}{Z^*} \le K-1$, ***where K is the number of machines on the path with the smallest number of machines. The bound is obtainable for bottleneck network flow shop.***

Proof: Let r be the index for the bottleneck machine. We have $Z^* \ge \sum_{i=1}^{n} p_{ir}$. Now suppose all the jobs are routed through the path (L), the path with the minimum number of processors. Then:

$Z^{MSP} \le \sum_{i=1}^{n} C_i^o \le \sum_{j\in L} p_{ij} \le K\sum_{i=1}^{n} p_{ir}$. Therefore: $\frac{Z^{MSP}-Z^*}{Z^*} \le \frac{K\sum_{i=1}^{n} p_{ir} - \sum_{i=1}^{n} p_{ir}}{\sum_{i=1}^{n} p_{ir}} = K-1.$

We use the instance in Proposition 3.6 to show the tightness of the bound. Note in that example machine n is the bottleneck machine. □

Corollary 3.10: ***The relative error bound of the MSP heuristic for the proportionate network flow shop is*** $\frac{Z^{MSP} - Z^*}{Z^*} \leq K-1$, ***where K is the length of the path with minimum number of machines. The bound is obtainable.***

Proof: Note that in the exactly proportionate network flow shop, there exists at least one simple path, which is the common shortest path for all the jobs. The remainder of the proof follows from Proposition 3.9. □

Next, we provide a tight error bound when there are *K classes* of jobs. Within each class the jobs are assumed to be completely identical.

Proposition 3.11: ***If the jobs belong to K distinct classes, the relative error bound of the MSP heuristic is*** $\frac{Z^{MSP} - Z^*}{Z^*} \leq K-1$ ***and there exists a network flow shop for which this is a tight bound.***

Proof: The proof consists of two parts. In the first part we show the optimality of the MSP procedure when K=1. In the second part we establish the error bound for K>1. When K=1 the release sequence does not influence the scheduling objective, however routing decision does impact the performance criteria. The proof can be completed by performing an induction on the number of jobs being scheduled. It is easy to see that the first job is optimally scheduled. Now, we show that if jobs 1 through k have been optimally scheduled, then the procedure optimally routes the (k+1)-th job. Since the jobs have similar processing times on each machine, choices made in scheduling jobs 1 through k may not be reconsidered for scheduling of the (k+1)-th job. Furthermore, Step 4 of the heuristic finds the optimal finishing time for job (k+1). Thus the procedure constructs an optimal schedule. For instances where $K \geq 2$, we have $Z^* \geq Z^*(k)$. Where $Z^*(k)$ is the optimum solution for any given class of jobs. Now, if the classes of jobs are processed sequentially such that one class finishes completely before the next one starts, we have: $Z^{MSP} \leq \sum_{k=1}^{K} Z^*(k) \leq KZ^*$.

Therefore $\frac{Z^{MSP} - Z^*}{Z^*} \leq K-1$. To show the tightness of the bound, set m=n=K in the example given in Proposition 3.6. □

Next, we evaluate the performance of the MSP heuristic for the network flow shops with controllable processing times. To solve the problem, we propose the following two step procedure. The procedure is similar to Nowicki and Smutnicki [11].

Myopic Shortest Path with Controllable Processing Times

Step 1. Solve the network flow shop problem with the following processing times.

$p_{ij} \Leftarrow p_{ij} - z_{ij}$, where $z_{ij} = g(c_{ij})u_{ij}$, $0 \le x_{ij} \le u_{ij}$ and u_{ij} is the max imum compression amount .

$g(y) = \max\{\min\{\frac{1+\alpha(m-1)}{\alpha m} - \frac{y}{\alpha}, 0\}, 0\}$, $y \in [0,1]$ and $\alpha = 1 - \frac{\rho m}{[\rho + \sqrt{\rho(m-1)}]^2}$

and ρ is the worst case ratio of the network flow shop.

Step 2. Given the schedule in Step 1, determine the optimal makespan among all compression vectors.

Note that the optimization problem in Step 2 can be formulated as a standard linear program, which is solvable in polynomial time. In the next proposition, we provide the relative error bound for the above-mentioned heuristic.

***Proposition 3.12:** The relative error bound of the MSP heuristic with controllable processing times* $\frac{Z^{MSPC} - Z^*}{Z^*} \le \rho + \frac{\rho(m-\rho)}{2\rho + 2\sqrt{\rho(m-1)} - 1} - 1$, *where* $\rho \in [0,m]$. *The bound is tight for* $\rho = m$.

Proof: The proof given by Nowicki and Smutnicki [11] is derived for permutation schedules only. Now, suppose we restrict ourselves to the permutation schedules. With this focus, the relative error bound of the Myopic Shortest Path heuristic does not change. The network flow shop, where the routes have been decided, is a regular flow shop problem with zero processing times allowed. The relative error bound obtained by the Myopic Shortest Path for this flow shop problem satisfies the requirement for $\rho \in [0,m]$ and consequently the results follow. The example in Proposition 3.6 shows the tightness of the bound for $\rho = m$. □

Section 4. The Cycle Time Problem

The premise of just-in-time production systems has attracted considerable attention to the cyclic schedules, where the smallest set which satisfies the demand ratio of product mix is produced cyclically until the cumulative demand is produced. Although the cyclic schedules may not contain the optimal schedule, its repetitive nature simplifies the management of shop floor logistics and scheduling. Essentially, three classes of cyclic scheduling problems have been defined: i) The *no-wait* problem, which minimizes the cycle time subject to a no-wait constraint between machines; ii) The *minimum-wait* problem, which minimizes the average work-in-process inventory subject to the constraint that jobs must be processed with maximum throughput or equivalently with minimum cycle time; iii) The *finite-buffer* problem, which minimizes cycle time subject to the constraint that jobs cannot leave for the next machine for processing if the downstream machine's buffer is full. This problem is also known as the blocking problem. McCormick et al. [9] have shown that the finite-buffer problem is a special case of the no-buffer problem, by considering each buffer as a machine with zero processing times. For further reading on cyclic scheduling the reader is referred to McCormick et al. [9], Roundy [15] and Wittrock [20]. In the following, we investigate the complexity of the network flow shop cyclic scheduling problem.

Proposition 4.1: The problem of minimizing cycle time for a cyclic network flow shop with a no-wait constraint is NP-hard in the strong sense.

Proof: It is easy to see that the decision version of the problem is in NP, since the solution can be guessed and checked by a nondeterministic algorithm in polynomial time. We transform Numerical Matching with Target Sums, which is known to be NP-complete in the strong sense (see Garey and Johnson), to an instance of a no-wait rooted tree flow shop problem.

Numerical matching with target sums is defined as follows. Given disjoint sets X and Y, each containing n elements, sizes a_i and $b_i \in Z^+$ for each element $a_i \in X$ and $b_i \in Y$ and a target vector $c_i \in W$ with positive integer entries can $X \cup Y$ be partitioned into n disjoint subset $A_1, A_2, \ldots, A_n$ each containing exactly one elements from each of X and Y, such that $a_i + b_i = c_i$ for $i=1, \ldots, n$? Clearly a "yes" answer requires that $\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} c_i$. We now define an instance of no-wait cyclic scheduling in a network flow shop with 3n jobs:

Class I:	$p_{i1} = a_i + H,$	$p_{i2} = 2H,$	$p_{i3} = nH$	$\text{for } i = 1, \ldots, n.$
Class II:	$p_{i1} = b_i + 2H,$	$p_{i2} = 4H,$	$p_{i3} = nH$	$\text{for } i = n+1, \ldots, 2n.$
Class III:	$p_{i1} = 3H,$	$p_{i2} = nH,$	$p_{i3} = 6H + c_i$	$\text{for } i = 2n+1, \ldots, 3n$

where $H = 2\sum_{i=1}^{n} c_i$.

If Numerical Matching with Target Sums has a solution, then the cyclic schedule of jobs given by Figure 4.1 results in a cycle time equal to $(12n+1)\sum_{i=1}^{n} c_i$.

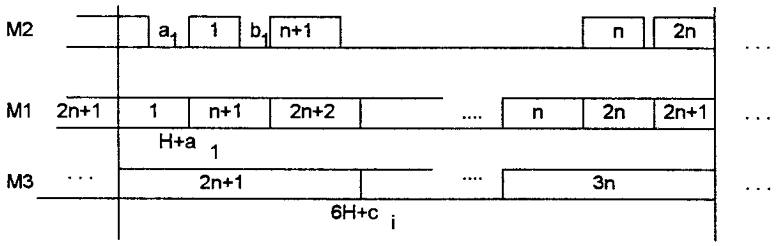


Figure 4.1

Although there are two routes for processing each job, the choice of the routing is quite clear. Also note that the cyclic schedule in Figure 4.1 there are no idle time on machines one and three.

Conversely if there is a cyclic schedule π with a cycle time less than or equal to $(12n+1)\sum_{i=1}^{n} c_i$, we show that it should be of the form given in Figure 4.1 and we make the following claims:

i) The schedule on machines one and three have no idle time,

ii) In the schedule π, job 2n+1 is immediately followed by job 1 and job n+1,

iii) In the schedule π, a job from class III should immediately follow the partial sequence of jobs (2n+1, 1, n+1).

Using claims ii and iii and an induction argument, it can be shown that the cyclic schedule π has the same form as the schedule given in Figure 4.1. If numerical matching with target sum does not exist, and $a_i + b_i < c_i$, idle time would be created on machine 1, resulting in a contradiction. Therefore $a_i + b_i \geq c_i$, and since in order to have a "yes" answer to the numerical matching with target sum problem we should have $\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} c_i$, we conclude that $a_i + b_i = c_i$ and therefore disjoint subsets should exist. Now we consider the proof of claims i, ii, iii, and iv.

i) The total processing time on machine one and machine three in a cycle for schedule π is equal to the cycle time given in Figure 4.1. Any idle time on these machine would result in a contradiction.

ii) There are eight other combination of jobs that could follow job 2n+1. It can be easily shown that all these combinations would create an idle time on either machine one or machine three, resulting in a contradiction.

iii) If a job from class I or II is scheduled after the partial schedule, then an idle time would be created on machine one, resulting in a contradiction.

From Lemma 3.2, we conclude that cyclic scheduling with no-wait restriction in an assembly tree is NP-hard in the strong sense. This completes the proof. □

Corollary 4.2: ***The problem of minimizing cycle time with no-buffer and finite-buffer restriction in a cyclic network flow shop are NP-hard in the strong sense.***

Proof: The proof of Proposition 3.4 could be used to show the first part of the corollary. Also, as discussed earlier the no-buffer problem can be reduced to a finite-buffer problem. □

Proposition 4.3: ***The minimum-wait cyclic network flow shop problem is NP-hard in the strong sense.***

Proof: The problem of minimum-wait in a two-machine flow shop has been shown to be NP-hard in the strong sense, (see Matsuo [8]). By restricting a network flow shop we conclude the proposition. □

Next we establish the relative error bound of the MSP heuristic for minimizing cycle time with no-buffer and finite buffer.

Proposition 4.4: ***The relative error bound of the MSP heuristic for minimizing cycle time subject to no-wait is*** $\frac{Z^{MSP}-Z^{*}}{Z^{*}} \leq m-1$. ***There is a cyclic network flow shop for which this is the best possible bound.***

Proof: The proof is similar to the proof of Proposition 3.6. The following instance shows that the bound is tight for an m machine flow shop.

$n = m+1$, $p_{ii} = 1$ for $1 \leq i \leq m-1$ and $p_{ij} = \varepsilon$ for $1 \leq i \leq m$ and $1 \leq j \leq m-1$ and $i \neq j$.

$p_{im} = i\varepsilon$ for $1 \leq i \leq n-1$, $p_{mm} = 1+m\varepsilon$, $p_{m+1,j} = \varepsilon$ for $j = 1,\ldots,m$.

The heuristic generates the sequence (m+1,1,2,...,m), whereas the optimum sequence is (m+1,m,m-1,...,1). The heuristic cycle time (CT^{MSP}) and optimum cycle time (CT^{*}) are:

$CT^{MSP} = m+2(m-1)\varepsilon$ and $CT^{*} = 1+\varepsilon+\frac{m(m+1)}{2}\varepsilon$. Therefore $\lim_{\varepsilon \to 0} \frac{Z^{MSP}-Z^{*}}{Z^{*}} = m-1$. □

Corollary 4.5: ***The relative error bound of the MSP heuristic for no-wait proportionate and bottleneck cyclic network flow shops are*** $\frac{Z^{MSP}-Z^{*}}{Z^{*}} \leq K-1$. ***This bound is tight.***

Proof: The proof is similar to the proof of Proposition 3.9. The same set of inequalities can be written for the cycle time. The example given in Proposition 4.4 can be used to show the tightness of the bound. □

Corollary 4.6: ***If the jobs belong to K distinct classes, the relative error bound of the MSP heuristic*** $\frac{Z^{MSP}-Z^{*}}{Z^{*}} \leq K-1$ ***and there exists a no-wait cyclic network flow shop for which this is a tight bound.***

Proof: The proof is similar to the proof of Proposition 3.11. The example given in Proposition 4.4 establishes the tightness of the bound. □

The MSP heuristic results in an arbitrarily bad relative error bound for the minimum-wait cyclic network problem. To show the bound we use the following example.

$m = 2, n = 3, p_{11} = 4, p_{12} = 2, \quad p_{21} = 6, p_{22} = 3, \quad p_{31} = 3, p_{32} = 6$

The MSP heuristic and the optimum schedules are shown in Figure 4.2.

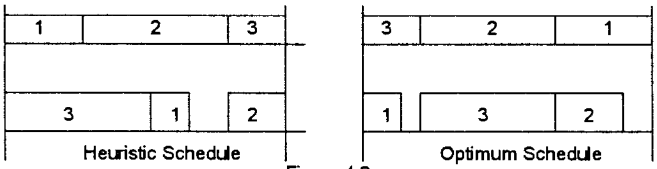


Figure 4.2

The cycle time is 13. The optimum schedule results in no waiting time, but the MSP heuristic has a total waiting time of 2. And the worst case relative error bound is arbitrarily bad. The same result could be obtained for the special cases of cyclic network flow shops, such as bottleneck and proportionate network flow shops.

Section 5. Average Flow Time Problem

In this section, we concentrate on the analysis of the MSP heuristic for the average and weighted average flow time problem.

Proposition 5.1: Minimizing mean flow time in a network flow shop is unary NP-hard.

Proof: The network flow shop subsumes a general instance of a simple flow shop problem. It is sufficient to note that minimizing mean flow time in two machine flow shops is unary NP-hard, (see Gonzalez and Sahni [3]). □

The next proposition provides a job dependent bound for the MSP heuristic.

Proposition 5.2: The relative error bound of the MSP heuristic is $\frac{Z^{MSP} - Z^*}{Z^*} \le \frac{n-1}{2}$ ***and there are network flow shops for which this is the best bound.***

Proof: We have $Z^* \ge \sum_{i=1}^{n} C_i^o$. From the heuristic we know: $C_1^o \le C_2^o \le \ldots \le C_k^o$. Consider the job order (1, 2, ..., n). Let C_k^{MSP} be the completion time of job k in the MSP heuristic. We have:

$C_k^{MSP} \le C_1^o + C_2^o + \ldots + C_k^o$. Therefore: $Z^{MSP} \le nC_1^o + (n-1)C_2^o + \ldots + 2C_{n-1}^o + C_n^o$.

$R = \dfrac{Z^{MSP} - Z^*}{Z^*} \le \dfrac{(n-1)C_1^o + (n-2)C_2^o + \ldots + C_{n-1}^o}{\sum_{i=1}^{n} C_i^o}$. R is maximized when :

$$C_1^o = C_2^o = \ldots = C_n^o \Rightarrow \frac{Z^{MSP} - Z^*}{Z^*} \le \frac{\frac{n(n-1)}{2} C_n^o}{nC_n^o} = \frac{n-1}{2}.$$

To show the tightness of the bound we use the example in Proposition 3.6. We have:

$Z^{MSP} = \dfrac{n(n+1)}{2} + \varepsilon f(\varepsilon)$; where $f(\varepsilon)$ is a polynomial function in ε. And $Z^* = n + \varepsilon g(\varepsilon)$.

$\lim_{\varepsilon \to 0} \dfrac{Z^{MSP} - Z^*}{Z^*} = \dfrac{n-1}{2}$. □

The next proposition provides a machine dependent bound for the MSP heuristic.

Proposition 5.3: *The relative error bound of the MSP heuristic is* $\dfrac{Z^{MSP} - Z^*}{Z^*} \le m-1$ *and there are network flow shops for which this is the best bound.*

Proof: For each job k we have $C_k^{MSP} \le C_1^o + \ldots + C_k^o$ if the shortest path for each job is used consecutively. Therefore $Z^{MSP} \le nC_1^o + (n-1)C_2^o + \ldots + C_n^o$ and $Z^* \ge \dfrac{n}{m} C_1^o + \dfrac{n-1}{m} C_2^o + \ldots + \dfrac{1}{m} C_n^o$.

Therefore: $\dfrac{Z^{MSP} - Z^*}{Z^*} \le \dfrac{C_1^o(n - \frac{n}{m}) + C_2^o(n - 1 - \frac{n-1}{m}) + \ldots + C_n^o(1 - \frac{1}{m})}{\frac{1}{m}(nC_1^o + (n-1)C_2^o + \ldots + C_n^o)} \le (m-1)$

to show tightness of the bound set n=2m-1 in the example used in Proposition 5.2. □

Proposition 5.4: *The MSP heuristic has a relative error bound of* $\dfrac{Z^{MSP} - Z^*}{Z^*} \le n-1$ *, when applied to the weighted flow time problem.*

Proof: Note that the order of the jobs is given as $C_1^o \le C_2^o \le \ldots \le C_n^o$. Therefore $Z^* \ge \sum_{i=1}^{n} w_i C_i^o$, and

$$Z^{MSP} \le w_1 C_1^o + w_2(C_1^o + C_2^o) + \ldots + w_n(\sum_{i=1}^{n} C_i^o)$$

$$R = \frac{Z^{MSP} - Z^*}{Z^*} = \frac{\sum_{i=1}^{n} w_i \sum_{j=1}^{i} C_i^o - \sum_{i=1}^{n} w_i C_i^o}{\sum_{i=1}^{n} w_i C_i^o} = \frac{C_1^o \sum_{i=2}^{n} w_i + \ldots + C_n^o w_n}{w_1 C_1^o + \ldots + w_n C_n^o}$$

note if $\frac{a}{b} \geq \frac{c}{d} \geq \frac{x}{y} \Rightarrow \frac{a}{b} \geq \frac{a+c+x}{b+d+y}$. Now let $\frac{C_k^o \sum_{i=k+1}^{n} w_i}{w_k C_k^o} = \max_{l=1}^{n} \frac{C_l^o \sum_{i=l+1}^{n} w_l}{w_l C_l^o}$,

by subtitutingevery termwith thelargest termwe get:

$$R \leq \frac{(n-1)C_k^o \sum_{i=k+1}^{n} w_i}{w_k C_k^o} = (n-1).$$

To establish the tightness of the bound, we modify the example used in Proposition 3.6, with the following weights. $w_i = 0$ for $i = 1,\ldots,n-1$ and $w_n = 1$. Therefore:

$$Z^{MSP} = \sum_{i=1}^{n} w_i C_i^{MSP} = w_n C_n^{MSP} = n + 3(n-1)\varepsilon \quad \text{And}$$

$$Z^* = \sum_{i=1}^{n} w_i C_i^* = w_n C_n^* = 1 + (2n-1)\varepsilon,$$

$$\frac{Z^{MSP} - Z^*}{Z^*} = \frac{n + 3(n-1)\varepsilon - 1 - (2n-1)\varepsilon}{1 + (2n-1)\varepsilon} \text{ and } \lim_{\varepsilon \to 0} \frac{Z^{MSP} - Z^*}{Z^*} = n - 1.$$

Proposition 5.5: The Weighted Myopic Shortest Path heuristic has a relative error bound of $\frac{Z^{MSP} - Z^*}{Z^*} \leq m - 1$ ***, when applied to the weighted flow time problem. There exists an instance of a network flow shop for which this is the best bound.***

Proof: We have $Z^* \geq \frac{1}{m} \sum_{i=1}^{n} w_i \sum_{j=1}^{i} C_i^o$, assuming that the jobs are processed based on the Weighted Myopic Shortest Path (WMSP) but are equally divided among all the machines. Also:

$$Z^{WMSP} \leq \sum_{i=1}^{n} w_i \sum_{j=1}^{i} C_i^o \text{ therefore } \frac{Z^{WMSP} - Z^*}{Z^*} = m - 1$$

We add the following weights to the example given in Proposition 3.6 to show the tightness of the bound. For $w_i = 1 + i\varepsilon$, for $i = 1,\ldots,n$. The heuristics generates the sequence (1,2,...,n), but the optimal solution is given by (n,n-1,...,1). The bound can be similarly established. □

Section 6. A Polylogarithmic Approximation Algorithm for the Makespan Problem

In this section we concentrate on developing a deterministic algorithm that generates a polylogarithmic approximation to the network flow shop problem, when the objective is to minimize makespan. The algorithm may be called a δ-approximation. If a polynomially time bounded algorithm always provides a makespan of at most δZ^*, then it is called a δ-approximation algorithm. The algorithm is shown to find a schedule with makespan $O(\log^2(m)C^*)$ in polynomial time.

Some of the most promising approximation algorithms for the general job shop scheduling problem of minimizing makespan are by: Sevastyanov [17], [18], Barany [1], and Fiala[2]. This stream of work may be classified as *geometric approach* to scheduling. Another line of research on the approximation algorithm, by Raghavan [13], Raghavan and Thompson [14], may be referred to as *randomized approach*. The work of Shmoys, Stein, and Wein [19] integrates the two approaches and provides the first polylogarithmic performance guarantee for a randomized and deterministic polynomial-time approximation algorithm for a job shop scheduling problem. In this section we extend this work to the network flow shop scheduling problem.

The idea behind the algorithm is as follows. We first relax the machine capacity constraint, in which each machine processes at most one job at any given time. Next, the jobs are scheduled through the network along their shortest paths. At this time more than one job may be assigned to any machine at any time. We show that we can delay the starting time of each job such that no more than a prespecified number of jobs are scheduled on any machine during any time unit. Next, a feasible schedule, which is at most within a constant from the optimal schedule, is constructed. The algorithm may be formally stated as:

The δ-approximation Algorithm:

Step 1. Define a *conflict* schedule, by performing each job starting at time zero and running continuously in the network, on the job's *shortest path.*

Step 2. Modify the *conflict* schedule by delaying the start time of the first operation of each job by a *random* amount from a discrete uniform distribution in the range $[0, \frac{\max_{i=1}^{n} C_i^o}{\log(m)}]$, such that no more than O(log(m)) jobs are scheduled on any machine at any given time.

Step 3. *Expand* this schedule such that at any given point in time all jobs being processed have the same size. *Flatten* this schedule to obtain a feasible schedule.

Note that the makespan of the *conflict* schedule is $\max_{i=1}^{n} C_i$. Step 2 of the algorithm requires delaying the start time of the jobs in the *conflict* schedule such that no more than a given number of jobs are concurrently processed on any machine at any point in time. We first show that with high probability no machine is assigned too many jobs at any time. Then, we show how to deterministically allocate delays to each job to generate a schedule in which the machines have O(log(m)) jobs running concurrently at any one point. This step, which is referred to *derandomization*, requires approximately solving an NP-complete integer programming problem. Our procedure differs from [19] in step 2. Recall that unlike job shops, in network flow shops the total workload of each machine and length of the jobs are functions of the process routing and is not fixed apriori. Therefore, we base the construct of the conflict schedule on the shortest path for the jobs, which is a lower bound to the optimal schedule.

Proposition 6.1: ***Given a network flow shop problem, the strategy of delaying each job with an initial integral amount selected randomly from a discrete uniform distribution from*** $[0, \frac{\max_{i=1}^{n} C_i^o}{\log(m)}]$, ***and then processing its operations in sequence, will produce an invalid schedule with makespan of at most*** $\max_{i=1}^{n} C_i + \frac{\max_{i=1}^{n} C_i^o}{\log(m)}$ ***and that with high probability has no more than O(log(m)) jobs scheduled on any machine within any unit time interval.***

Proof: The proof is shown for the case that maximum processing time (p_{max}) is bounded by a polynomial in m. In [19] it is proved that any instance of the job shop scheduling problem can be transformed into one with $p_{max} = O(m)$ with the property that a schedule of the modified instance with makespan of kZ^* can be converted in polynomial time to a schedule for the original instance of makespan $(k+1)\,Z^*$.

Now, we concentrate on a machine j and time t. Let p = probability[at least τ units of processing are allocated on machine j at time t] and $\Omega = \max_{i=1}^{n} C_i^o$. It is clear that there are at most $\binom{\Omega}{\tau}$ combinations for selecting τ units of processing time from all those jobs whose shortest path goes through machine j. The probability that any particular τ is scheduled at time t is $\frac{1}{\Omega}$. If all the τ units are from different jobs, then with a probability of at most $\left(\frac{1}{\Omega}\right)^{\tau}$ the different jobs are scheduled at time t. Otherwise, the probability that all τ are scheduled at that time is 0, since that is impossible. We therefore have:

$$p \le \binom{\Omega}{\tau}\left(\frac{1}{\Omega}\right)^{\tau} \le \left(\frac{e\Omega}{\tau}\right)^{\tau}\left(\frac{1}{\Omega}\right)^{\tau} \le \left(\frac{e}{\tau}\right)^{\tau}.$$

If $\tau = k(\log(m))$ then p would be sufficiently very small. To bound the probability that any machine at any time has more than $k(\log(m))$ jobs using it, we multiply p by $\max_{i=1}^{n} C_i + \frac{\max_{i=1}^{n} C_i^o}{\log(m)}$ for the number of time units in the schedule, and by m for the number of machines. The quantity $\max_{i=1}^{n} C_i$ is bounded by a polynomial in m, since we have assumed that p_{max} is similarly bounded. Therefore choosing a large enough k yields that, with high probability, no more than $k(\log(m))$ jobs are scheduled for any machine during any unit of time.

The Derandomization Procedure:

In this part, we develop an integer program to *derandomize* Step 2 of the δ-approximation algorithm. To achieve this, we formalize the problem as a vector selection problem and formulate it as a pure integer program, which is solved approximately.

Since we have assigned delays in the range $[0, \frac{\max_{i=1}^{n} C_i^o}{\log(m)}]$, the resulting schedule has length $L \le \max_{i=1}^{n} C_i + \frac{\max_{i=1}^{n} C_i^o}{\log(m)}$. Consequently, the processing of a job i with initial delay Γ can be represented by an (mL)-length $\{0,1\}$-vector where each position corresponds to a machine j and a given time t. It takes a value of 1 if machine j processes job i at time t, and 0 otherwise. Corresponding to each job i and each possible delay Γ, we have a vector $V_{i\Gamma}$ which indicates the assignment of delay Γ to job i. Now, let λ_i be the set of vectors $\{V_{i1}, \ldots, V_{i\Gamma_{max}}\}$, where $\Gamma_{max} = \frac{\max_{i=1}^{n} C_i^o}{\log(m)}$, and let $V_{i\Gamma}(k)$ be the k-th element of the $V_{i\Gamma}$. Let $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ be the set of vectors and $X_{i\Gamma}$ be the 0-1 variable if $V_{i\Gamma}$ is selected from λ_i. The problem can be stated as follows:

$$\begin{aligned}
&\text{Min} \quad W \\
\text{s.t.} \quad &\sum_{\Gamma=1}^{\Gamma_{max}} X_{i\Gamma} = 1 && i = 1,\ldots,n, \\
&\sum_{i=1}^{n} \sum_{\Gamma=1}^{\Gamma_{max}} V_{i\Gamma}(k) X_{i\Gamma} \le W && k = 1,\ldots mL \\
&X_{i\Gamma} \in (0,1)
\end{aligned}$$

where W is the maximum number of jobs that use a machine at any given time. The integer programming problem is shown to be NP-complete [13]. Therefore, we solve it by first solving its linear programming relaxation and then use the randomized rounding procedure given in [13] to obtain a solution which is within a constant of the optimum solution. From Proposition 6.1, we know that optimum value of W_{opt} =O(log(m)). The next proposition bounds the value of the heuristic solution.

Proposition 6.2: The randomized algorithm results in a solution which is $O(W_{opt} + \log(m))$ in polynomial time.

Proof: The result follows directly from the results in [13].

Step 3 of the approximate algorithm *expands* the schedule obtained in Step 2 such that at any given point in time all jobs being processed have the same size, and then *flattens* the schedule to get a feasible solution. The next proposition is due to [19].

Proposition 6.3: ***Given a schedule S^* of length L that has at most c jobs scheduled on one machine during any unit of time, there exists a polynomial-time algorithm that produces a valid schedule of length*** $O(cL\log(\max_{i=1}^{n} p_{ij}))$.

By applying the Proposition 6.3 we get the desired feasible schedule. The schedule can be shown to be of the length $O(\log^2(m)C^*)$, which has a polylogarithmic performance guarantee for network flow shop. Note that for the bottleneck and proportionate network flow shops, the bound is $O(\log^2(K)C^*)$, since we can construct the schedule based on a sample with K machines.

Section 7. Computational Experience

In this section, we report the result of our computational experiments that compare the two heuristics developed in the earlier sections. We test our proposed heuristics on a collection of random problems where the objective function is makespan, sum of completion time and cycle time. The problems tested have the following characteristics. The number of jobs in the experiments are set to be 20, 30, 40, and 50. The network configurations tested are: assembly tree (AT), rooted tree (RT), with three machines, layered network (LN) with three machines at each layer and with three layers, and the random network (RN) represented in Figure 1.1. The processing times of jobs for each machine was generated randomly from a discrete uniform distribution ranging from 10 to 50. Table 7.1 summarizes our computational results.

Table 7.1
Result of Computational Comparisons

No	Network Type	Jobs	Makespan	Sum of Completion	Cycle Time
1	AT	20	1.032	1.013	1.012
2		30	1.024	1.025	1.046
3		40	1.021	0.993	1.087
4		50	0.985	0.997	1.094
5	RT	20	1.097	1.032	1.033
6		30	1.056	0.997	1.068
7		40	1.023	0.995	1.013
8		50	1.024	1.014	0.978
9	LN	20	1.077	1.017	1.067
10		30	1.067	1.012	1.043
11		40	1.044	1.024	1.015
12		50	1.062	0.998	1.006
13	RN	20	0.983	0.996	1.026
14		30	1.084	1.038	1.033
15		40	1.036	1.017	1.016
16		50	1.014	1.013	1.014
Ave			**1.039**	**1.011**	**1.034**

Each entry in Table 7.1 reports the average ratio of solutions from using the MSP to the polylogarithmic heuristic. The results are obtained from solving 20 problems in each instance. Notice that on the average the Myopic Shortest Path heuristic performs much better than the Polylogarithmic heuristic, when the objective function is makespan or cycle time. For the network flow shop problems with sum of completion time as the objective function the two procedure perform quite similarly.

Section 8. Final Remarks

In this paper we have primarily concentrated on analysis of the makespan, cycle time, and flow time. The following example indicates that the MSP heuristic performs arbitrarily bad, when the objective function is to minimize total tardiness or the weighted number of tardy jobs.
$m = n = 3$. $p_{11} = M, p_{12} = 1, p_{13} = 1$. $p_{21} = 1+\varepsilon, p_{22} = M$, $p_{23} = 1$. $p_{31} = 1$, $p_{32} = 1, p_{33} = M+2\varepsilon$ with the due dates $d_1 = M+4+2\varepsilon, d_2 = M+3+2\varepsilon, d_3 = M+2+2\varepsilon$. The optimum sequence is $(3,2,1)$ with $\sum_{i=1}^{3} T_i = 0$, whereas the MSP heuristic generates the sequence$(1,2,3)$ with $\sum_{i=1}^{3} T_i = 3M+3$. Similarly, the same instance results in an arbitrarily bad error bound, for weighted number of tardy jobs. These observations indicate the need for more sophisticated versions of the MSP heuristic for due date related performance measures. The results presented in this paper is categorized in Table 8.1.

The network flow shop problem is a generalization of the classical scheduling problem considered in the literature. Commonly, the following categories have been used to classify the scheduling problems; single machine, single stage, flow shop, job shop, open shop and finally dag shop. (In a dag shop, only a partial order for visiting the processors is given, however the jobs complete their processing when a complete order to visit all the machines is defined.) The two constructs of the network flow shop, the graph G and the type of sample path, can be used to form an instance of any scheduling problem. For example, if G is a directed linear graph, the network flow shop reduces to a simple flow shop problem. When the graph G is a complete graph and any node is a potential source and terminal node in the graph, and the paths required for processing the parts are Hamiltonian paths rather than simple paths, then the problem is similar to an open shop problem, which every path is a viable order for completing the jobs. Similarly, the network flow shop equivalent to a dag shop is a clique but with partial order defined for processing the parts in the graph.

In the scheduling problem addressed in this paper, we have ignored the time required to move the jobs through the system. However, in many FMS's the time to transport a part from one machine

Table 8.1

Summary Results

Performance	**Restriction**	**Condition**	**Complexity**	**Error Bound**
Makespan	-	-	Unary NP-hard	m-1
Makespan	no-wait	-	Unary NP-hard	m-1
Makespan	-	Layered Network	Unary NP-hard	$\sum_{l=1}^{L}(2-\frac{1}{k_l})-1$
Makespan	-	Bottleneck Graph	Unary NP-hard	K-1
Makespan	-	Proportionate	NP-hard	K-1
Makespan	-	K Job-Classes	Unary NP-hard	K-1
Cycle Time	no-wait	-	Unary NP-hard	m-1
Cycle Time	no-wait	Proportionate	NP-hard	K-1
Cycle Time	no-wait	K Job-Classes	Unary NP-hard	K-1
Min-wait	Max throughput		Unary NP-hard	Arbitrarily bad
Flow Time	-	-	Unary NP-hard	(n-1)/2
Flow Time	-	-	Unary NP-hard	m-1
Weighted Flow Time	-	-	Unary NP-hard	n-1
Weight Flow Time	WMSP	-	Unary NP-hard	m-1
Weighted Number of Tardy Jobs	-	-	Unary NP-hard	Arbitrarily Bad
Total Tardiness	-	-	Unary NP-hard	Arbitrarily Bad

to another might be significant compared to the processing time of the jobs, and therefore should be considered explicitly in the scheduling decision making. Many of the results presented in this paper would hold for network flow shops when the graph is both edge and node weighted, where the weights on the arcs are the transport time.

Acknowledgment: The author would like to thank professor Marc Posner and the anonymous referee for their valuable suggestions.

References

[1] I. Barany, "A vector-sum theorem and its applications to improving flow shop guarantees", Math. of Opns. Res., Vol. 6, No.3, 1981.

[2] T. Fiala, "An algorithm for the open-shop problem", Math. of Opns. Res., Vol. 8, No. 1, 1983.

[3] T. Gonzalez and S. Sahni, "Flowshop and jobshop schedules: Complexity and approximation", Opns. Res. 26(11), 36-52 (1978).

[4] S.C. Graves, H.C. Meal, D. Stefek, and A.H. Zeghmi, "Scheduling of re-entrant flow shops", J. of Opns. Management, 3(4), (1983).

[5] R.L. Graham, "Bounds for certain multiprocessing anomalies", Bell System Tech. J. 45, 1563-1581, (1966).

[6] H. Kamoun and C. Sriskandarajah, " The complexity of jobs in repetitive manufacturing systems", Eur. J. of Opns. Res., 70, 350-364(1993).

[7] E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, "Recent developments in deterministic sequencing and scheduling: A survey", in: M.A.H. Dempster et al. (eds.), Deterministic and Stochastic scheduling, D. Reidel Publishing Co. Boston, MA, 35-73, (1982).

[8] H. Matsuo, "Cyclic sequencing problems in the two-machine permutation flowshop: Complexity, worst-case and average-case analysis", Naval Res. Log. 37, 679-694(1990).

[9] S.T. McCormick, M.L. Pinedo, S. Shenker, and B. Wolf, "Sequencing in an assembly line with blocking to minimize cycle time", Opns. Res. 37(6), 925-935(1989).

[10] Nowicki, E., and Zdrzalka, S., " A survey of results for sequencing problems with controlable processing times", Disc. Applied Math., 26, 271-287, 1990.

[11] Nowicki, E., and Smutnicki, C., " Worst-case analysis of an approximation algorithm for flow-shop sequencing problem", Opns. Res., 8, 171-177, 1989.

[12] P.S. Ow, "Focused scheduling in proportionate flowshops", Management Sci. 31(7), 852-869, (1985).

[13] P. Raghvan, " Probabilistic construction of deterministic algorithms: approximating packing integer programs", J. of Comp. and Sys. Sci., 37, 130-143, 1988.

[14] P. Raghvan and Thompson, C.D., "Provably good routing in graphs: regular arrays", Proc. of 17th Annual ACM Symp. on Theory of Computing, 79-87, 1985.

[15] R. Roundy, "Cyclic schedules for job shops with identical jobs", Math. of Opns. Res. 17(4), 842-865(1992).

[16] C. Sriskandarajah and S.P. Sethi, "Scheduling algorithms for flexible flowshops: worst case and average case performance", Eur. J. of Opns. Res. 43, 143-160 (1989).

[17] Sevastyanov, S.V., " Bounding algorithm for the routing problems with arbitrary paths and alternative servers", Kibernetika, 22 (6), 74-79, 1986.

[18] Sevastyanove, S.V., " Efficient construction of schedules close to optimal for the cases of arbitrary and alternative routes of parts", Soviet Math. Kokl., 29(3), 447-450, 1984.

[19] D. B. Shmoys, C. Stein, and J. Wein, "Improved approximation algorithm for shop scheduling problems", SIAM Jour. on Comp., Vol. 23, pp 617-632, 1994.

[20] R.J. Wittrock, "Scheduling algorithms for flexible flow line", IBM J. Res. Develop. Vol. 29, No. 401-412, July (1985).