Approximating Two-Machine Flow Shop Problem with Delays when Processing Times Depend Only on Machines

Alexander Ageev and Alexei Baburin

Sobolev Institute of Mathematics, pr. Koptyuga 4, Novosibirsk, Russia
ageev@math.nsc.ru, ababur@sicex.ru

Abstract. We study the two-machine flow shop problem with (minimum) delays.The current best approximation factor achieved for this problem is $11/6 \approx 1.833$ (due to Karuno and Nagamochi; their algorithm runs in time $O(n \log n)$). Our main result is a 1.628-approximation algorithm for the case when processing times depend only on machines. This case is strongly NP-hard even when all processing times are equal. The running time of our algorithm is $O(n^2)$.

Keywords: approximation algorithm, scheduling problem, flow shop, minimum delays, worst-case ratio

Introduction

In the two-machine flow shop problem with (minimum) delays there are two machines available from time zero onwards for processing n jobs. Each machine can process at most one job at a time. Each job j consists of two operations; the second operation must start no earlier than l_j time units after the completion of the first operation. The first (second) operation has to be executed by machine 1 (machine 2) and processing the first (second) operation takes time p_{1j} (p_{2j}). The objective is to minimize the makespan, or the schedule length, that is the maximum job completion time. As in [13], we denote this problem by $F2 \mid l_j \mid C_{\max}$.

The problems with minimum delays arise, in particular, in manufacturing where there may be a transportation time from one production facility to another, and in computer systems where the output of a task on one processor may require a communication time so as to become the input to a subsequent task on another processor.

The first result related to $F2 \mid l_j \mid C_{\max}$ is due to Johnson [5] who presents an $O(n \log n)$ algorithm for $F2 \mid\mid C_{\max}$. Johnson [6] and Mitten [10] show that a modification of Johnson's algorithm for $F2 \mid\mid C_{\max}$ can be used to find an optimal permutation schedule, i.e., the schedule in which the jobs are executed by each machine in the same order. If all delays are equal, the set of optimal schedules contains a permutation schedule. However, this is not the case when there are at least two distinct delay values (see [6]. Kern and Nawijn [8] consider a single-machine problem with two operations per jobs

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

and intermediate minimum delays. Following the extension [13] of the three-field notation scheme introduced by Graham et al. [4] we denote this problem by $1 \mid l_j \mid C_{\max}$. Yu et al. [12, 13] show that this problem is equivalent to $F2 \mid l_j \mid C_{\max}$. Kern and Nawijn [8] show that $1 \mid l_j \mid C_{\max}$ is weakly NP-hard. This result is strengthened to NP-hardness in the strong sense for $F2 \mid l_j \mid C_{\max}$ by Lenstra [9], for $F2 \mid l_j, p_{1j} = p_{2j} \mid C_{\max}$ by Dell'Amico and Vaessens [3], and for $F2 \mid l_j \in \{0, l\}, p_{1j} = p_{2j} \mid C_{\max}$ by Yu [12]. Yu et al. [13] prove that $F2 \mid l_j, p_{ij} = 1 \mid C_{\max}$ is NP-hard in the strong sense. Dell'Amico [2] presents several 2-approximation algorithms with running time $O(n \log n)$ where n is the number of jobs. Karuno and Nagamochi [7] develop an 11/6 -approximation algorithm with running time $O(n \log n)$. Ageev [1] presents a 3/2-approximation algorithm for $F2 \mid l_j, p_{1j} = p_{2j} \mid C_{\max}$. Zhang and van de Velde [14] present a PTAS for $F2 \mid l_j \mid C_{\max}$ under the assumption that $l_{\max} \leq \mu p_{\min}$ where $l_{\max}$ is the maximum delay, $p_{\min}$ is the smallest positive processing time of any operation and μ is a fixed number (that is, it is not part of the problem instance).

The question whether there exists a polynomial-time algorithm for the problem $1 \mid l_j \mid C_{\max}$ with a better approximation ratio remains open though it has been posed by several researchers (see, for example, Strusevich [11]).

We consider the special case where $p_{1j} = a$ and $p_{2j} = b$ for all jobs j, or problem $F2 \mid l_j, p_{1j} = a, p_{2j} = b \mid C_{\max}$. Notice that this problem remains strongly NP-hard as it follows from the above mentioned result due to Yu et al. [13]. We present an approximation algorithm, whose approximation ratio is bounded by $\min\{1+\frac{2q+2}{q+4}, 2-q\}$ where

$$q = \frac{\max\{a, b\} - \min\{a, b\}}{\max\{a, b\}}.$$

The algorithm can be implemented in $O(n^2)$ time. Since it can be shown that $\min\{1 + \frac{2q+2}{q+4}, 2 - q\} < 1.628$, this implies a 1.628-approximation for $F2 \mid l_j, p_{1j} = a, p_{2j} = b \mid C_{\max}$. For the case when $a = b$ we get a 3/2-approximation. Note that the same approximation factor for this case follows from the above mentioned result by Ageev [1]. One of the ingredients of the algorithm is an observation that the length of *any* reasonable schedule in $F2 \mid l_j \mid C_{\max}$ is at most twice the length of a shortest schedule. The reasonable schedule (we further call them short) is meant to be the schedule in which given an arbitrary order of executing jobs on machine 1 (machine 2), the order of executing jobs on machine 2 (machine 1) is optimal if $\sum_j p_{1j} \leq \sum_j p_{2j}$ (if $\sum_j p_{1j} > \sum_j p_{2j}$). Note that given a job order on one machine, finding an optimal order on the other is equivalent to solving the trivial single-machine problem with release times. As a by-product this observation provides a fairly simple 2-approximation algorithm for solving $F2 \mid l_j \mid C_{\max}$. The crucial part of our analysis is a lower bound for the length of a shortest schedule which is a straightforward extension of an observation due to Yu [12].

The paper is organized as follows. In Section 2 we introduce basic notation and definitions. We also make a few observations and assumptions that do not restrict generality. In Section 3 we give a simple 2-approximation for $F2 \mid l_j \mid C_{\max}$. In Section 4 we present an approximation algorithm for $F2 \mid l_j, p_{1j} = a, p_{2j} = b \mid C_{\max}$.

1 Preliminaries

Before proceeding to the main part we introduce basic notation and make some observations and assumptions that do not restrict generality. An instance of $F2 \mid l_j \mid C_{\max}$ consists of a set of jobs $J = \{1, \ldots, n\}$. Each job $j \in J$ consists of two operations $O_{1,j}$ and $O_{2,j}$ whose lengths will be denoted by a_j and b_j, respectively. For each $j \in J$ operation $O_{2,j}$ is separated from $O_{1,j}$ by a delay of length at least l_j time units. For any $j \in J$, we denote by $\sigma(1, j)$ and $\sigma(2, j)$ the starting times of $O_{1,j}$ and $O_{2,j}$, respectively. Note that

$$\sigma(2, j) \geq \sigma(1, j) + a_j + l_j.$$

for all $j \in J$. For a schedule $\sigma = (\sigma(1, 1), \sigma(2, 1), \ldots, \sigma(2, n))$, denote by $C_j(\sigma)$ the completion time of job j in σ; then $C_j(\sigma) = \sigma(2, j) + b_j$. The length of a schedule σ is $C_{\max}(\sigma) = \max_{j \in J} C_j(\sigma)$. Let $A = \sum_{j \in J} a_j$, $B = \sum_{j \in J} b_j$, and $L = \max_{j \in J} l_j$. Let $C^*_{\max}$ denote the length of a shortest schedule.

Observe that any feasible schedule for an instance of $F2 \mid l_j \mid C_{\max}$ can be replaced by a schedule such that it has the same orders of jobs on machines and the same length but both machines process the jobs without idle times. Thus we may assume that both machines process the jobs without idle times and machine 1 starts processing at time 0. Then any feasible schedule σ defines a pair of permutations (φ, ψ) of the set J such that φ specifies the order of processing operations on machine 1 and ψ specifies that on machine 2. More specifically, $\varphi(k)$ ($\psi(k)$) is the k-th job executed by machine 1 (machine 2) (see Fig 1). Thus if (φ, ψ) is the pair of permutations of a schedule σ, then

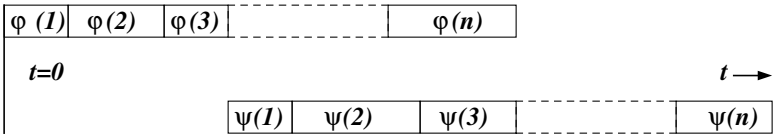

Fig. 1. $\varphi(k)$ ($\psi(k)$) is the k-th job executed by machine 1 (machine 2)

$\sigma(1, \varphi(1)) = 0$ and for any $k = 2, \ldots, n$,

$$\sigma(1, \varphi(k)) = \sigma(1, \varphi(k-1)) + a_{\varphi(k-1)},$$
$$\sigma(2, \psi(k)) = \sigma(2, \psi(k-1)) + b_{\psi(k-1)}.$$

Note that for any $j \in J$, $\varphi^{-1}(j)$ ($\psi^{-1}(j)$) means the order number in which job j is processed on machine 1 (machine 2). We will further represent the permutations φ and ψ by the sequences $(\varphi(1), \ldots, \varphi(n))$ and $(\psi(1), \ldots, \psi(n))$.

For any pair of permutations (φ, ψ), denote by $[\varphi, \psi]$ the unique schedule having the shortest length among all feasible schedules that run jobs according to the permutations (φ, ψ). We say that σ is an *early* schedule if $\sigma = [\varphi, \psi]$. Note that given a pair of

permutations (φ, ψ), the early schedule $[\varphi, \psi]$ can be found in linear time. Indeed, to find σ it suffices to determine $x = \sigma(2, \psi(1))$. For any $k = 1, \ldots, n$ we have

$$\sigma(2, \psi(k)) = x + \sum_{i=1}^{k-1} b_{\psi(i)} \geq \sigma(1, \varphi(k)) + a_{\varphi(k)} + l_{\varphi(k)}.$$

Thus $x = \min\{\sigma(1, \varphi(k)) + a_{\varphi(k)} + l_{\varphi(k)} - \sum_{i=1}^{k-1} b_{\psi(i)} : k = 1, \ldots, n\}$, which can be found in linear time.

It is easy to see that a schedule σ is feasible if and only if

$$\min\{\sigma(2, j) - \sigma(1, j) - a_j - l_j : j \in J\} \geq 0.$$

Observe that for a schedule σ with the job permutations (φ, ψ)

$$\min\{\sigma(2, j) - \sigma(1, j) - a_j - l_j : j \in J\} = 0 \tag{1}$$

if and only if $\sigma = [\varphi, \psi]$.

2 Short schedules

Let σ be a feasible schedule for an instance of $F2 \mid l_j \mid C_{\max}$. Since the length of a shortest schedule is at least the maximum machine load and the maximum job length,

$$C^*_{\max} \geq \Lambda = \max\{A, B, \max_{j \in J}\{a_j + b_j + l_j\}\}. \tag{2}$$

We say that a feasible schedule is *short* if it has the shortest length among all schedules with the same (fixed) order of processing the operations on machine 1 when $A \leq B$ and if it has the shortest length among all schedules with the same (fixed) order of processing the operations on machine 2 when $A > B$. It is clear that any optimal schedule is a short schedule.

Given a job order on machine 1, a corresponding short schedule can be found in $O(n \log n)$ time (it is obviously equivalent to solving the single-machine scheduling problem with release times). The following theorem shows that the length of any short schedule is at most twice the length of a shortest schedule, thereby providing a fairly simple 2-approximation algorithm for solving $F2 \mid l_j \mid C_{\max}$.

Theorem 1. *Let I be an instance of $F2 \mid l_j \mid C_{\max}$ and let σ be a short schedule of I. Then $C_{\max}(\sigma) \leq (1 + \frac{\min\{A,B\}}{\Lambda})C^*_{\max}$.*

Proof. Since the problem is symmetric with respect to the machine indices and the time axis, it suffices to consider the case when $A \leq B$. Let (φ, ψ) be the pair of job permutations corresponding to σ and let (φ^*, ψ^*) be that corresponding to an optimal schedule σ^*. Consider the schedule σ' whose pair of jobs permutations is that of σ^* and such that all operations on machine 2 start exactly A time units later than in σ^*. In other words, for any $j \in J$,

$$\begin{aligned} \sigma'(1, j) &= \sigma^*(1, j), \\ \sigma'(2, j) &= \sigma^*(2, j) + A. \end{aligned} \tag{3}$$

It is clear that σ' is a feasible schedule. Note that by (2) $A = \frac{A}{\Lambda}\Lambda \leq \frac{A}{\Lambda}C^*_{\max}$. Thus by the construction of σ',

$$C_{\max}(\sigma') = C_{\max}(\sigma^*) + A \leq C_{\max}(\sigma^*) + \frac{A}{\Lambda}C^*_{\max} = (1 + \frac{A}{\Lambda})C^*_{\max}. \quad (4)$$

Now consider the schedule σ'' whose pair of jobs permutations is (φ, ψ^*) and such that $\sigma''(2, \psi^*(1)) = \sigma'(2, \psi^*(1))$. Thus the schedule σ'' may differ from the schedule σ' on machine 1 and coincides with σ' on machine 2. Since the completion time of the last operation on machine 1 is A for all schedules, $\sigma''(1, j) \leq \sigma^*(1, j) + A$ and so

$$\begin{aligned} \sigma''(1, j) + a_j + l_j &\leq \sigma^*(1, j) + a_j + l_j + A \\ &\leq \sigma^*(2, j) + A \\ \text{(by (3))} \quad &= \sigma'(2, j) = \sigma''(2, j) \end{aligned}$$

for all $j \in J$. It follows that σ'' is a feasible schedule. Since σ' and σ'' coincide on machine 2, $C_{\max}(\sigma'') = C_{\max}(\sigma')$. However, since σ and σ'' have the same jobs permutation φ on machine 1 and σ is a short schedule, we have that $C_{\max}(\sigma) \leq C_{\max}(\sigma'')$. Finally, by (4) we obtain

$$C_{\max}(\sigma) \leq C_{\max}(\sigma'') = C_{\max}(\sigma') \leq (1 + \frac{A}{\Lambda})C^*_{\max}.$$

□

Given a feasible schedule σ, we will denote by $\overline{\sigma}$ a short schedule whose order of jobs on machine 1 coincides with that of σ if $A \leq B$ and whose order of jobs on machine 2 coincides with that of σ if $A > B$. As it was mentioned above, $\overline{\sigma}$ can be found in $O(n \log n)$ time.

3 Algorithm for $F2 \mid l_j, p_{1j} = a, p_{2j} = b \mid C_{\max}$

In this section we present an approximation algorithm for the special case of the problem when $p_{1j} = a$ and $p_{2j} = b$ for all $j \in J$. Recall that for brevity we set $a_j = p_{1j}$ and $b_j = p_{2j}$.

Since the the case when $a \leq b$ trivially reduces to the case when $a \geq b$ (and vice versa), we will further assume that $a \geq b$. Recall that this case of the problem remains strongly NP-hard as it follows from the NP-hardness result due to Yu et al. [13].

We now proceed to the description of the algorithm. The algorithm successively constructs k schedules, selects a schedule having minimum length among them and outputs the short schedule obtained from it. The schedules are generated by cyclic shifts of operations executed on machine 1 while the operations on machine 2 are executed in a fixed order. The idea behind the construction is to apply Lemma 1 (see below) in the worst-case analysis.

Algorithm SPECIAL

Step 0. Sort the jobs in nondecreasing order of delays. For convenience, we will further assume that

$$l_1 \le l_2 \le \ldots \le l_n = L. \tag{5}$$

Step k. For $k = 1, \ldots, n$ construct the schedule $\sigma_k = [\varphi_k, \psi]$ where $\psi = (1, \ldots, n)$ and $\varphi_k = (k+1, \ldots, n, 1, \ldots, k)$ if $k \le n-1$ and $\varphi_n = (1, \ldots, n)$ (see Fig. 2).

Pick out a schedule $\sigma \in \{\sigma_1, \ldots, \sigma_n\}$ having the shortest length and construct a schedule $\tau = \overline{\sigma}$. Output τ.

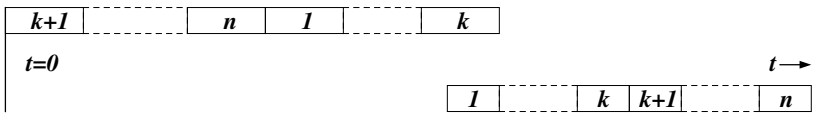

Fig. 2. The schedule constructed at Step 2 of algorithm SPECIAL

Running time. As it was shown in Section 2, the early schedule σ_k for each $k = 1, \ldots, n$ can be constructed in linear time. Furthermore, as it was noticed in Section 3, the schedule $\overline{\sigma}$ can be obtained from σ in $O(n \log n)$ time. Thus the total running time of the algorithm is $O(n^2)$.

Approximation ratio. We first show a lower bound for $C^*_{\max}$ that will play a crucial role in establishing an upper bound on the approximation ratio of algorithm Special. The lemma below is a straightforward extension of an observation due to Yu [12, 13] (Lemma 2 in [13]).

Lemma 1.

$$C^*_{\max} \ge \left\lceil \frac{(a+b)(n+1)}{2} + \frac{\sum_{j \in J} l_j}{n} \right\rceil \tag{6}$$

Proof. Let σ be a feasible schedule with the job permutations φ and ψ. Then for any $j \in J$,

$$\begin{aligned} C_{max}(\sigma) &\ge \sigma(1, j) + a_j + l_j + \sum_{k=\psi^{-1}(j)}^{n} b_{\psi(k)} \\ &= \sum_{k=1}^{\varphi^{-1}(j)} a_{\varphi(k)} + l_j + \sum_{k=\psi^{-1}(j)}^{n} b_{\psi(k)}. \end{aligned}$$

By taking into account that $a_j \equiv a$ and $b_j \equiv b$, it follows that

$$C_{max}(\sigma) \ge \frac{1}{n}\Big(\sum_{j \in J} \big(\sum_{k=1}^{\varphi^{-1}(j)} a_{\varphi(k)} + l_j + \sum_{k=\psi^{-1}(j)}^{n} b_{\psi(k)} \big) \Big)$$

$$
\begin{aligned}
&= \frac{1}{n}\Big(\sum_{j\in J}\sum_{k=1}^{\varphi^{-1}(j)} a_{\varphi(k)} + \sum_{j\in J}\sum_{k=\psi^{-1}(j)}^{n} b_{\psi(k)}\Big) + \frac{\sum_{j\in J} l_j}{n} \\
&= \frac{1}{n}\Big(a\sum_{j\in J}\varphi^{-1}(j) + b\sum_{j\in J}(n-\psi^{-1}(j)+1)\Big) + \frac{\sum_{j\in J} l_j}{n} \\
&= \frac{1}{2n}\Big(an(n+1) + bn(n+1)\Big) + \frac{\sum_{j\in J} l_j}{n} \\
&= \frac{(n+1)(a+b)}{2} + \frac{\sum_{j\in J} l_j}{n}.
\end{aligned}
$$

By combining (6) with (2), we obtain the following lower bound:

$$C^*_{\max} \geq LB = \max\{a+b+L, an, \frac{(a+b)(n+1)}{2} + \frac{\sum_j l_j}{n}\}. \tag{7}$$

We now proceed to establishing an upper bound on the length of the schedule returned by algorithm `Special`.

Lemma 2. *For any $k = 1, \ldots, n$,*

$$C_{\max}(\sigma_k) \leq \max\{X_k, Y_k\} \tag{8}$$

where $X_k = a(n-k) + b + L$ and $Y_k = an + b(n-k+1) + l_k$.

Proof. Let $1 \leq k \leq n$. Note that by construction σ_k satisfies (1). Since $\sigma_k(1,k) = a(n-1)$ and $\sigma_k(1,n) = a(n-k-1)$, (1) implies that

$$\min\{\sigma_k(2,k) - an - l_k, \sigma_k(2,n) - a(n-k) - l_n\} \geq 0.$$

We now claim that in fact

$$\min\{\sigma_k(2,k) - an - l_k, \sigma_k(2,n) - a(n-k) - l_n\} = 0. \tag{9}$$

Assume to the contrary that $\sigma_k(2,k) > an + l_k$ and $\sigma_k(2,n) > a(n-k) + l_n$. Assume first that $1 \leq j \leq k$. Then

$$\sigma_k(1,j) = \sigma_k(1,n) + a + a(j-1) = a(n-k-1) + aj = a(n-k+j-1)$$

and $\sigma_k(2,j) + b(k-j) = \sigma_k(2,k)$. Since $l_j \leq l_k$ and $a \geq b$, it follows that

$$
\begin{aligned}
\sigma_k(2,j) - \sigma_k(1,j) &= \sigma_k(2,k) - b(k-j) - a(n-k+j-1) \\
&> an + l_k - b(k-j) - a(n-k+j-1) \\
&= l_k + (a-b)(k-j) + a \\
&\geq l_j + a,
\end{aligned}
$$

i. e., $\sigma_k(2,j) - \sigma_k(1,j) - l_j - a > 0$. Assume now that $k+1 \leq j \leq n$. Then $\sigma_k(1,j) = a(j-k-1)$ and $\sigma_k(2,j) + b(n-j) = \sigma_k(2,n)$. Similarly to the above, we have

$$
\begin{aligned}
\sigma_k(2,j) - \sigma_k(1,j) &= \sigma_k(2,n) - b(n-j) - a(j-k-1) \\
&> a(n-k) + l_n - b(n-j) - a(j-k-1) \\
&= (a-b)(n-j) + l_n + a \\
&\geq l_j + a,
\end{aligned}
$$

which means that $\sigma_k(2,j) - \sigma_k(1,j) - l_j - a > 0$. Summing up, we arrive at the conclusion that $\sigma_k(2,j) - \sigma_k(1,j) - l_j - a > 0$ for all $j \in J$, which contradicts the fact that σ_k is an early schedule.

By the claim, either $\sigma_k(2,k) = an + l_k$, or $\sigma_k(2,n) = a(n-k) + l_n$. If the former holds, then

$$C_{\max}(\sigma_k) = \sigma_k(2,k) + b(n-k+1) = an + l_k + b(n-k+1) = Y_k.$$

If the latter is true, then

$$C_{\max}(\sigma_k) = \sigma_k(2,n) + b = a(n-k) + l_n + b = X_k.$$

Thus (8) does hold true, as required. □

Theorem 2. *Let I be an instance of the problem $F2 \mid l_j, p_{1j} = a, p_{2j} = b \mid C_{\max}$ and τ be a schedule output by algorithm* `SPECIAL`. *Then*

$$C_{\max}(\tau) \le \min\{1 + \frac{2q+2}{q+4}, 2-q\} C^*_{\max} \tag{10}$$

where $q = \frac{a-b}{a}$.

Proof. First, observe that since τ is a short schedule, by Theorem 1

$$C_{\max}(\tau) \le (1 + \frac{b}{a}) C^*_{\max} = (2-q) C^*_{\max}.$$

On the other hand, by the definition of short schedule, $C_{\max}(\tau) = C_{\max}(\overline{\sigma}) \le C_{\max}(\sigma)$. Thus, to prove the theorem it suffices to justify the following inequality:

$$C_{\max}(\sigma) \le \Big(1 + \frac{2q+2}{q+4}\Big) C^*_{\max}. \tag{11}$$

For $k = 1, \ldots, n$, let $\theta(k) = Y_k - X_k$. Since

$$\begin{aligned} \theta(k) &= an + b(n-k+1) + l_k - a(n-k) - b - L \\ &= (a-b)k + bn + l_k - L, \end{aligned}$$

$\theta(k)$ is a non-decreasing function of k and $\theta(n) = an > 0$, i. e., $Y_n > X_n$. Thus the following two cases are possible: either $Y_k \ge X_k$ for all $k = 1, \ldots, n$, or there exists an index $r \in \{1, \ldots, n-1\}$ such that $Y_r < X_r$ and $Y_k \ge X_k$ for all $k = r+1, \ldots, n$.

Case 1: *$Y_k \ge X_k$ for all $k = 1, \ldots, n$.* Then by the construction of σ,

$$\begin{aligned} C_{\max}(\sigma) &\le \frac{\sum_{k=1}^n C_{\max}(\sigma_k)}{n} \le \frac{\sum_{k=1}^n Y_k}{n} \\ &= \frac{ann + \sum_{k=1}^n b(n-k+1) + \sum_{k=1}^n l_k}{n} \\ &= an + b\frac{n+1}{2} + \frac{\sum_{k=1}^n l_k}{n} \\ &= \frac{a(n-1)}{2} + a\frac{n+1}{2} + b\frac{n+1}{2} + \frac{\sum_{k=1}^n l_k}{n} \\ \text{(by (7))} &\le \frac{3}{2} C^*_{\max}. \end{aligned} \tag{12}$$

Case 2: *there exists an index* $r \in \{1, \ldots, n-1\}$ *such that* $Y_r < X_r$ *and* $Y_k \geq X_k$ *for all* $k = r+1, \ldots, n$. Then by the construction of σ,

$$C_{\max}(\sigma) \leq \min\Big\{\frac{\sum_{k=r+1}^{n} C_{\max}(\sigma_k)}{n-r}, C_{\max}(\sigma_r)\Big\} \leq \min\{S_1, S_2\} \tag{13}$$

where

$$S_1 = \frac{\sum_{k=r+1}^{n} Y_k}{n-r} = \frac{an(n-r) + \sum_{k=r+1}^{n} b(n-k+1) + \sum_{k=r+1}^{n} l_k}{n-r}, \tag{14}$$

$$S_2 = X_r = a(n-r) + b + L. \tag{15}$$

Set $t = n - r$ and $\mu = \frac{t}{n}$. Then by (15) and (7),

$$S_2 \leq \frac{t}{n}an + b + L \leq (1+\mu)C^*_{\max}. \tag{16}$$

Moreover, by rearranging the expression (14) we obtain that

$$\begin{aligned} S_1 &= an + b(n+1) - b\frac{n+r+1}{2} + \frac{\sum_{k=r+1}^{n} l_k}{n-r} \\ &= a\frac{n+r-1}{2} + a\frac{t+1}{2} + b\frac{t+1}{2} + \frac{\sum_{k=n-t+1}^{n} l_k}{t} \\ &= R + \big(a\frac{n+1}{2} + b\frac{n+1}{2} + \beta\mu\big) + \frac{1}{2}\big(a\frac{t+1}{2} + b\frac{t+1}{2} + \beta\big) + \beta(\frac{1}{2} - \mu) \end{aligned}$$

where $\beta = \frac{1}{t}\sum_{k=n-t+1}^{n} l_k$ and

$$\begin{aligned} R &= a\frac{2n-t-1}{2} + a\frac{t+1}{4} + b\frac{t+1}{4} - a\frac{n+1}{2} - b\frac{n+1}{2} \\ &= a\frac{2n-t-3}{4} - b\frac{2n-t+1}{4}. \end{aligned}$$

By (2),

$$R \leq \frac{2n-t}{4}(a-b) = \frac{2n-t}{4n}anq = \frac{2-\mu}{4}anq \leq \frac{2-\mu}{4}qC^*_{\max}.$$

Furthermore, by Lemma 1 we have

$$a\frac{n+1}{2} + b\frac{n+1}{2} + \beta\mu = \frac{(a+b)(n+1)}{2} + \frac{\sum_{k=n-t+1}^{n} l_k}{n} \leq C^*_{\max}.$$

Now consider the instance I' of the problem formed by the subset of jobs $\{n-t+1, \ldots, n\}$ with the same parameters as in I. It is clear that the length of a shortest schedule in I' is at most $C^*_{\max}$ whereas by Lemma 1, it is bounded from below by

$$a\frac{t+1}{2} + b\frac{t+1}{2} + \beta = a\frac{(a+b)(t+1)}{2} + \frac{\sum_{k=n-t+1}^{n} l_k}{t}.$$

So we obtain that

$$a\frac{t+1}{2} + b\frac{t+1}{2} + \beta \leq C^*_{\max}.$$

Summing up, we arrive at the following inequality:

$$S_1 \leq \frac{2-\mu}{4} q C^*_{\max} + \frac{3}{2} C^*_{\max} + \beta(\frac{1}{2} - \mu).$$

Thus from (13), (16), and the last inequality we obtain

$$C_{\max}(\sigma) \leq \min\{(1+\mu)C^*_{\max}, \frac{2-\mu}{4} q C^*_{\max} + \frac{3}{2} C^*_{\max} + \beta(\frac{1}{2} - \mu)\}.$$

It follows that $C_{\max}(\sigma) \leq \frac{3}{2} C^*_{\max}$ if $\mu \leq \frac{1}{2}$ and

$$C_{\max}(\sigma) \leq \min\{1+\mu, \frac{2-\mu}{4} q + \frac{3}{2}\} C^*_{\max}$$

if $\mu > \frac{1}{2}$. Thus $C_{\max}(\sigma) \leq (1+\mu^*) C^*_{\max}$ where μ^* satisfies

$$\frac{2-\mu^*}{4} q + \frac{3}{2} = 1 + \mu^*,$$

which implies that $\mu^* = \frac{2q+2}{q+4}$. Together with (12) this implies (11), as required. □

Observe that

$$\min\{1 + \frac{2q+2}{q+4}, 2-q\} \leq (2-q')$$

where q' satisfies

$$q'^2 + 5q' - 2 = 0.$$

By resolving this quadratic equation, we get $q' = -\frac{5}{2} + \frac{1}{2}\sqrt{33}$ and so

$$\min\{1 + \frac{2q+2}{q+4}, 2-q\} \leq (2 + \frac{5}{2} - \frac{1}{2}\sqrt{33}) = \frac{9-\sqrt{33}}{2} < 1.628.$$

Thus we have the following

Corollary 1. *Algorithm* `SPECIAL` *solves* $F2 \mid l_j, p_{1j} = a, p_{2j} = b \mid C_{\max}$ *with an approximation factor of* $\frac{9-\sqrt{33}}{2} < 1.628$ *in* $O(n^2)$ *time. In the case when* $a = b$ *the approximation ratio of algorithm* `Special` *is bounded by* 1.5. □

References

1. Ageev A.A.: A 3/2-approximation for the proportionate two machine flow shop scheduling with minimum delays. In: Approximation and Online Algorithms, LNCS, vol.4927, pp. 55–66 (2008)
2. Dell'Amico M.: Shop problems with two machines and time lags. Operations Research 44, 777–787 (1996)
3. Dell'Amico M., Vaessens R.J.M.: Flow and open shop scheduling on two machines with transportation times and machine-independent processing times is NP-hard. In: Materiali di discussione 141, Dipartimento di Economia Politica, Università di Modena (1996)

4. Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326 (1979)
5. Johnson S.M.: Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–68 (1954)
6. Johnson S.M.: Discussion: Sequencing n jobs on two machines with arbitrary time lags. Management Science 5, 293–298 (1958)
7. Karuno Y., Nagamochi H.: A better approximation for the two-machine flow shop scheduling problem with time lags. In: Algorithms and Computation, LNCS, vol. 2906, pp. 309–318 (2003)
8. Kern W., Nawijn W.M.: Scheduling multi-operation jobs with time lags on a single machine, In: U. Faigle and C. Hoede (eds), Proceedings of the 2nd Twente Workshop on Graphs and Combinatorial Optimization, Enschede, 1991.
9. Lenstra J.K: Private communication, 1991.
10. Mitten L.G.: Sequencing n jobs on two machines with arbitrary time lags. Management Science 5, 293–298 (1958)
11. Strusevich V.A.: A heuristic for the two-machine open-shop scheduling with transportation times. Discrete Applied Mathematics 93, 287–304 (1999)
12. Yu W.: The two-machine shop problem with delays and the one-machine total tardiness problem. Ph.D. thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (1996)
13. Yu W., Hoogeveen H., Lenstra J.K.: Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. J. Sched. 7, 333–348 (2004)
14. Zhang X., van de Velde S.: Polynomial-time approximation schemes for scheduling problems with time lags. J. Sched. 13, 553–559 (2010)