Approximation algorithms for the multiprocessor open shop scheduling problem

Citation for published version (APA):

Document status and date:
Published: 01/01/1997

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Approximation algorithms for the multiprocessor open shop scheduling problem *

Petra Schuurman † Gerhard J. Woeginger ‡

Abstract

We investigate the multiprocessor multi-stage open shop scheduling problem. In this variant of the open shop model, there are \( s \) stages, each consisting of a number of parallel identical machines. Each job consists of \( s \) operations, one for each stage, that can be executed in any order. The goal is to find a non-preemptive schedule that minimizes the makespan.

We derive two approximation results for this \( \mathcal{NP} \)-hard problem. First, we demonstrate the existence of a polynomial-time approximation algorithm with worst case ratio 2 for the case that the number \( s \) of stages is part of the input. This algorithm is based on Rasmäny's concept of dense schedules. Secondly, for the multiprocessor two-stage open shop problem we derive a family of polynomial-time approximation algorithms whose worst case ratios can be made arbitrarily close to \( \frac{2}{\sqrt{3}} \).

Keywords: open shop, scheduling, worst case analysis, approximation algorithm.

1 Introduction

Problem statement. An open shop is a multi-stage production process where the order in which the jobs pass through the stages is immaterial. There are \( n \) jobs \( J_j (j = 1, \ldots, n) \), and each job \( J_j \) consists of \( s \) operations \( O_{ij} \). Operation \( O_{ij} (i = 1, \ldots, s) \) has to spend a time \( p_{ij} \) at stage \( i \) of the production process; \( p_{ij} \) is called the processing time or the length of operation \( O_{ij} \). In the classical open shop problem, there is only a single machine available for each stage. In the multiprocessor open shop problem, at every stage \( i \) there is a number \( m_i \) of identical machines available that can operate in parallel. At any time, every job can be processed by at most one machine and every machine can process at most one job. Throughout this paper we assume that preemption is not allowed, i.e., once an operation is started, it must be processed without interruption till completion. The goal is to compute a schedule that minimizes the makespan, i.e., the maximum completion time of all jobs. The optimum makespan is denoted by \( C_{\text{max}}^* \). In the standard scheduling notation, makespan minimization in a classical \( s \)-stage open shop is denoted by \( \text{O}_{\text{S}}[C_{\text{max}}] \) and \( \text{O}[C_{\text{max}}] \) (depending on whether

---

*This research has been supported by the START program Y13-MAT of the Austrian Ministry of Science.

† petra@win.tue.nl, Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands.

‡ gwoegi@opt.math.tu-graz.ac.at, Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria.
or not the number \( s \) of stages is specified as part of the input), and makespan minimization in a multiprocessor \( s \)-stage open shop is denoted by \( O_s(P)[C_{\text{max}} \text{ and } O(P)]C_{\text{max}} \), respectively.

**Computational complexity and approximation algorithms.** Very few open shop problems can be solved in polynomial time; the main well-solvable non-preemptive case is \( O(2)[C_{\text{max}} \text{ and } O(P)]C_{\text{max}} \) (Gonzalez & Sahni [1976]). In the same paper Gonzalez and Sahni prove that, for any given \( s \geq 3 \), \( O_s[C_{\text{max}} \text{ is } \mathcal{NP}\text{-hard in the ordinary sense. The problem } O_s(P)[C_{\text{max}} \text{ is easily seen to be } \mathcal{NP}\text{-hard in the strong sense, and the multiprocessor two-stage open shop problem where there are two machines in one stage and only a single machine in the other stage is obviously } \mathcal{NP}\text{-hard in the ordinary sense. The case of the multiprocessor open shop where the number of stages is part of the input is strongly } \mathcal{NP}\text{-hard (see e.g. Williamson et al. [1997])}. As a consequence of the inherent difficulty of this scheduling problem, research has focused on obtaining polynomial-time approximation algorithms, i.e. fast algorithms that construct schedules whose makespan is not too far away from the optimum makespan.

We say that an approximation algorithm has *performance guarantee* or *worst case ratio* \( \rho \), for some real \( \rho > 1 \), if it always delivers a solution with makespan at most \( \rho C_{\text{max}} \). Such an approximation algorithm is then called a \( \rho \)-approximation algorithm. A family of polynomial-time \((1+\varepsilon)\)-approximation algorithms over all \( \varepsilon > 0 \) is called a *polynomial-time approximation scheme*, or PTAS for short. If the time complexity of a PTAS is also polynomial in \( 1/\varepsilon \), then it is called a *fully polynomial-time approximation scheme*, or FPTAS for short. It is known that the existence of a FPTAS for a strongly \( \mathcal{NP}\)-complete problem would imply \( \mathcal{P}=\mathcal{NP} \) (see Garey & Johnson [1979], page 141).

**Previous work on the open shop.** The known results on the approximability of the multiprocessor open shop problem are summarized in Table 1. The two-stage open shop can be solved in polynomial time if \( m_1 = m_2 = 1 \) (Gonzalez & Sahni [1976]). For the case where the number of stages is constant and where there is a single machine per stage, Sevastianov & Woeginger [1997] describe a PTAS. It is straightforward to carry the PTAS of Sevastianov & Woeginger [1997] over to the case where the number of stages and the numbers of machines per stage are all constant. This explains the ‘PTAS’-entries in Table 1. For the cases where the number of stages or the number of machines per stage is part of the input, the approximability of the multiprocessor open shop is not well understood. On the negative side, Williamson et al.

<table>
<thead>
<tr>
<th>Number of stages</th>
<th>Number of machines per stage</th>
<th>1</th>
<th>constant</th>
<th>part of input</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>poly-time</td>
<td>PTAS</td>
<td>wcr \leq \frac{3}{2} + \varepsilon *</td>
<td></td>
</tr>
<tr>
<td>constant \geq 3</td>
<td>PTAS</td>
<td>PTAS</td>
<td>wcr \leq 2 *</td>
<td></td>
</tr>
<tr>
<td>part of input</td>
<td># PTAS</td>
<td># PTAS</td>
<td># PTAS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>wcr \leq 2</td>
<td>wcr \leq 2 *</td>
<td>wcr \leq 2 *</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The approximability landscape of the multiprocessor open shop problem. The results marked by an asterisk are the contribution of this paper.
[1997] prove that the existence of a polynomial-time approximation algorithm with worst case ratio less than $\frac{5}{4}$ for an open shop problem with an arbitrary number of stages would imply $P=NP$. On the positive side, Racsamány gave a simple greedy algorithm for the classical open shop problem that yields a worst case ratio of 2.

We are aware of only two other papers on the multiprocessor open shop. Lawler, Luby & Vazirani [1982] investigate the preemptive version of the multiprocessor open shop and derive a polynomial-time algorithm for the most general version of this problem where the number of stages and the number of machines are both part of the input. Their results completely settle the complexity and approximability status of the preemptive multi-stage multiprocessor open shop problem. Chen & Strusevich [1993] deal with the non-preemptive variant. For the case where the number of stages and the number of machines are both part of the input they develop a polynomial-time algorithm with worst case ratio $2 + \varepsilon$, where $\varepsilon > 0$ can be made arbitrarily close to 0. For the case with only two stages, they derive an improved worst case ratio of $2 - \frac{2}{m}$, where $m = \max\{m_1, m_2\} \geq 2$.

**Contribution of this paper.** Racsamány (cf. Bárány & Fiala [1982]) introduced the concept of *dense* schedules for the classical open shop problem with a single machine per stage. It has been shown that the makespan of any dense schedule is at most twice the optimum makespan. In this paper, we show that dense schedules also yield a 2-approximation algorithm for the multiprocessor problem $O(P)||C_{\text{max}}$. Although the proof of this result is very simple, it outperforms the best previously known approximation algorithm for this problem (which is due to Chen & Strusevich [1993]), both with respect to the quality of approximation and with respect to the running time.

We also take a closer look at $O2(P)||C_{\text{max}}$, and we derive a $(\frac{3}{2} + \varepsilon)$-approximation algorithm for this problem, where $\varepsilon > 0$ can be made arbitrarily small. This approximation algorithm is mainly based on simple observations on the position of the 'long' operations in the optimum schedule. As a subproblem, the parallel machine problem $P|r_j|C_{\text{max}}$ comes up and is handled by the methods of Hall & Shmoys [1989].

**Organization of the paper.** The paper is organized as follows. Section 2 describes and analyzes the 2-approximation algorithm for the general problem $O(P)||C_{\text{max}}$. Section 3 contains the $(\frac{3}{2} + \varepsilon)$-approximation algorithm for $O2(P)||C_{\text{max}}$. Section 4 concludes the paper with a short discussion.

## 2 Dense schedules for the multiprocessor open shop

A feasible schedule for an open shop problem is called *dense* when any machine is idle if and only if there is no job which currently could be processed on that machine. This concept was introduced by Racsamány (cf. Bárány & Fiala [1982]). It has been shown that the makespan of any dense schedule is at most twice the optimum makespan. We remark that this result can also be derived as a corollary from a more general theorem by Aksenov [1988]. In this section, we show that dense schedules also yield a 2-approximation algorithm for the multiprocessor problem $O(P)||C_{\text{max}}$. For that, we use the following algorithm that produces dense schedules.
Algorithm A

Starting at time $t = 0$, repeat the following step until all operations have been scheduled. Every time some machine $M$ in some stage $i$ is idle, check whether there exists an operation $O_{ij}$ such that (i) $O_{ij}$ has not been processed yet, and (ii) no other operation of the corresponding job $J_j$ is currently being processed on some other machine. If one or more such operations $O_{ij}$ exist, start processing one of them on machine $M$; ties are broken in favor of the job with the lowest index.

Now consider some fixed instance of $O(P)\mid C_{\text{max}}$. Two straightforward lower bounds on the optimum makespan $C^*_{\text{max}}$ are the average machine load in stage $i$

$$L_i := \frac{1}{m_i} \sum_{j=1}^{n} p_{ij} \leq C^*_{\text{max}} \quad \text{for } 1 \leq i \leq s, \quad (1)$$

and the total processing time of job $J_j$

$$P_j := \sum_{i=1}^{s} p_{ij} \leq C^*_{\text{max}} \quad \text{for } 1 \leq j \leq n. \quad (2)$$

Let $\sigma$ be the schedule that is produced by Algorithm A. We denote the makespan of $\sigma$ by $C_{\text{max}}(\sigma)$. Let $J_k$ be a job for which some operation completes at time $C_{\text{max}}(\sigma)$; without loss of generality we assume that $J_k$ is completed on stage one. We define $t$ as the starting time of job $J_k$ on stage one, i.e. $t = C_{\text{max}}(\sigma) - p_{1k}$. First, observe that during any period of idle time before time $t$ on any machine on stage one, job $J_k$ must be processed on another stage. Hence, the total idle time on stage one before time $t$ is at most $\sum_{i=2}^{m_1} p_{ik}$. Secondly, the total processing time on stage one of the jobs processed until time $t$ is at most $\sum_{j=1, j \neq k}^{n} p_{1j}$. From these two observations, we determine the following upper bound on $t$.

$$t \leq \frac{1}{m_1} (m_1 \sum_{i=2}^{s} p_{ik} + \sum_{j=1, j \neq k}^{n} p_{1j}).$$

Hence,

$$C_{\text{max}}(\sigma) = t + p_{1k} \leq \sum_{i=1}^{n} p_{ik} + \frac{1}{m_1} \sum_{j=1, j \neq k}^{n} p_{1j} \leq P_k + L_1 \leq 2C^*_{\text{max}}. \quad (3)$$

**Theorem 2.1** For the problem $O(P)\mid C_{\text{max}}$, there exists a polynomial-time approximation algorithm with worst case guarantee 2.  

It is also straightforward to construct instances for which the makespan $C_{\text{max}}(\sigma)$ of a dense schedule $\sigma$ constructed by Algorithm A is arbitrarily close to $2C^*_{\text{max}}$ (cf. Graham [1969]). Just let the first $m_1(m_1 - 1)$ jobs all have processing time 1 in the first stage, and let the last job have processing time $m_1$ in the first stage. The processing in all other stages takes time 0. Then $C_{\text{max}} = m_1$ and $C_{\text{max}}(\sigma) = 2m_1 - 1$. 
3 An approximation algorithm for the multiprocessor two-stage open shop

In this section, we take a closer look at the multiprocessor two-stage open shop problem. In Section 3.1 we define an auxiliary makespan minimization problem on parallel machines, and we investigate the connections between this auxiliary problem and $O2(P)||C_{\text{max}}$. In Section 3.2 we then derive a $\left(\frac{3}{2} + \varepsilon\right)$-approximation algorithm for $O2(P)||C_{\text{max}}$ that is based on these connections.

3.1 Two auxiliary problems on parallel machines

Consider an arbitrary instance $I$ of $O2(P)||C_{\text{max}}$ with job processing times $p_{ij}$ (1 $\leq$ $i$ $\leq$ 2, 1 $\leq$ $j$ $\leq$ $n$), with $m_1$ machines on the first stage and $m_2$ machines on the second stage. Let $\lambda$ be a positive number that fulfills

$$\lambda \geq \max_{i,j} p_{ij}. \quad (4)$$

We construct from $I$ two instances $P'(\lambda)$ and $P''(\lambda)$ of the makespan minimization problem with release dates on parallel machines, $P[r_j|C_{\text{max}}]$. In instance $P'(\lambda)$, there are $m_1$ parallel machines and $n$ jobs $J'_1, \ldots, J'_n$. The processing time $p'_j$ of job $J'_j$ is $p_{1j}$ (i.e. the length of the first-stage operation of job $J'_j$). Its release date is $r'_j$, where

$$r'_j = \begin{cases} 0, & \text{if } p_{2j} \leq \lambda/2 \\ 2p_{2j} - \lambda, & \text{otherwise.} \end{cases} \quad (5)$$

Instance $P''(\lambda)$ is defined in a symmetric way, with the roles of $p_{1j}$ and $p_{2j}$ exchanged. There are $m_2$ parallel machines, and the $n$ jobs $J''_1, \ldots, J''_n$ have processing times $p''_j = p_{2j}$ and release dates $r''_j$, where

$$r''_j = \begin{cases} 0, & \text{if } p_{1j} \leq \lambda/2 \\ 2p_{1j} - \lambda, & \text{otherwise.} \end{cases} \quad (6)$$

Note that $\lambda \geq \max_{i,j} p_{ij}$ implies that all release dates $r'_j$ and $r''_j$ are at most $\lambda$.

Lemma 3.1 If the instance $I$ of the open shop problem $O2(P)||C_{\text{max}}$ has a feasible schedule with makespan at most $\lambda$, then the instances $P'(\lambda)$ and $P''(\lambda)$ of problem $P[r_j|C_{\text{max}}]$ both have feasible schedules with makespan at most $\lambda$.

Proof. We only prove the statement for the instance $P'(\lambda)$; the proof for $P''(\lambda)$ is done in a symmetric way.

Consider a feasible schedule $\sigma$ for $I$ with makespan $\lambda$, which exists by the assumptions of the lemma. For $1 \leq k \leq m_1$, let $\mathcal{M}_k$ denote the set of indices of the open shop jobs $J_j$ whose first operation is processed on the $k$-th stage-one machine in schedule $\sigma$. Note that this implies

$$\sum_{j \in \mathcal{M}_k} p'_j \leq \lambda \quad (7)$$
for every $k$. Define a feasible schedule $\sigma'$ for $I'(\lambda)$ as follows. For $1 \leq k \leq m_1$, the $k$-th machine in schedule $\sigma'$ processes exactly the jobs $J'_j$ with index $j \in M'_k$. The jobs are processed in order of increasing due date by the Earliest Due Date rule (EDD-rule). Then the completion time $C'_k$ of the $k$-th machine is given by

$$C'_k = \max_{\tau \leq \lambda} \left( \tau + \sum \left\{ p'_j \mid j \in M'_k \text{ and } r'_j \geq \tau \right\} \right).$$

(8)

It remains to be shown that $C'_k \leq \lambda$ holds for all $1 \leq k \leq m_1$. We do this by proving that for every $\tau$, $0 \leq \tau \leq \lambda$, the value in the right-hand side of the expression (8) is bounded from above by $\lambda$. Indeed, if $\tau = 0$, the correctness of this claim follows directly from inequality (7). Hence, we assume that $\tau > 0$.

Consider some job $J'_j$ with $r'_j \geq \tau$. By rewriting the inequality (5) one sees that in the open shop instance $I$ the processing time of the stage-two operation $O_{2j}$ fulfills $p_{2j} \geq \frac{1}{2}(\lambda + \tau)$. Consequently, in schedule $\sigma$ the processing of operation $O_{2j}$ starts between time 0 and time $\frac{1}{2}(\lambda - \tau)$, and it completes between time $\frac{1}{2}(\lambda + \tau)$ and time $\lambda$. Hence, the processing of $O_{2j}$ always covers the time interval $\left[ \frac{1}{2}(\lambda - \tau), \frac{1}{2}(\lambda + \tau) \right]$, and during this time interval the corresponding stage-one operation $O_{1j}$ cannot be processed. This yields that all stage-one operations $O_{1j}$ with $p_{2j} \geq \frac{1}{2}(\lambda + \tau)$ are processed during the two time intervals $\left[ 0, \frac{1}{2}(\lambda - \tau) \right]$ and $\left[ \frac{1}{2}(\lambda + \tau), \lambda \right]$. This in turn yields that the total processing time of all such operations that are processed on the $k$-th machine has to be less than or equal to $\lambda - \tau$. The claim follows.

**Lemma 3.2** If the instances $I'(\lambda)$ and $I''(\lambda)$ of problem $P|p_j|C_{\text{max}}$ both have feasible schedules with makespan at most $\lambda$, then the instance $I$ of the open shop problem $O2(P)||C_{\text{max}}$ has a feasible schedule with makespan at most $\frac{3}{2}\lambda$.

**Proof.** Let $\sigma'$ and $\sigma''$ denote the schedules with makespan at most $\lambda$ for the instances $I'(\lambda)$ and $I''(\lambda)$, respectively. For $1 \leq k \leq m_1$, let $M'_k$ denote the set of indices of the jobs $J'_j$ that are processed on the $k$-th machine in schedule $\sigma'$; for $1 \leq k \leq m_2$, let $M''_k$ denote the set of indices of the jobs $J''_j$ that are processed on the $k$-th machine in schedule $\sigma''$.

From this, we define a feasible schedule $\sigma$ for the open shop instance $I$ as follows. In $\sigma$, the $k$-th stage-one machine processes exactly the stage-one operations of jobs with indices in $M'_k$, and the $k$-th stage-two machine processes exactly the stage-two operations of jobs with indices in $M''_k$. These sets of operations are said to be *preassigned* to their corresponding machines. The precise schedule $\sigma$ is constructed by the following algorithm.

**Algorithm B**

*Initialization.* For any machine $M$ for which there is a preassigned operation of length $\frac{1}{2}\lambda$, this operation is scheduled on machine $M$ at time 0.

*Loop.* Repeat the following step until all operations have been scheduled. Every time some machine $M$ is idle, check whether there exists an operation (i) that has been preassigned to $M$, (ii) that has not yet been processed, and (iii) whose job is not simultaneously being processed on another machine. If one or more such operations exist, start processing one of them on machine $M$; ties are broken in favor of the job with the lowest index.
Since the schedules \( \sigma' \) and \( \sigma'' \) have makespan at most \( \lambda \), the total length of the operations preassigned to any machine is at most \( \lambda \). Hence, for every machine there is at most one preassigned job of length greater than \( \frac{1}{2} \lambda \), and the initialization step is well defined. The loop constructs a kind of dense schedule, in an analogous way as Algorithm A does.

Consider the schedule \( \sigma \) that is produced by Algorithm B, let \( C_{\text{max}}(\sigma) \) denote the makespan of this schedule, and let \( M \) denote a machine that completes at time \( C_{\text{max}}(\sigma) \). Without loss of generality, we assume that \( M \) is a stage-two machine. Note that the total length of the preassigned operations for \( M \) is at most \( \lambda \). If \( M \) is never idle, then \( C_{\text{max}}(\sigma) \leq \lambda \) and there is nothing to show. Hence, we assume that \( M \) is idle for some time and we denote by \text{IDLE} the total amount of idle time on machine \( M \). Let \( Z \) denote the set of operations that are processed after the last idle interval on \( M \), let \( p(Z) \) denote the total length of the operations in \( Z \), and let \( \tau \) denote the starting time of the earliest operation in \( Z \).

Let \( O_{2j} \) be an arbitrary operation in \( Z \). Every time machine \( M \) was idle, operation \( O_{2j} \) could not be started since \( O_{1j} \) was run on some other machine at that time. Consequently,

\[
\text{IDLE} \leq p_{1j} \tag{9}
\]

must hold. Now we finally prove that \( C_{\text{max}}(\sigma) \leq \frac{3}{2} \lambda \) holds. Since \( C_{\text{max}}(\sigma) \leq \lambda + \text{IDLE} \), inequality (9) proves \( C_{\text{max}}(\sigma) \leq \frac{3}{2} \lambda \) for the cases where \( p_{1j} \leq \frac{1}{2} \lambda \) holds for some operation \( O_{1j} \) with \( O_{2j} \) in \( Z \). Hence, it remains to consider the case where \( p_{1j} > \tau > \frac{1}{2} \lambda \) holds for all operations \( O_{1j} \) with \( O_{2j} \) in \( Z \). For such operations, \( O_{1j} \) is the first operation processed by its machine. Now let us argue via the combinatorial structure of schedule \( \sigma'' \) for \( I''(\lambda) \). The jobs \( J''_j \) that correspond to the operations \( O_{2j} \) in \( Z \) all have release dates at least \( 2 \tau - \lambda \). Since \( \sigma'' \) has makespan \( \leq \lambda \), this yields that

\[
p(Z) \leq \lambda - (2 \tau - \lambda) = 2(\lambda - \tau). \tag{10}
\]

Since \( \tau > \frac{1}{2} \lambda \), we finally arrive at

\[
C_{\text{max}}(\sigma) = \tau + p(Z) \leq \tau + 2(\lambda - \tau) \leq \frac{3}{2} \lambda. \tag{11}
\]

This completes the proof of Lemma 3.2.

**Lemma 3.3** Let \( \lambda_1 \leq \lambda_2 \) be positive real numbers. If the instances \( I'(\lambda_1) \) and \( I''(\lambda_1) \) of problem \( P[r_j] | C_{\text{max}} \) both have feasible schedules with makespan at most \( \lambda_1 \), then the instances \( I'(\lambda_2) \) and \( I''(\lambda_2) \) both have feasible schedules with makespan at most \( \lambda_2 \).

**Proof.** The two instances \( I'(\lambda_1) \) and \( I'(\lambda_2) \) differ only in the job release dates. By the definition in (5), the release dates in \( I'(\lambda_2) \) are all less than or equal to the corresponding release dates in \( I'(\lambda_1) \), and hence they are less stringent.

### 3.2 The approximation algorithm

Throughout this subsection, our goal is to find a feasible schedule for a given instance of problem \( O2[P] | C_{\text{max}} \) whose makespan is at most a factor of \( \frac{3}{2} + \varepsilon \) away from the optimum, where \( \varepsilon \) is a positive constant smaller than 1. We define \( \delta \) to be the root of the equation

\[
\frac{3}{2}(1 + \delta)^2 = \frac{3}{2} + \varepsilon \tag{12}
\]

that fulfills \( 0 < \delta < 1 \).
Proposition 3.4 (Hall & Shmoys [1989]) There exists an algorithm Hall-Shmoys with polynomial running time for the problem $P|r_j|C_{\max}$ that behaves as follows. Hall-Shmoys takes as input an instance $I^*$ of $P|r_j|C_{\max}$ and it outputs a schedule with makespan at most $1+\delta$ times the optimum makespan of $I^*$.

With the help of the notation of Section 2 as introduced in (1) and (2), we define a value $\text{aux}$ by

$$\text{aux} = \max_{1 \leq j \leq n} P_j + \max_{1 \leq i \leq 2} L_i.$$  \hfill (12)

By inequality (3), $\frac{1}{2}\text{aux} \leq C^*_{\max} \leq \text{aux}$ holds. This is also the initialization of the binary search in the following algorithm.

**MAIN**

Set $\text{upp} = \text{aux}$ and $\text{low} = \frac{1}{2}\text{aux}$.

while $(1+\delta)\text{low} < \text{upp}$ do

$\lambda = \frac{1}{2}(\text{upp} + \text{low})$

if Hall-Shmoys($I'(\lambda)$) and Hall-Shmoys($I''(\lambda)$) both yield

schedules with makespan at most $(1+\delta)\lambda$

then $\text{upp} := \lambda$

else $\text{low} := \lambda$

end while.

Let $\lambda^*$ be the current value of $\text{upp}$.

With the help of Algorithm B, compute from the schedules for $I'(\lambda^*)$

and $I''(\lambda^*)$ a schedule for $I$ with makespan at most $\frac{3}{2}\lambda^*$.

Let us first argue that throughout the while-loop in MAIN, the inequality $\text{low} \leq C^*_{\max}$ holds. This certainly is true when the while-loop is entered for the first time, since $\frac{1}{2}\text{aux} \leq C^*_{\max}$. Inside the while-loop, $\text{low}$ is only updated to $\lambda$ if Hall-Shmoys($I'(\lambda)$) or Hall-Shmoys($I''(\lambda)$) yield a schedule for $I'(\lambda)$ or for $I''(\lambda)$ with makespan larger than $(1+\delta)\lambda$. By Proposition 3.4, in this case the optimum makespan of $I'(\lambda)$ or $I''(\lambda)$ is larger than $\lambda$, and by Lemma 3.1 the optimum makespan $C^*_{\max}$ of $I$ is larger than $\lambda$. Consequently, $\text{low} \leq C^*_{\max}$ holds indeed throughout the while-loop.

Next, we argue that throughout MAIN we know feasible schedules for $I'(\lambda)$ and $I''(\lambda)$ with makespan at most $(1+\delta)\text{upp}$. By Lemma 3.1, this is true when the while-loop is entered for the first time since $C^*_{\max} \leq \text{aux}$ holds. Every time $\text{upp}$ is updated inside the while-loop, the claim follows from the condition in the if-then statement.

When the stopping condition of the while-loop is fulfiled, $(1+\delta)\text{low} \geq \text{upp}$ holds. Hence, at the end of the while-loop we have

$$\lambda^* \leq (1+\delta)\text{low} \leq (1+\delta)C^*_{\max}$$  \hfill (13)

and we also have feasible schedules for $I'(\lambda^*)$ and $I''(\lambda^*)$ with makespan at most $\mu := (1+\delta)\lambda^*$. By Lemma 3.3, we get schedules for $I'(\mu)$ and $I''(\mu)$ with makespan at most $\mu$. Algorithm B transforms these schedules into a schedule for $I$ with makespan at most

$$\frac{3}{2}\mu = \frac{3}{2}(1+\delta)\lambda^* \leq \frac{3}{2}(1+\delta)^2C^*_{\max} = (\frac{3}{2} + \epsilon)C^*_{\max}.$$  \hfill (14)
Finally, we note that the number of steps MAIN performs is polynomial in the size of the input. Summarizing, for any instance $I$ of $O2(\{P\}||C_{\text{max}})$, MAIN outputs a schedule with makespan at most $(\frac{3}{2} + \varepsilon)C^{*}_{\text{max}}$ in polynomial time.

**Theorem 3.5** For every $\varepsilon > 0$, there exists a polynomial-time approximation algorithm that computes for any instance $I$ of problem $O2(\{P\}||C_{\text{max}})$ a schedule with makespan at most $(\frac{3}{2} + \varepsilon)C^{*}_{\text{max}}$.

We conclude this section with the observation that there are instances for which the schedule that algorithm MAIN outputs has makespan arbitrarily close to $\frac{5}{4}C^{*}_{\text{max}}$. Consider the following instance with two machines on the first stage and one on the second. There are two jobs, each with processing time $\frac{1}{2} + \varepsilon$ in the first stage and processing time $\frac{1}{2} - \varepsilon$ in the second stage. Clearly our algorithm outputs a schedule of length $\frac{5}{4} - \varepsilon$, while the optimal makespan equals 1.

<table>
<thead>
<tr>
<th>Number of stages</th>
<th>1</th>
<th>constant</th>
<th>part of input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of machines per stage</td>
<td>poly-time</td>
<td>FPTAS</td>
<td>PTAS</td>
</tr>
<tr>
<td>constant ≥ 3</td>
<td>FPTAS</td>
<td>FPTAS</td>
<td>PTAS</td>
</tr>
<tr>
<td>part of input</td>
<td>wcr$^*$ = $\frac{3}{2}$</td>
<td>wcr$^*$ = $\frac{3}{2}$</td>
<td>wcr$^*$ = $\frac{3}{2}$</td>
</tr>
</tbody>
</table>

Table 2: The conjectured approximability landscape of the multiprocessor open shop.

4 Discussion

In this paper, we have derived two new approximability results for the multiprocessor open shop scheduling problem. First, we constructed a polynomial-time approximation algorithm with worst case ratio 2 for $O(P)||C_{\text{max}}$. Secondly, for the two-stage problem $O2(\{P\}||C_{\text{max}}$ we derived a polynomial-time approximation algorithm whose worst case ratio can be made arbitrarily close to $\frac{3}{2}$.

Clearly, our results are just an intermediate step in understanding the approximability behavior of the open shop. We conjecture that the known approximability results for the multiprocessor open shop will eventually look like in Table 2. In this table, wcr$^*$ means the best possible worst case ratio that can be reached under $P \neq NP$. Comparing the conjectured results in Table 2 to the currently known results in Table 1, one sees that we have in fact formulated eight conjectures. For the first disproof of any of these eight conjectures, we offer a reward of ten US dollars.
References


