

MULTISTAGE HYBRID FLOWSHOP SCHEDULING WITH IDENTICAL JOBS AND UNIFORM PARALLEL MACHINES

SUSHIL VERMA AND MAGED DESSOUKY*

Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089, U.S.A.

SUMMARY

The single-stage scheduling problem to minimize the makespan of identical jobs with general release times on uniform parallel machines is known to be solvable in polynomial time using the latest start time (LST) rule to determine the optimal schedule. In this paper, we first show that the two-stage problem is NP-hard using a known result that the three-stage problem with equal job release times is NP-hard. The rest of the paper consists of two parts. In the first paper, we compare the extension of the LST rule to the general multistage problem with other heuristics. We compare the heuristics based on the theoretically determined worst-case absolute error bound and on the experimentally determined average deviation from a developed lower bound. The analysis shows that in the presence of many stages, the LST rule does not perform as well as the other heuristics even though they are suboptimal for the single-stage problem. In the second part of this paper, we present a $(2 + \epsilon)$ -approximation algorithm for the multi-stage problem. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: ■■■

1. INTRODUCTION

It is common in capital intensive industries such as semiconductor manufacturing to find newer, more modern machines running side by side with older, less efficient machines which are kept in operation because of high replacement cost. The older machines may perform the same operations as the newer machines but with longer processing times. Lee and Vairaktarakis [1] define a hybrid flowshop design as a multi-stage flowshop with parallel machines at each stage. We will refer to a hybrid flowshop with uniform parallel machines at each stage as a hybrid uniform flowshop. We consider the scheduling of a batch of identical jobs in a hybrid uniform flowshop to minimize the completion time of the last job in the batch, i.e. makespan.

The majority of the literature for hybrid flowshops assumes non-identical jobs and identical parallel machines (see for example the work by Guinet and Solomon [2], Gupta [3], Gupta and Tunc [4], and Lee and Vairaktarakis [1]). Our focus will be on identical jobs and uniform parallel machines. The term *identical* jobs means that all jobs have the same processing time on a given machine and the term *uniform* machines means that the processing time of a job on a particular

*Correspondence to: Maged Dessouky, Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089-0193, U.S.A. E-mail: Maged@RCF.USC.edu

machine is the ratio of the processing time of the job on a machine with a standard speed to the speed of the particular machine [5].

Our problem can be formally stated as follows. A set of n independent jobs, J_1, J_2, \dots, J_n , released at times r_1, r_2, \dots, r_n respectively, have to be scheduled on a set of parallel machines M_{ik} ($i = 1, 2, \dots, m_k$) at each operation (stage) k for $k = 1, 2, \dots, q$. The processing time of a job on machine M_{ik} is p_{ik} for $i = 1, 2, \dots, m_k$ and $k = 1, 2, \dots, q$. Note that the processing time is only a function of the machine. For any feasible schedule, denote C_{jk} as the completion time of job J_j at stage k , $j = 1, \dots, n$ and $k = 1, \dots, q$. The makespan $C_{\max} = \max_{j=1, \dots, n} C_{jq}$. We use the classification scheme presented by Graham *et al.* [6] and Lenstra *et al.* [7] to formally state the problem as $F_q(Q_1, \dots, Q_q) | r_j, p_j = 1 | C_{\max}$. We denote it in short by F_q . As further notation, we use $p_k^{\max}(p_k^{\min})$ as the processing time of the slowest (fastest) machine at stage k , $r^{\max}(r^{\min})$ as the latest (earliest) release time, and $m = \sum_{k=1}^q m_k$. We will assume that the jobs are indexed in increasing order of the release time.

Dessouky *et al.* [8] propose an $O(n \log n)$ procedure based on the latest start time (LST) rule to solve the single-stage problem $F_1(Q_1) | r_j, p_j = 1 | C_{\max}$. This paper extends their research in various ways. We first show that in presence of general release times, the two-stage problem is NP-hard. For two stages and equal release times at the first stage ($F_2(Q_1, Q_2) | p_j = 1 | C_{\max}$), Dessouky *et al.* [9] present a $O(n \log n + n \log m)$ algorithm. For more than two stages, they show that the problem is NP-hard with equal release times. We use this result to show that in presence of unequal release times, the two-stage problem becomes NP-Hard.

The rest of the paper deals with the algorithms for the general multistage problem. In the first part of this paper, we evaluate the performance of four simple heuristics on the general q -stage problem. Each heuristic solves a series of q single-stage problems. All heuristics use the same rule for the last stage because an optimal schedule always exists with the last stage scheduled by LST as previously shown by Dessouky *et al.* [9]. The heuristics are evaluated based on the theoretical worst-case absolute deviation from optimality and on the average performance on randomly generated problem sets.

The first heuristic that is evaluated is the natural extension of the single stage algorithm. The LST rule is used in each stage and the completion time of a job at a particular stage is the release time of the job for the successor stage. We call this heuristic as Latest Start Time Heuristic (LSTH). Although using the LST rule guarantees optimality for the single-stage problem, it becomes suboptimal in the presence of more stages.

The other heuristics evaluated make use of scheduling rules in each stage that may be suboptimal for the single-stage problem, but in the presence of many stages have the potential to outperform the LST rule. These heuristics are: Earliest Completion Time Heuristic (ECTH), Fastest Available Machine Heuristic (FAMH) and the Mixed Heuristic (MH). Under ECTH, a single-stage rule known as Earliest Completion Time (ECT) is used in all but the last stage of the problem. In this rule, the machine which finishes the first available job the earliest is assigned to that job. By design the rule prefers the fast machines and uses them even if it has to induce extra waiting time. The rule is equivalent to the LST rule if the release times of all the jobs are equal. FAMH is similarly based upon a single-stage rule known as Fastest Available Machine (FAM), used in all but the last stage. In the FAM rule, the unscheduled job with the earliest release time is assigned to the first machine which becomes available. In case of a choice, the faster machine is chosen. This rule keeps the average waiting time lower at the cost of higher average processing times. Each of the two rules (ECTH and FAMH) outperforms the other for extremal values of the problem parameters. A mixture of these heuristics (MH), which uses a rule based on a mixed form

of the ECT and FAM rules at each stage, possesses the best aspects of the two heuristics and is comparable to the best of the two over the entire range of parameters in an average sense.

In the next section, we present a formal description of these heuristics which is followed by their worst-case analysis. We show that in the worst-case, ECTH, MH, and LSTH provide a makespan which is within $\sum_{k=1}^{q-1} p_k^{\max}$ of the optimal makespan. The deviation from the optimal is more ($2 \sum_{k=1}^{q-1} p_k^{\max}$) for FAMH. We show that on randomly generated problems the performance of the different heuristics is quite varied, in contrast to their rather similar worst-case guarantees. We show that for small values of n and q , ECTH outperforms both LSTH and FAMH and for large values of n and q , FAMH outperforms both ECTH and LSTH. The Mixed Heuristic performs comparably with the better of its components, FAMH and ECTH. We conclude the heuristic analysis section by comparing solutions generated by the mixed heuristic with lower bounds to the problem. Although each of the heuristics gives a makespan with load-independent absolute deviation from the optimal makespan (which guarantees strictly improving relative performance with increasing number of jobs), their relative performance for any number of jobs is not clear.

In the second part of this paper we present a $(2 + \varepsilon)$ -approximation algorithm. It does not use a stage-wise approach. Instead, a global network flow problem is used to get the desired approximation. The reader should note that for hybrid flow shop problems with non-identical jobs no constant approximation algorithm is known.

2. HEURISTICS

We start the section by describing the method known as Latest Start Time (LST). As we have stated earlier this procedure solves the single-stage problem $Q|r_j, p_j = 1|C_{\max}$ optimally [8]. The main idea is that there exists an optimal solution in which the job to machine assignments are independent of the release times. The assignments are first determined without taking release times into account and then a schedule is created accounting for the release times.

Procedure LST

Step 1: Let S_j be the latest time the j th job can start while allowing the completion of all jobs by time zero. We will identify S_j for all j . The first job will be started at time $-p^{\min}$ because this is the latest it can start and still finish at time zero. The second job is next scheduled to start at such a time so that a machine is available to perform the task and it is the latest the job can start and still finish at or before time zero. Continuing this process, we determine the job to machine assignments as well as the latest start times for all the jobs. This procedure returns the latest start times (S_1, \dots, S_n) as an ordered set.

Step 2: Suppose, without loss of generality that $r_1 \leq r_2 \leq \dots \leq r_n$. Then, job J_j is matched with S_{n-j+1} and the corresponding machine assignment. Given the machine assignments and the job release times, r_j , the jobs are scheduled without any unnecessary delays to determine their completion times, $C_j, j = 1, \dots, n$, and $C_{\max} = \max_{j=1, \dots, n} C_j$.

Procedure LST runs in time $O(n \log n)$ to solve Problem F_1 . Unfortunately, we cannot solve the two-stage problem F_2 by using Procedure LST for both stages. We show that Problem F_2 is indeed NP-hard. The following theorem, quoted from previous work, will form the basis for our assertion.

Theorem 2.1. Problem $F_3(Q_1, Q_2, Q_3)|r_j = 0, p_j = 1|C_{\max}$ is NP-hard [9]. The problem remains NP-hard even if $n = m_1$.

To apply this theorem to $F_2(Q_1, Q_2)|r_j, p_i = 1|C_{\max}$, we first note the following property of the LST rule which was proven in [8].

Minimality Property. Suppose $C_1 \leq C_2 \leq \dots \leq C_n$ be the completion times of n jobs for a schedule. The schedule is said to possess the minimality property if there does not exist another schedule, $C'_1 \leq C'_2 \leq \dots \leq C'_n$, in which $C'_j < C_j$ for any $j = 1, \dots, n$. The LST rule when applied to $Q|r_j = 0, p_j = 1|C_{\max}$ provides a schedule in which the minimality property of completion time holds.

Let us consider the class of those $F_3(Q_1, Q_2, Q_3)|r_j = 0, p_j = 1|C_{\max}$ problems where $n = m_1$. The minimality property implies that we can solve this problem by solving the pair of problems $F_1(Q_1)|r_j = 0, p_j = 1|C_{\max}$ for the first stage and $F_2(Q_2, Q_3)|r_j = C_{j1}, p_j = 1|C_{\max}$ for the second and third stage combined, in that order. For each of these two problems, the input length is a polynomial in the input length of the original three-stage problem. Since the second problem is just a special case of Problem $F_2(Q_1, Q_2)|r_j, p_j = 1|C_{\max}$, we have the following result.

Corollary 2.2. Problem $F_2(Q_1, Q_2)|r_j, p_j = 1|C_{\max}$ is NP-hard.

As expected, the minimality property ceases to hold in the presence of unequal release times under the LST rule. The reason being that it might be beneficial to put a job which is released earlier on a show machine as this can leave the faster machines available for jobs which will be released significantly later. This behaviour causes initial jobs of finish later than they could. In fact, one can show that the for certain problems no single schedule can achieve minimum completion times for all the jobs.

A natural extension to the approach taken in [9] for Problem F_q is to use the LST rule for each stage independently, proceeding from the first stage to the last stage. This is what we call the Latest Start Time Heuristic (LSTH). Though it guarantees optimality for that stage, it becomes suboptimal in the presence of more stages. We will frequently refer to single-stage problems without explicitly prescribing the machines used. We use the implicit context that if the single-stage problem is to solve the k th stage in the main problem, then we also use only the machines available at the k th stage.

Latest Start Time Heuristic (LSTH): LSTH is given by a sequence of q steps ($LSTH(k)$), one for each stage k . Recall that for and feasible schedule, C_{jk} denotes the completion time of job J_j at stages $k, j = 1, \dots, n$ and $k = 1, \dots, q$:

$$LSTH(k) = \begin{cases} \text{LST on } Q|r_j, p_j = 1|C_{\max} & \text{if } k = 1 \\ \text{LST on } Q|r_j = C_{j, (k-1)}, p_j = 1|C_{\max} & \text{if } k > 1 \end{cases} \quad (1)$$

The reader should note that independent of q , it is optimal to use the LST rule for the last stage of the problem. The observation follows from the fact that the LST rule solves Problem $Q|r_j, p_j = 1|C_{\max}$ optimally. In the remainder of the paper, we would indeed always use the LST rule for the last stage.

We show that this greedy approach is myopic in presence of many stages. It is shown experimentally that some simple heuristics outperform the LST rule for larger values of q . We first introduce the underlying single-stage rules and present some theoretical bounds on their performance.

Consider again the problem $Q|r_j, p_j = 1|C_{\max}$. Even though the LST rule solves this problem optimally, we define two new heuristic for it, the Earlier Completion Time (ECT) rule and the Fastest Available Machine (FAM) rule. Both are suboptimal for this problem. However, they have the potential to outperform LSTH when applied to a large number of stages.

Procedure ECT. Order jobs J_1, J_2, \dots, J_n in non-decreasing order of ready times. At step j , schedule the j th job in the ordered list on the machine which will complete the job at the earliest time.

The reader should observe that the ECT rule is optimal if $r_j = 0$.

Procedure FAM. Order jobs J_1, J_2, \dots, J_n in non-decreasing order to ready times. At step j , schedule the j th job in the ordered list on an idle machine or the first available machine, if an idle machine is not available. Break ties by selecting the fastest machines.

Based upon these procedures we introduce heuristics for the multistage problem.

Earliest Completion Time Heuristic (ECTH). ECTH is given by a sequence of q steps ($ECTH(k)$), one for each stage k .

$$ECTH(k) = \begin{cases} \text{ECT on } Q|r_j, p_j = 1|C_{\max} & \text{if } k = 1 \\ \text{ECT on } Q|r_j = C_{j, (k-1)}, p_j = 1|C_{\max} & \text{if } 1 < k < q \\ \text{LST on } Q|r_j = C_{j, (q-1)}, p_j = 1|C_{\max} & \text{if } k = q \end{cases} \quad (2)$$

Fastest Available Machine Heuristic (FAMH). FAMH is given by a sequence of q steps ($FAMH(k)$), one for each stage k .

$$FAMH(k) = \begin{cases} \text{FAM on } Q|r_j, p_j = 1|C_{\max} & \text{if } k = 1 \\ \text{FAM on } Q|r_j = C_{j, (k-1)}, p_j = 1|C_{\max} & \text{if } 1 < k < q \\ \text{LST on } Q|r_j = C_{j, (q-1)}, p_j = 1|C_{\max} & \text{if } k = q \end{cases} \quad (3)$$

The schedule produced by ECTH is characterized by a steady output of jobs at each stage. While the LST rule produces the minimum values for the maximum completion time, it ignores the completion time of jobs which are completed before the last job. As a result, the completion time of early jobs at this stage tend to be higher under the LST rule as compared to the ECT rule. Therefore, the next stage cannot start till later pushing the whole schedule forward. We will see the experimental evidence of this intuition in the forthcoming section.

The rationale underlying Procedure FAM is the conventional wisdom prevailing in a flowshop: do not make a machine wait and if there is more than one machine waiting, choose the fastest one. If the shop runs continuously for a long period of time with a large number of jobs and consists of machines with similar speeds, then it is expected that the higher level of machine utilization would also drive the makespan lower. We will see later that though FAMH is worse than ECTH for a small number of jobs and stages, it outperforms ECTH for a large number of jobs and stages.

This (n, q) -dependent behaviour of ECTH and FAMH motivated us to coin a mixed heuristic which selectively used both the rules. It uses the following single-stage procedure to solve $Q|r_j, p_j = 1|C_{\max}$.

Procedure ECT-FAM. Order jobs J_1, J_2, \dots, J_n in non-decreasing order of ready times. Find the optimal makespan of Problem $Q|r_j, p_j = 1|C_{\max}$ using the LST rule. Call it C_{\max}^0 . At step j ,

schedule the j th job according to the FAM rule if the resulting completion time is no more than C_{\max}^o . Otherwise, use the ECT rule.

The corresponding multistage heuristic is called the Mixed Heuristic..

Mixed Heuristic. MH is given by a sequence of q steps (MH(k)), one for each stage k :

$$MH(k) = \begin{cases} \text{ECT-FAM on } Q|r_j, p_j = 1 | C_{\max} & \text{if } k = 1 \\ \text{ECT-FAM on } Q|r_j = C_{j, (k-1)}, p_j = 1 | C_{\max} & \text{if } 1 < k < q \\ \text{LST on } Q|r_j = C_{j, (q-1)}, p_j = 1 | C_{\max} & \text{if } k = q \end{cases} \quad (4)$$

The motivation for the new single-stage rule is to use the good facet of each rule. The new rule tends to use the FAM rule more often for initial jobs and tends to only use the ECT rule for the later jobs. The goal is to not make a machine wait unless using it necessarily increases the makespan. Note that if $r_j = 0$, Procedure ECT-FAM, like Procedure ECT produces the optimal schedule, unlike pure FAM. We show experimentally that the Mixed Heuristic nearly uniformly beats the pure FAMH and pure ECTH over the entire range of parameters in an average sense.

We note that all the presented heuristics for the q -stage problem have the same computational complexity of $O(qn \log n)$ since each single-stage heuristic is $O(n \log n)$. Since the input consists of n release times, and $m (\geq q)$ machine processing times, all heuristics run in polynomial time.

3. WORST-CASR BOUNDS

Let t_{jk} be the completion times at atage k (arranged in an increasing order) under a heuristic and let \bar{t}_{jk} be the minimum makespan of the problem $Q|r_l = C_{l, k-1}, p_l = 1 | C_{\max}$ for $k > 1$ and $Q|r_l = r_j, p_l = 1 | C_{\max}$ for $k = 1$ with a total of j jobs. The value of \bar{t}_{jk} represents the minimum possible value of the j th completion time at stage k , given the completion times of the jobs at stage $k - 1$. Since these minima may not be achievable for all the jobs at the same time, sequence $(\bar{t}_{1k}, \dots, \bar{t}_{nk})$ may not represent a feasible solution.

Note that $\bar{t}_{jk} \leq t_{jk}$ for all j . Let δ_k be the maximum difference over $k = 1, \dots, q$.

$$\delta_k = \max_{j \leq n} (t_{jk} - \bar{t}_{jk}) \quad \text{for } k = 1, \dots, q \quad (5)$$

The value of δ_k plays an important role in evaluating the performance of a heuristic. The following Proposition 3.1, Theorem 3.2 and Corollary 3.3 have all appeared in [9] for the case of $q = 3$ with equal release times. In this paper, they appear in a generalized form for the multistage problem with unequal release times.

Proposition 3.1. Suppose that C is the makespan of the schedule provided by a heuristic applied on problem F_q and C_{\max} is the optimal makespan. Also suppose that δ_k is calculated using the completion times of the jobs under the heuristic. Then $C - C_{\max} \leq \sum_{k=1}^{q-1} \delta_k$.

The above proposition will be used to bound the performance of all the heuristics analysed in this paper. For each heuristic, we will first bound δ_k . For LSTH, we quota a generalized version of the theorem in [9].

Theorem 3.2. Under LSTH, $\delta_k \leq p_k^{\max}$ for all $k = 1, \dots, q$.

The main idea behind the proof of the theorem is the following. In the latest starting times provided by the LST rule, all the machines finish processing the jobs at the same time. If we let their finish times differ by a quantity x , then the resulting makespan for that stage would be suboptimal by no more than x . The quantity \bar{t}_{jk} equals to the optimal makespan if only the first j jobs are to be scheduled. This case is where the machines finish processing the jobs at the same time. The quantity t_{jk} refers to the time the first j of n jobs finish processing. During the process of mapping the release times to the latest start times in the context of the first j jobs, the machines' finish times may differ from each other but by no more than p^{\max} . For a detailed proof, the reader is referred to the paper [8]. The following corollary summarizes the final error bound result for the Latest Start Time Heuristic.

Corollary 3.3. Suppose C is the makespan of the schedule produced by the Latest Start Time Heuristic applied on $F_q(Q_1, Q_2, \dots, Q_q) | r_j, p_j = 1 | C_{\max}$ and C_{\max} is the minimum makespan. Then $C - C_{\max} \leq \sum_{k=1}^{q-1} p_k^{\max}$.

In this paper, we prove a similar result for ECTH, FAMH and MH. The approach is to prove a result analogous to Theorem 3.2 for the different heuristics. We first prove a supporting result. The availability time of a machine refers to the time the machine first becomes available for processing jobs.

Lemma 3.4. Consider the single-stage problem $Q | a_i, r_j = r, p_j = 1 | C_{\max}$ where $a_i \geq 0$ is the availability time of machine i . Then,

- (i) Procedure ECT yields the minimality property for the completion time for all jobs on this problem,
- (ii) Procedure ECT-FAM yields the optimal makespan, and
- (iii) Procedure FAM provides a makespan no more than p^{\max} larger than the optimal makespan.

Proof: Consider the ordered union of sequences $\{a_i + p_i, a_i + 2p_i, \dots\}$ for all i . Each of these sequences represents the times a machine finishes if run continuously from the time it becomes available. Call this ordered union S . It is clear that the completion time of each job belongs to the sequence S under each of the three procedures. Under Procedure ECT, the machine which finishes the job the earliest is used first. Therefore, the completion time of job j is just equal to the j th number in the sequence S . The statement (i) follows. Under Procedure ECT-FAM, the completion times are not mapped in this clear manner to the members of S . However, we are *a priori* given the value of the optimal makespan. By the definition of Procedure ECT-FAM, we do not choose a machine which results into a completion time which is more than the optimal makespan. Therefore, we map the completion times of the n jobs to some of the initial numbers of the sequence S which are no greater than the value of the optimal makespan. Independent of the specific mapping meeting the above criterion, Procedure ECT-FAM will yield the optimal makespan. The statement (ii) follows. Under Procedure FAM, the completion times may be mapped to numbers greater than the optimal makespan. Nonetheless, if a machine M_i is picked for the d th time and is assigned to job j , then before this machine is picked again for assignment, all the members of sequence S which lie between $a_i + (d-1)p_i$ and $a_i + dp_i$ are mapped to some forthcoming jobs. The reason is that the machines associated with completion times lying in this range are available earlier than machine M_i and hence will be picked before M_i under Procedure FAM. It implies that we are not going to overshoot the optimal makespan by more than p^{\max} . The statement (iii) follows. \square

Proposition 3.5. Suppose C is the makespan of the schedule produced by any one of the Procedures ECT, FAM or ECT-FAM applied on the single-stage problem $Q | r_j, p_j = 1 | C_{\max}$ and C_{\max} is the minimum makespan. Then,

- (i) under Procedure ECT and Procedure ECT-FAM, $C - C_{\max} \leq p^{\max}$, and
- (ii) under Procedure FAM, $C - C_{\max} \leq 2p^{\max}$.

Here, p^{\max} is the processing time of the slowest machine.

Proof: Suppose the jobs are indexed in order of their release times and s_j is the start time for job J_j . Also suppose that job \bar{j} is the last job for which $r_{\bar{j}} = s_{\bar{j}}$. It implies that all the jobs from $\bar{j} + 1$ to n wait before they start getting processed. In other words, the release times $r_{\bar{j}+1}, \dots, r_n$ do not affect the heuristic at all. Suppose that after finishing the first $\bar{j} - 1$ jobs machine i becomes available at time a_i for all i . Note that $a_i - r_{\bar{j}} \leq p^{\max}$ for all i . Consider the $n - \bar{j} + 1$ job problem $Q | a_i, r_j = r_{\bar{j}}, p_j = 1 | C_{\max}$. Suppose \bar{C} is the optimal makespan of this problem. Then, $C_{\max} \geq \bar{C} - \max_i(a_i - r_{\bar{j}}) \geq \bar{C} - p^{\max}$.

- (i) Under ECT and ECT-FAM, $\bar{C} = C$ according to Lemma 3.4 (statements (i) and (ii)). Therefore, $C_{\max} \geq C - p^{\max}$. The result follows.
- (ii) Under FAM, $\bar{C} + p^{\max} \geq C$ according to Lemma 3.4 (statement (iii)). Therefore, $C_{\max} \geq C - 2p^{\max}$. The result follows. □

Theorem 3.6. (i) Under ECTH, $\delta_k \leq p_k^{\max}$ for $k = 1, \dots, q - 1$.

(ii) Under FAMH, $\delta_k \leq 2p_k^{\max}$ for $k = 1, \dots, q - 1$.

(iii) Under MH, $\delta_k \leq p_k^{\max}$ for $k = 1, \dots, q - 1$.

Proof: Recall that each of the heuristics (ECTH, FAMH and MH) apply the LST rule to the last stage. ECTH uses Procedure ECT in the initial stages, FAMH uses Procedure FAM and MH uses Procedure ECT-FAM. Interpreting the ordered completion times of stage $k - 1$ as the release times for stage k , we can portray the problem at stage k as a single-stage problem, $Q | r_j, p_j = 1 | C_{\max}$. Recall that δ_k equals $\max_{j \leq n} (t_{jk} - \bar{t}_{jk})$ for $k = 1, \dots, q$. In the context of the single-stage problem, we can interpret t_{jk} as the makespan for the first j jobs under a specific heuristic applied to this problem. Similarly, \bar{t}_{jk} is equal to the optimal makespan for the first j jobs for the above problem. Therefore, the difference $t_{jk} - \bar{t}_{jk}$ is just an absolute bound on the performance of a heuristic on the single stage problem with j jobs. Proposition 3.5 provides these bounds for each of the heuristics. Therefore we can conclude independently of j that under Procedure ECT and Procedure ECT-FAM, $t_{jk} - \bar{t}_{jk} \leq p_k^{\max}$, and under Procedure FAM, $t_{jk} - \bar{t}_{jk} \leq 2p_k^{\max}$. The results follow. □

The absolute error bound result for the three heuristics is summarized below.

Corollary 3.7. Suppose C is the makespan of the schedule produced by ECTH, FAMH or MH applied on $F_q(Q_1, Q_2, \dots, Q_q) | p_j = 1 | C_{\max}$ and C_{\max} is the minimum makespan. Then,

- (i) under ECTH and MH, $C - C_{\max} \leq \sum_{k=1}^{q-1} p_k^{\max}$, and
- (ii) under FAMH, $C - C_{\max} \leq 2 \sum_{k=1}^{q-1} p_k^{\max}$.

Proof: The results follow from Proposition 3.1 and Theorem 3.6. □

In relative terms (i.e. the ratio of the heuristic solution over the optimal solution), it is clear that all the heuristics perform within a factor of $\max_k p_k^{\max} / p_k^{\min}$ to the optimal makespan. It is easy to

see that this is nearly a tight bound for FAMH.[†] It can also be shown easily that ECTH performs no worse than a factor of $q - 1$ of the optimal. On average, FAMH and ECTH perform much better, as we will see in the next section.

4. EXPERIMENTAL RESULTS

The worst-case bounds proved in the last section do not serve well to distinguish between the various heuristics we have presented in this paper. That is, all the heuristics seem to perform evenly in the worst case. In this section we instead resort to an average experimental analysis and show that these heuristics perform significantly differently from each other in the average sense. We also estimate their performance in absolute sense by comparing them individually to a lower bound on the optimal makespan.

The values tested for the number of jobs were 100 and 1000 and for the number of machines at each stage were 2, 10, 50, 100 and 1000. The processing time of each machine at each stage is sampled from a uniform integer random number from 1 to 10. The set of experiments assume equal release time at the first stage since the heuristics will give similar schedules with a large deviation between the release times at the first stage and with a small deviation between the release times the unequal release time problem is similar to the equal release time problem in the presence of many stages. We first set the number of machines at all the stages equal to each other because that reduces the chance of creating an obvious bottleneck in the system. The problem remains NP-Hard even with this assumption [9]. We later test the performance of the heuristics in the presence of a bottleneck stage.

For each combination of number of machines at each stage and jobs, 30 random samples were generated. The results are presented in a series of comparative plots. Figure 1 shows the comparison between the LSTH and ECTH (i.e. the ratio of LSTH solution over the ECTH solution). It shows clearly that for most values of the parameters, ECTH becomes steadily more dominating as the number of stages, or number of machines per stage increase. This is in contrast to the identical absolute error bounds on these two heuristics. Recall, Heuristic LSTH simply attempts to minimize the completion time of the last job at a given stage neglecting the completion times of the other jobs. As a consequence, compared to ECTH, LSTH tends to have a higher completion time for the earlier jobs. Hence, for large q this increase in the completion times for the earlier jobs in the intermediate stages increases the final makespan.

Figure 2 shows the comparison between ECTH and FAMH. It is apparent that as the system becomes less congested (i.e. small ratio of n/m) ECTH dominates FAMH when the number of stages is not large because in this case it is beneficial to wait for faster machines that will become idle soon. However, with increasing congestion FAMH eventually takes over ECTH and becomes steadily more dominant as q increases. FAMH tends to always keep all the machines busy if there is a large number of jobs to process whereas ECTH may keep a machine needlessly

[†]Consider the q -stage problem with two jobs and two machines at each stage with equal and negligible processing time ($= \varepsilon$) except at the second to last stage where $p_{q-1}^{\max} \gg p_{q-1}^{\min}$. The optimal makespan for this problem is equal to $(q - 1)\varepsilon + 2p_{q-1}^{\min}$. However, the FAMH makespan is equal to $(q - 1)\varepsilon + p_{q-1}^{\max}$. The reasons being that at the second to last stage both the jobs and both the machines become available for processing at the same time and hence both would be used under FAMH. The makespan ratio is equal to $p_{q-1}^{\max}/2p_{q-1}^{\min}$ as ε goes to zero

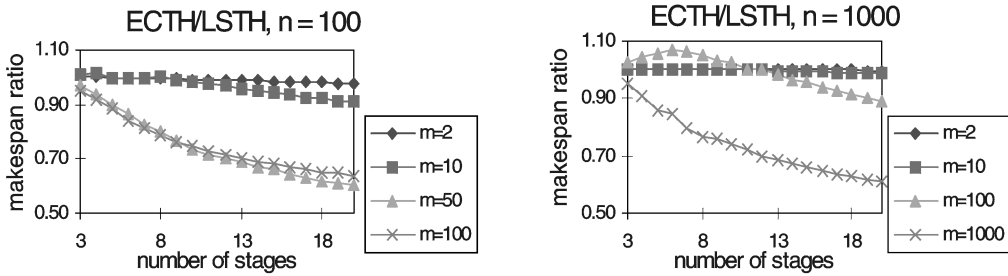


Figure 1. Comparison of the Latest Start Time Heuristic and the Earliest Completion Time Heuristic

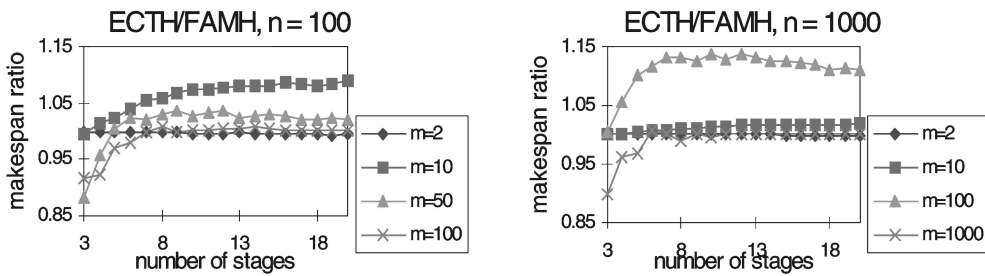


Figure 2. Comparison of the Earliest Completion Time Heuristic and the Fastest Available Machine Heuristic

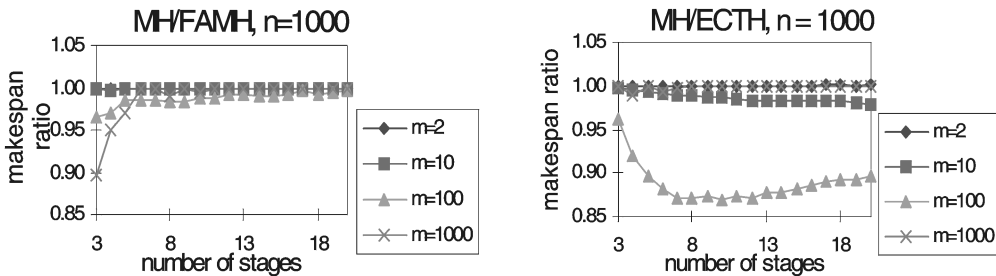


Figure 3. Comparison of pure ECTH and pure FAMH with the Mixed Heuristic

idle to wait for a faster machine. Figure 3 shows how the Mixed Heuristic nearly dominates both ECTH and FAMH individually over the entire range of parameters. It provides evidence for the hypothesis that MH consists of the strong points of both ECTH and FAMH as was argued earlier in the paper.

The next set of experiments tested the sensitivity of the results with changes in some of the problem parameters. In the first set of experiments we varied the range on the processing times at each stage. Let p be the maximum processing time at any machine and the processing time at a machine was randomly sampled from a discrete uniform random number from $[1, p]$. The previous set of experiments had $p = 10$ and we made additional runs with $p = 5, 50, 100$. The results showed that the relative performance of the mixed heuristic, MH, to ECTH is relatively

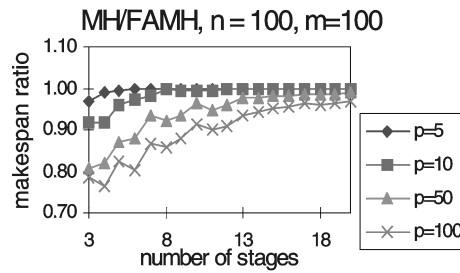


Figure 4. Comparison of the Mixed Heuristic with FAMH over a range of processing times

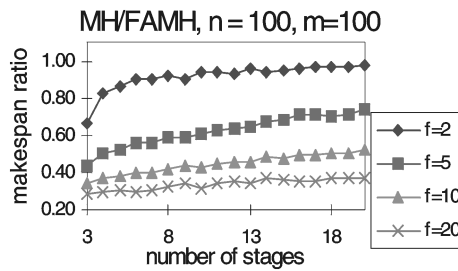


Figure 5. Comparison of the Mixed Heuristic with FAMH over a range of bottleneck speed factors

insensitive on the processing time range, p . However, the performance of FAMH worsens as p increases. Figure 4 shows the ratio of the MH solution over the FAMH solution as a function of p for the case $n = 100$ and $m = 100$. The reason the performance of FAMH deteriorates as the processing time range increases because it will schedule a job on a machine with a large processing time when the other machines are busy. Heuristics ECTH and MH will not schedule a job on the machine with the large processing time if a busy machine with a much smaller processing time will become idle soon.

The second set of experiments tested the sensitivity of the results in the presence of a clear bottleneck stage. In this set of experiments we randomly selected a bottleneck stage for each run and the processing time for a machine at the bottleneck stage was sampled from a discrete uniform random number from $[1, 10f]$ where $f = 2, 5, 10, 20$. For the non-bottleneck stages the processing times were sampled from a discrete uniform random number from $[1, 10]$. Note that the larger the f value the more significant the bottleneck becomes. Similar to the experiments with varying p , the results showed the relative performance of MH to ECTH is relatively insensitive to changes in f . Figure 5 shows the ratio of the MH solution over the FAMH solution as a function of f for the case $n = 100$ and $m = 100$. The performance of FAMH worsens as f increases because the problem essentially reduces to a single-stage and this is the case in which it performs poorly.

4.1. Lower bound

An effective lower bound on the value of the optimal makespan can be found by the following procedure.

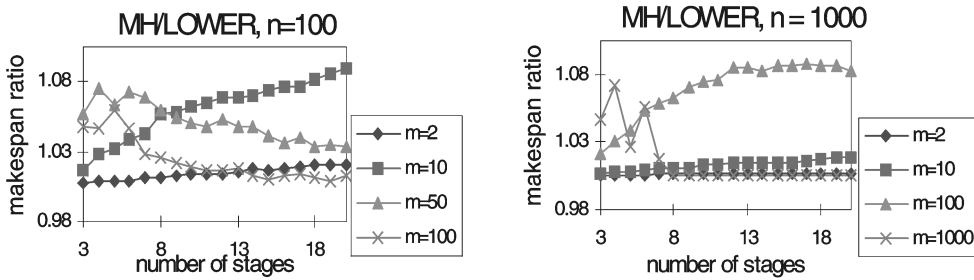


Figure 6. Comparison of the Mixed Heuristic with the lower bound

Procedure Lower Bound. At stage 1, use the given release times to calculate the vector $(\bar{t}_{11}, \dots, \bar{t}_{n1})$. Set $\hat{t}_{i1} = \bar{t}_{n1}$. For a general stage k , use the vector $(\hat{t}_{1,k-1}, \dots, \hat{t}_{n,k-1})$ as the ordered release time vector to calculate the vector $(\hat{t}_{1,k}, \dots, \hat{t}_{n,k})$ where \hat{t}_{jk} is the minimum makespan for the problem $Q|r_l = \hat{t}_{l,k-1}, p_l = 1|C_{\max}$ with a total of j jobs. This problem can be solved using the LST rule. Do this sequentially for all the stages up to stage q . The value of \hat{t}_{nq} gives a lower bound on the value of the optimal makespan.

Note that given the release times at stage $k - 1$, the vector $(\hat{t}_{1k}, \dots, \hat{t}_{nk})$ gives us the minimum possible completion time for each job j . Such a vector of completion times does not represent a feasible schedule in general. However, it does provide a lower bound on all the completion times and hence on all the release times of the next stage. This process, when completed for all the stages, gives a bound on the overall makespan.

In Figure 6, we show the comparison between the Mixed Heuristic (which was found to be the best overall in the average sense) and the lower bound given by the above procedure. The Mixed Heuristic also performs well in an absolute sense. Although as expected the performance of MH grows worse as the number of stages increases, the graphs show that the ratio levels off and in most cases the ratios are close to one. In terms of the performance as a function of the number of jobs and the number of machines at each stage, the graph shows that the ratio is the worst when the number of jobs is lower than the number of machines at each stage. Figure 6 plotted the average ratio of the mixed heuristic over the lower bound. We remark that the worst-case ratio in any single run was less than 1.25.

5. APPROXIMATION ALGORITHM

In an average sense on randomly selected problems, the four heuristics presented in this paper have been shown to provide a makespan which is within a factor of two of the lower bound. In addition, the absolute bounds on the error produced by any of the four heuristics has been proved to be independent of the total number of jobs. It implies that all the heuristics are asymptotically optimal as n becomes large. However, for a small number of jobs, the relative performance of all the heuristics except for FAMH is an open question and is subject to further research. For FAMH, we earlier showed by example that the ratio of the FAMH solution over the optimal solution can be unbounded for small values of n . In this section, we provide a $(2 + \epsilon)$ -approximation

algorithm that takes a global approach by considering all the stages at once in determining the schedule. Note that the previous heuristics find a solution to the q -stage problem by solving q single-stage problem.

Before we proceed to describe the approximation, we would discuss an extreme case which we would like to eliminate right away. If two successive release times are separated by a very large duration (enough to process all the jobs released before the gap through all stages), then the problem essentially gets partitioned into two parts: one catering to jobs before this gap and the other catering to jobs released after the gap. Moreover, only the solution to the second problem has any bearing on the overall makespan. Therefore, we can safely assume in the remainder of this section that gaps are not this large. Specifically, we will assume that $r_{j+1} - r_j < n \sum_{m=1}^q p_k^{\min}$ for $j \leq n - 1$. Note that this implies, $r^{\max} - r^{\min} < n^2 \sum_{k=1}^q p_k^{\min}$.

To achieve the desired approximation, we constrain the completion times of a machine to be a multiple of its processing time. It is shown that the optimal solution of the constrained problem serves as a nice approximation for the original problem. This constrained problem can be formulated as a min-cost max-flow network flow problem which can be solved efficiently. For a similar application of a min-cost max-flow formulation to solve a network load balancing problem, the reader is referred to [10].

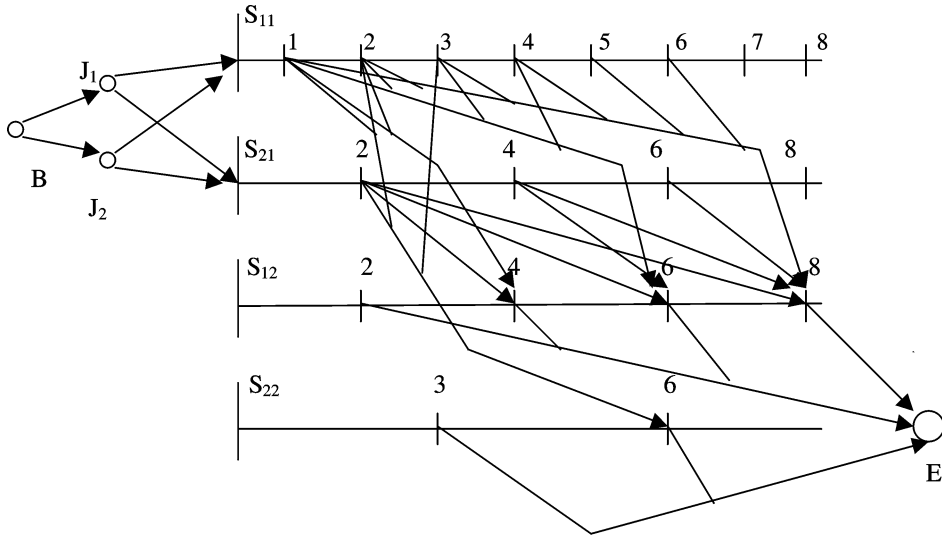
The constrained problem is denoted as $F_q^C(S_{11}, \dots, S_{ik}, \dots)$ (or just F_q^C if there is no danger of confusion), where S_{ik} denotes the set of allowable completion times for machine M_{ik} for $k = 1, 2, \dots, q$ and $i = 1, 2, \dots, m_k$. Let s_{ikl} denote the l th member of set S_{ik} .

We construct a directed network associated with the above problem. It consists of a start node B , and end node E , a node each for the n jobs (denoted by J_1, J_2, \dots, J_n) and a node for each member of each set S_{ik} . To avoid unnecessary notation, we would denote these nodes by s_{ikl} . Note that there are exactly $n + \sum_{i,k} |S_{ik}| + 2$ nodes. There is a directed arc (B, J_j) for each $j = 1, 2, \dots, n$. There is a directed arc (J_j, s_{i1l}) for $j = 1, 2, \dots, n$, $i = 1, 2, \dots, m_1$, and $l = 1, 2, \dots, |S_{i1}|$. There is a directed arc $(s_{ikl}, s_{i'(k+1)l'})$ if $s_{ikl} + p_{i'(k+1)l'} \leq s_{i'(k+1)l'}$ for $k = 1, 2, \dots, q - 1$, $i = 1, 2, \dots, m_k$, $l = 1, 2, \dots, |S_{ik}|$, $i' = 1, 2, \dots, m_{k+1}$, $l' = 1, 2, \dots, |S_{i'(k+1)l'}|$. Lastly, there is a directed arc (s_{iql}, E) for $i = 1, 2, \dots, m_q$, and $l = 1, 2, \dots, |S_{iq}|$. All arcs have a capacity of one. In addition, all nodes in S_{ik} have a capacity of one. Note that this can be easily modelled in a directed network by splitting each node into one incoming and one outgoing node with a new arc connecting the two with a capacity of one. The arcs (s_{iql}, E) have a cost of $m_q^{\text{rank}(s_{iql})}$ for $i = 1, 2, \dots, m_q$, and $l = 1, 2, \dots, |S_{iq}|$. The function $\text{rank}(s_{iql})$ gives the position of element s_{iql} in the ordered union of elements of type s_{iql} for $i = 1, 2, \dots, m_q$, and $l = 1, 2, \dots, |S_{iq}|$. The rest of the arcs have zero cost. To illustrate the network for a small problem, see Figure 7.

It is clear that the above network has a maximum flow of n units which is achievable if and only if Problem F_q^C has a feasible solution. The costs on the last set of arcs, (s_{iql}, E) , increase exponentially with the value of s_{iql} . Since s_{iql} have integral values, it ensures that a later production slot is used in the last stage only if all the previous production slots together cannot enable a feasible schedule. The solution of the network flow problem therefore corresponds to the minimum makespan of Problem F_q^C .

Lemma 5.1. The number of arithmetic steps required to solve Problem $F_q^C(S_{11}, \dots, S_{ik}, \dots)$ is a polynomial in n and $\sum_{i,k} |S_{ik}|$.

Proof. The min-cost max-flow problem is known to be solvable in time which is a polynomial in the size of the network [11]. □



Two jobs; zero release times; two stages;
 Two machines at each stage;
 $P_{11} = 1, P_{21} = 2, P_{12} = 2, P_{22} = 3$;
 There is an arc from J_1 and J_2 to each node in set S_{11} and S_{21} with flow of one each.
 All nodes can carry at most one unit of flow.
 The optimal solution has makespan of 4. The network solution has makespan of 6.

Figure 7. An example network

Suppose C is an upper bound on the makespan of the q -stage problem. For machine M_{ik} with processing time p_{ik} , we define for all $k = 1, 2, \dots, q$ and $i = 1, 2, \dots, m_k$,

$$S_{ik} = \{ \lceil r^{\min} / p_{ik} \rceil p_{ik}, (\lceil r^{\min} / p_{ik} \rceil + 1)p_{ik}, \dots, \lceil C / p_{ik} \rceil p_{ik} \} \quad (6)$$

Proposition 5.2. If Problem F_q has an optimal makespan C , then the optimal solution of Problem $F_q^C(S_{11}, \dots, S_{ik}, \dots)$, where S_{ik} is given by equation (6), is no more than $2C$.

Proof. Consider an optimal solution to Problem F_q . Suppose T_{jk} and C_{jk} represent the optimal starting and completion time for job j at stage k and machine m_{jk} represents the machine assigned to job j at stage k . We now construct a feasible solution to Problem F_q^C . We use the same machine to job assignment (m_{jk}) . However, the starting time for job j at stage k is given by the unique member of $S_{m_{jk}}$ (say s) such that both $s - p_{m_{jk}}$ and s belong to the interval $(2T_{jk}, 2C_{jk}]$. This is possible because the length of the interval $(2T_{jk}, 2C_{jk}]$ is exactly $2p_{m_{jk}}$ and hence contains some exactly two successive members of the set $S_{m_{jk}}$. It is clear that in the above schedule, the maximum completion time for any job is no more than $2C$. The result follows. \square

Note that the optimal solution to Problem F_q^C when applied to Problem F_q will have unnecessary forced idle time due to the preset completion times from set S_{ik} . An improvement to the schedule can be made by using only the machine to job assignments from the solution of Problem F_q^C . Then, schedule forward using these assignments without needlessly inserting idle time.

To evaluate the complexity of the min-cost flow problem, we require a bound on the cardinality of set S_{ik} which in turn requires an upper bound on the makespan, C , in terms of the problem parameters. We use the obvious bound that $C \leq r^{\max} + n \sum_{k'=1}^q p_{k'}^{\min}$. Using the assumption made earlier that $r^{\max} - r^{\min} < n^2 \sum_{k'=1}^q p_{k'}^{\min}$, the bound on C implies $C - r^{\min} \leq (n + n^2) \sum_{k'=1}^q p_{k'}^{\min}$. Therefore, $|S_{ik}| \leq (n + n^2)/p_{ik} \sum_{k'=1}^q p_{k'}^{\min} + 2 \leq (n + n^2)/p_k^{\min} \sum_{k'=1}^q p_{k'}^{\min} + 2$. The cardinality of the set $|S_{ik}|$ is therefore dependent upon the relative values of the processing times of the fastest machine at each stage. Unfortunately these ratios, in general, can be exponential in the size of the problem.

To achieve a truly polynomial approximation algorithm, we divide the stages into two categories, *fast* and *slow*. Suppose $p^{\min} = \max_k p_k^{\min}$. Then, for $\epsilon > 0$, stage k is termed ϵ -fast if $p^{\min}/p_k^{\min} \geq 2nq/\epsilon$, otherwise it is termed ϵ -slow. Suppose G represents the set of ϵ -fast stages and \bar{G} represents the set of ϵ -slow stages.

Consider the reduced version of Problem F_q where all the ϵ -fast stages have been removed. We call this problem $RF_q(\epsilon)$.

Proposition 5.3. *If a solution of Problem $RF_q(\epsilon)$ has a makespan C , then it can be used to construct a solution to Problem F_q with makespan no more than $C(1 + \epsilon/2)$.*

Proof. In the solution of Problem $RF_q(\epsilon)$, we simply insert the ϵ -fast stages assuming that we use only the fastest machine in these stages. The new makespan is no more than $C + n \sum_{k \in G} p_k^{\min} \leq C + n \sum_{k \in G} (\epsilon/(2nq)) p^{\min} \leq C + (\epsilon/2) p^{\min} \leq C(1 + \epsilon/2)$. \square

Corollary 5.4. *There exists a $(2 + \epsilon)$ -approximation algorithm for Problem F_q . Its running time is of the order of running time of a min-cost max-flow problem on a directed network with $O(mn^3 q^3/\epsilon)$ nodes where m is the total number of machines in the hybrid flowshop.*

Proof. Let Problem $RF_q^C(\epsilon)$ be Problem $F_q^C(S_{11}, \dots, S_{ik}, \dots)$ with the ϵ -fast stages removed and S_{ik} given by equation (6) with $C = n \sum_{k \in \bar{G}} p_k^{\min}$. We first solve Problem $RF_q^C(\epsilon)$. Under Lemma 5.1, the running time for this problem is polynomial in n and the cardinalities of the sets S_{ik} . Since only the ϵ -slow stages are part of Problem $RF_q^C(\epsilon)$, $|S_{ik}| \leq (n + n^2)/p_k^{\min} \sum_{k' \in \bar{G}} p_{k'}^{\min} + 2 \leq (n + n^2) \sum_{k' \in \bar{G}} (p_{k'}^{\min}/p_k^{\min}) + 2 \leq (n + n^2) \sum_{k' \in \bar{G}} (p^{\min}/p_k^{\min}) + 2 \leq (n + n^2)q(2nq/\epsilon) + 2 = 2(n^2 + n^3)q^2/\epsilon + 2$. The total number of nodes in the network equals $(2 + 2(n^2 + n^3)q^2/\epsilon) \sum_{k \in \bar{G}} m_k + n + 2$ which is $O(mn^3 q^2/\epsilon)$. Therefore, Problem $RF_q^C(\epsilon)$ is solvable in polynomial time. Now assume that its makespan is C' . Then, under Proposition 5.2, Problem $RF_q(\epsilon)$ has a makespan no more than $2C'$. We can find a solution with this makespan by using the same machine to job assignments as in the solution of Problem $RF_q^C(\epsilon)$. Under Proposition 5.3, this further implies that Problem F_q has a makespan no more than $2C'(1 + \epsilon/2) = (2 + \epsilon)C'$. This makespan is achievable by simply inserting the fast stages amongst the slow ones. \square

We end the section by detailing all the prices of the approximation algorithm together.

Approximation Algorithm

Input: Release times r_1, r_2, \dots, r_n ; Machine processing times p_{ik} for $k = 1, 2, \dots, q$ and $i = 1, 2, \dots, m_k$; a parameter ϵ .

Output: Job completion times C_{jk} for $j = 1, 2, \dots, n$ and $k = 1, 2, \dots, q$.

1. Determine the set G . Stage $k \in G$ if $p^{\min}/p_k^{\min} \geq 2nq/\epsilon$. Set $\bar{G} = N/G$.
2. For each $k \in \bar{G}$ and $i = 1, 2, \dots, m_k$, calculate the set $S_{ik} = \{ \lceil r^{\min}/p_{ik} \rceil p_{ik}, (\lceil r^{\min}/p_{ik} \rceil + 1) p_{ik}, \dots, \lceil C/p_{ik} \rceil p_{ik} \}$.

3. Solve Problem $F_q^C(S_{11}, \dots, S_{ik}, \dots)$ by solving the min-cost max-flow problem described above. The output of the flow problem is job to machine assignments and the completion times C_{jk} for $j = 1, 2, \dots, n$ and $k \in \bar{G}$.
4. Insert the stages $k \in G$ by just using the fastest machine at each stage and separating the solution calculated in the last step. To illustrate, suppose k' is the first stage in set G . Assuming without loss of generality that $C_{j,k'-1}$ are ordered, set $C_{1k'} = C_{1,k'} + p_{k'}^{\min}$ and $C_{jk'} = \max(C_{j,k'-1}, C_{j-1,k'} + p_{k'}^{\min}) + p_{k'}^{\min}$ for $j = 2, 3, \dots, n$. Now update the completion times for all $k \in \bar{G} > k'$ by using $C_{j,k'}$ as release times and the job to machine assignments calculated in [8]. Repeat this for each stage in set G in sequence.

6. CONCLUDING REMARKS

We have studied the performance of four different heuristics as applied to the multi-stage hybrid flowshop problem to minimize the makespan with identical jobs and uniform parallel machines at each stage. It was shown that Heuristic FAMH has a worst-case absolute bound that was twice as large Heuristics LSTH, ECTH and MH. In terms of the average performance, it was shown that the optimal strategy for the single stage problem (Heuristic LSTH) does not perform as well when the number of stages increases. Procedure ECTH performs best when there are a small number of jobs while FAMH performs best when there are a large number of jobs. The quality of the solution generated by ECTH decreases as the number of jobs increases because in this case there is a high likelihood that the procedure needlessly inserts idle time. In terms of FAMH, its performance decreases as the number of jobs decreases because in this case there is a benefit for waiting for a faster machine that becomes idle soon. For these reasons, we developed a mixed heuristic MH which uses the best features of both FAMH and ECTH. We have also presented an algorithm which provides a $(2 + \epsilon)$ -approximation. The solution approach to the approximation algorithm differs from the four other heuristics since it takes into consideration the scheduling of all the stages at once.

REFERENCES

1. C. Y. Lee and G. L. Vairaktarakis, 'Minimizing makespan in hybrid flowshops', *Oper. Res. Lett.*, **16**, 149–158 (1994).
2. A. G. P. Guinet and M. M. Solomon, 'Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion-time', *Int. J. Prod. Res.*, **34**, 1643–1654 (1996).
3. J. N. D. Gupta, 'Two stage hybrid flowshop scheduling problem', *J. Oper. Res. Soc.*, **39**, 359–364 (1988).
4. J. N. D. Gupta and E. A. Tunc, 'Scheduling a 2-stage hybrid flowshop with separable setup and removal times', *European J. Oper. Res.*, **77**, 415–428 (1994).
5. T. C. E. Cheng and C. C. S. Sin, 'A state-of-the-art review of parallel-machine scheduling research', *European J. Oper. Res.*, **47**, 271–290 (1990).
6. R. L. Graham, E. L. Lenstra, J. K. Lenstra and A. H. G. Rinnooy Kan, 'Optimization and approximation in determining sequencing and scheduling: a survey', *Ann. Discrete Math.*, **5**, 287–326 (1979).
7. E. L. Lenstra, J. K. Lenstra and A. H. G. Rinnooy Kan, 'Recent developments in deterministic sequencing and scheduling', in M. A. H. Dempster, J. K. Lenstra and A. H. G. Rinnooy Kan (eds.), *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht, 1982, pp. 35–73.
8. M. I. Dessouky, B. Lageweg, J. K. Lenstra and S. L. van de Velde, 'Scheduling identical jobs on uniform parallel machines', *Statistica Neerlandica*, **44**, 115–123 (1990).
9. M. M. Dessouky, M. I. Dessouky and S. Verma, 'Flowshop scheduling with identical jobs and uniform parallel machines', *European J. Oper. Res.*, **109**, 620–631 (1997).
10. X. Deng, H. Liu, J. Long and B. Xiao, 'Competitive analysis of network load balancing', *J. Parallel Distributed Computing*, **40**, 162–172 (1997).
11. G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, NY, 1988.