Journal of Combinatorial Optimization (2022) 44:2195–2206
https://doi.org/10.1007/s10878-020-00593-1

A tight approximation algorithm for problem $P2 \rightarrow D|v = 1, c = 1|C_{max}$

Yinling Wang[1] · Yan Lan[2] · Xin Chen[3] · Xin Han[1] · Yong Piao[1]

Published online: 5 June 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract
This paper focuses on the scheduling problem on two parallel machines with delivery coordination. In particular, given a set of n jobs, we aim to find a schedule with a minimal makespan such that all jobs are first executed on two parallel machines then delivered at the destination with a transporter. This problem is known to be NP-hard Chang and Lee (Eur J Oper Res 158(2):470–487, 2004), cannot be solved with an approximation ratio strictly less than $3/2$ unless P=NP. We close the gap by proposing a polynomial time algorithm whose approximation ratio is $3/2+\varepsilon$ with $\varepsilon > 0$, improve the previous best ratio $14/9 + \epsilon$.

Keywords Scheduling with transportation · Bin packing · Approximation algorithm

1 Introduction

Over the past decades, supply chain management is one of the major concerns in operations research. Supply chain management is the series of activities required to plan,

✉ Xin Han
hanxin@dlut.edu.cn

Yinling Wang
yinling_wang@foxmail.com

Yan Lan
lanyan@dlnu.edu.cn

Xin Chen
chenxin.lut@hotmail.com

Yong Piao
piaoy@dlut.edu.cn

1 Dalian University of Technology, Dalian, China

2 School of Information and Communication Engineering, Dalian Minzu University, Dalian, China

3 School of Electronics and Information Engineering, Liaoning University of Technology, 121001 Jinzhou, China

control and execute a product's flow, starting from raw materials to the final product and finally the distribution to the final customer, in the most cost-effective way possible. This attracts the attention of researchers in designing methods for manufacturing products efficiently according to the standards of the factory.

In recent years, scheduling problems with transportation have received much attention. This is due to the working environment where factories have to produce many goods and need to deliver them to customers. It can be seen as a two-stage scheduling problem in which the first stage is job production, and the second stage is job delivery. Many approaches exist for single stage, and they can be solved efficiently. However, by applying these techniques independently, the resulting solution may not be good. Therefore, coordination is needed in order to improve the solution.

1.1 Problem definition

In this paper, we study the following scheduling problem. We are given a set of n jobs $J_1, J_2, \ldots, J_n$ that need to be processed on two parallel machines M_1, M_2. Each job J_i has a processing time p_i and size $s_i \in [0, 1]$. Once a job is completed, it can be transported to the destination. However, there is only a transporter in the system, and the transporter can only carry jobs with a total size of 1 and needs T_1 (resp. T_2) time-unit to arrive at the destination (resp. come back from the destination). To simplify notation, we have $T = T_1 + T_2$ which corresponds to the round trip duration of the transporter. Without loss of generality, we assume that the transporter is at the location of the machines at the beginning.

The goal is to schedule all jobs and deliver all of them to the destination with the minimum makespan, i.e., the shortest time such that all jobs are delivered to the destination and the transporter goes back to the location of the machines.

The three-field notation scheme was introduced by Graham et al. (1979), which is utilized to represent the scheduling problems. The problem in this paper can be denoted as $P2 \rightarrow D|v = 1, c = 1|C_{\max}$, which means that there are two identical parallel machines and one transporter. $C_{\max}$ is the objective function of the problem, and we aim to minimize the makespan.

1.2 Related works

There are also other scheduling problems with transportation, such as problems $TF2|v = 1, c|C_{\max}$ etc, where there is one transporter between two machines, any job is first processed on M_1, then delivered to M_2, finally processed on M_2, and each time the transporter can carry only c jobs. Kise (1991) proved problem $TF2|v = 1, c = 1|C_{\max}$ is NP-hard in an ordinary sense, In 2001 Hurink and Knust (2001) proved the problem $TF2|v = 1, c = 1|C_{\max}$ is strongly NP-hard. Lee and Chen (2001) further proved that the problem $TF2|v = 1, c \geq 3|C_{\max}$ is also strongly NP-hard and mentioned that the complexity of $TF2|v = 1, c = 2|C_{\max}$ is open. Recently Lan et.al proved that $TF2|v = 1, c = 2|C_{\max}$ is strongly NP-hard Lan et al. (2016). Chang and Lee (2004) were the first to consider the case in which each job has a different size for delivery, while Li et al. (2005) focused on the problem where

customers may have a different location. Woeginger (1994); Potts (1980); Hall and Shmoys (1992) studied the problem with a single machine and parallel machines; each job has its arrival times and transport times. Meanwhile, they assumed that the number of vehicles is unbounded such that a job can be delivered to the destination once they are processed. For these models, they provided the heuristics and worst-case analysis. More related works can be found in Potts and Kovalyov (2000); Potts and Van Wassenhove (1992); Webster and Baker (1995).

For the problem $P2 \rightarrow D|v = 1, c = 1|C_{\max}$, Chang and Lee (2004) first proposed this model and gave a polynomial-time algorithm with an approximation ratio of 2. Moreover, they proved that there is no approximation algorithm for the problem with an approximation ratio better than 3/2 unless P = NP. Then, Zhong et al. (2007) presented an improved algorithm with an approximation ratio of 5/3. Su et al. (2009) presented an improved algorithm for the problem and reduced the worst-case ratio to 8/5. Then Wang and Liu (2013) and Zhang et al. (2015) proposed $(14/9 + \varepsilon)$-approximation algorithms.

1.3 Our contributions

In this paper, we provide an algorithm with an approximation ratio of $3/2 + \varepsilon$, where ε is a positive real and arbitrarily close to 0. This matches the lower bound of the problem asymptotically. Observe that the previous approaches work for the case: the total size in the input is at least seven. For the other case, we give a new batching strategy: guessing the number of batches in an optimal solution, enumerating all the possible cases for large jobs, and appending small jobs into the batches by solving a linear programming with the guarantee that the total processing time in each batch is almost the same as the one in the optimal solution, transferring the fractional solution into integral solution with using at most one extra batch. Embedding the new batching strategy, we can prove that the approximation ratio is close 1.5 for the case: there are at most seven batches in an optimal solution.

2 Preliminary

In this section, we first introduce a bin packing problem then give some notations used later.

2.1 Bin packing problem

there are n items $x_1, x_2, \ldots, x_n$, each has size s_i in $(0, 1)$. There are also empty bins with a capacity of 1. The goal is to pack all the items into a minimum number of bins while each bin can pack any subset of items with total size at most 1. This problem is known to be NP-hard Johnson (2008).

An approximation algorithm for the classical one-dimensional bin packing problem called FFD (First Fit Decreasing) is provided in Johnson et al. (1974). The FFD algorithm first sorts the list of items into non-increasing order of their size then places

each item one by one with this order into the first bin where it will fit. It requires $\Theta(n \log n)$ time, where n is the number of items to be packed. The following lemma is from Dósa et al. (2013).

Lemma 1 *For any bin packing input L, the number of bins used by FFD is at most $11OPT(L)/9 + 6/9$, where $OPT(L)$ is the optimal value.*

2.2 Notations

Given a schedule, let C be its makespan, and C^* be the makespan of an optimal solution. Let $C(B_i)$ be the completion time of batch B_i in a given schedule. Let $C^*(B_i^*)$ be the completion time of batch B_i^* in an optimal schedule. Let the sum of the processing times of jobs in batch B_i be $p(B_i)$. Let P be the total processing times of all jobs. Let C_M^* be the completion time of jobs on machines by an optimal algorithm. Then we have
vspace*-6pt

$$C_M^* \geq \frac{P}{2}.$$

Let b^* be the number of batches in the optimal algorithm. Then we have

$$C^* \geq C_M^* + T, \quad C^* \geq C^*(B_1^*) + b^* \cdot T. \tag{1}$$

3 Framework of old algorithms

We first introduce a framework of old algorithms Wang and Liu (2013) and prove that it performs very well for the case where there are at least 8 batches in an optimal solution, i.e., its worst performance ratio is 1.5. In the framework, we first partition the input into batches, then schedule batches onto two machines and deliver them to the destination.

Algorithm 1 : Framework based on a batching strategy A

Input: a set of batches of jobs
Output: schedule batches on machines, then deliver them to the destination

1: Assign jobs to batches by a strategy A. Let the total number of resulting batches be b.
2: Let the sum of the processing times of jobs in batch B_k be $p(B_k)$, for $k = 1, 2 \cdots, b$. Re-index the batches such that $p(B_1) \leq p(B_2) \leq \cdots \leq p(B_b)$.
3: According to the order $\{B_1, B_2, \ldots, B_{b-1}\}$, assign the batches one by one to machines by LS rule(List Scheduling rule), where all the jobs in the same batch are assigned to the same machine. Within each batch, the jobs are sequenced in an arbitrary order.
4: We assign the jobs of the last batch B_b one by one by LS rule to two machines.
5: Dispatch each completed but undelivered batch whenever a vehicle becomes available. If multiple batches have been completed when a vehicle becomes available, dispatch the batch with the smallest index.

Given an algorithm A on input L, the makespan of A is denoted as $C_A(L)$. Then it is not difficult to get the following result:

$$C_A(L) = \max\{C(B_i) + (b + 1 - i)T | 1 \le i \le b\},$$

where $C(B_i)$ is the completion time of batch B_i and b is the number of batches. However by the following lemma from Chang and Lee (2004) Wang and Liu (2013) we only need to consider the following five cases, i.e., $i = 1, 2, 3, b - 1, b$.

Lemma 2 *For input L,* $C_A(L) = \max\{C(B_i) + (b + 1 - i)T | i = 1, 2, 3, b - 1, b\}$, *where T is the time from machines to the destination and back to machines.*

Lemma 3 Chang and Lee (2004) *For* $i = 1, 2$ *and* $b \ge 3 + i$, *if* $C_A(L) = C(B_i) + (b + 1 - i) \cdot T$, *we have* $p(B_i) \le 2T$.

Lemma 4 $C(B_b) \le \frac{3}{2}C^*_M$ *and* $C(B_{b-1}) \le C^*_M$ Wang and Liu (2013).

Proof For the sake of completeness, we give a detail proof here. In the above algorithm, we actually run LS algorithm on two machines, so by Graham et al. (1979), we have

$$C(B_b) \le \frac{3}{2}C^*_M.$$

It is not difficult to see that except for the last batch, batches $B_1, B_3, B_5, \ldots$. are assigned on one machine and $B_2, B_4, B_6, \ldots$ on the other machine. Since $p(B_1) \le p(B_2) \le \cdots \le p(B_b)$, we have

$$\begin{aligned} C(B_{b-1}) &= p(B_{b-1}) + p(B_{b-3}) + \cdots + p(B_{b-2k+1}) \\ &\le p(B_b) + p(B_{b-2}) + \cdots . + p(B_{b-2k+2}), \end{aligned}$$

where $1 \le k \le b/2$. Hence $C(B_{b-1}) \le C^*_M$. □

Lemma 5 *If* $b = 1$, *then* $C_A(L) \le \frac{3}{2}C^*$.

Proof When $b = 1$, we have $C_A(L) = C(B_1) + T$. By Lemma 4, we have $C(B_1) \le \frac{3}{2}C^*_M$. Then by (1), we have

$$\frac{C_A(L)}{C^*} \le \frac{C(B_1) + T}{C^*_M + T} \le \frac{3}{2}.$$

Hence we have this lemma. □

Using the techinques in Chang and Lee (2004) and Wang and Liu (2013), it is not difficult to get the following result.

Lemma 6 *In the above algorithm, if we run FFD algorithm for batching, then* $C \le \frac{3}{2}C^*$ *if* $b^* \ge 8$.

Proof By Lemma 2, at least one of the following equations holds.

- $C = C(B_b) + T$
- $C = C(B_{b-1}) + 2T$

- $C = C(B_{b-2}) + 3T$
- $C = C(B_2) + (b-1)T$
- $C = C(B_1) + b \cdot T$

Case 1: $C = C(B_b) + T$. By Lemma 4, since $C(B_b) \le \frac{3}{2}C_M^*$, we have

$$\frac{C(B_b)+T}{C^*} \le \frac{\frac{3}{2}C_M^* + T}{C_M^* + T} \le \frac{3}{2}$$

Case 2: $C = C(B_{b-1}) + 2T$.
By Lemma 4, we have $C(B_{b-1}) \le C_M^*$.
Thus we can obtain:

$$\frac{C(B_{b-1})+2T}{C^*} \le \frac{C(B_{b-1})+2T}{\max\{b^*T, C_M^*+T\}} \le \frac{T}{b^*T} + \frac{C_M^*+T}{C_M^*+T} < \frac{3}{2}$$

Case 3: $C = C(B_{b-2}) + 3T$.
By Lemma 4 and the above algorithm, $C(B_{b-2}) \le C(B_{b-1}) \le C^*$. Hence we have:

$$\frac{C(B_{b-2})+3T}{C^*} \le \frac{C_M^*+T}{C_M^*+T} + \frac{2T}{b^*T} \le 1 + \frac{2T}{b^*T} < \frac{3}{2},$$

since $b^* \ge 4$.

Case 4: $C = C(B_2) + (b-1) \cdot T$.
In this case we have $b \ge 5$, otherwise at least one of the three cases holds. By Lemme 3, we have $p(B_2) \le 2T$.
By Lemma 1, we can get:

$$\frac{C(B_2)+(b-1)T}{C^*} \le \frac{(b+1)T}{b^*T} \le \frac{\frac{11}{9}b^* + \frac{6}{9} + 1}{b^*} \le \frac{11}{9} + \frac{15}{9b^*} \le \frac{3}{2},$$

where $b^* \ge 6$.

Case 5: $C = C(B_1) + b \cdot T$.
In this case we have $b \ge 4$, otherwise at least one of Case 1, Case 2 and Case 3 holds. By Lemea 3, we have $C(B_1) = p(B_1) \le 2T$.
Then we can get:

$$\frac{C(B_1)+b\cdot T}{C^*} \le \frac{(b+2)T}{b^*T}.$$

- When $b^* = 8$, by Lemma 1, $b \le 10$, we have: $\frac{(b+2)T}{b^*T} \le \frac{12T}{8T} = \frac{3}{2}$.
- When $b^* = 9$, by Lemma 1, $b \le 11$, thus we have: $\frac{(b+2)T}{b^*T} \le \frac{13T}{9T} < \frac{3}{2}$.
- When $b^* \ge 10$, we have $\frac{C(B_1)+b\cdot T}{C^*} \le \frac{11b^*/9 + 6/9 + 2}{b^*} \le \frac{11}{9} + \frac{24}{9b^*} \le \frac{3}{2}$.

From the previous analysis, we can see that the lemma holds. □

4 ($\frac{3}{2}+\varepsilon$)-Algorithm for problem $P2 \rightarrow D$

We first give a new batching strategy to tackle the case where there are at most seven batches in an optimal solution, then embed the strategy into the old framework and prove that the approximation ratio is close to 1.5, finally combine the old algorithm and our algorithm, we prove that there is a $(1.5+\epsilon)$-approximation algorithm for the problem.

4.1 New batching strategy for small $b^* \leq 7$

In this subsection, we give a new batching strategy for the case where there are at most seven batches in an optimal solution. Normally we cannot know the exact assignment of jobs in an optimal solution. However if a configuration of batches in a feasible assignment is known, then we can guess a feasible assignment of jobs into batches such that the processing time in each batch is similar with each other and the number of batches is also similar with each other.

Assume that some partial information of a feasible assignment is known:

- the makespan value $\bar{C}$,
- the number of batches in the assignment $\bar{b}$,
- and a vector $(\bar{p}(B_1^{'}), \bar{p}(B_2^{'}), \ldots, \bar{p}(B_{\bar{b}}^{'}))$, where $\bar{p}(B_i^{'})$ is the total processing time in batch $B_i^{'}$.

We will assign all jobs into batches of $B_1, \ldots, B_b$, where $b \leq \bar{b}+1$ such that the total processing time $p(B_i)$ in B_i is at most as $\bar{p}(B_i^{'})$ and the total size $s(B_i)$ is at most one for all $1 \leq i \leq \bar{b}$ and $p(B_b) = \Theta(\frac{\varepsilon \bar{C}}{32})$ if $b = \bar{b}+1$, where $\varepsilon > 0$ is a given parameter. The details are given in the following algorithm.

Linear Programming for assigning jobs into $\bar{b}$ batches: after assigning large jobs into $\bar{b}$ batches, assume the total size of large jobs in B_i is S_i, the total processing time in B_i is P_i. Given n' small jobs, the following linear programming is used to assign small jobs to $\bar{b}$ batches.

$$\sum_{j=1}^{\bar{b}} x_{ij} = 1, \qquad i = 1, 2, \ldots, n^{'} \tag{2}$$

$$\sum_{i=1}^{n'} x_{ij} p_i + P_j \leq \bar{p}(B_j^{'}), \qquad j = 1, 2, \ldots, \bar{b} \tag{3}$$

$$\sum_{i=1}^{n'} x_{ij} s_i + S_j \leq 1, \qquad j = 1, 2, \ldots, \bar{b}$$

$$x_{ij} \geq 0 \qquad \forall i, j \geq 1 \tag{4}$$

Algorithm 2 : Assign all jobs into batches

Input: $\bar{C}, \bar{b}$ and $(\bar{p}(B_1^{'}), \bar{p}(B_2^{'}), \ldots, \bar{p}(B_{\bar{b}}^{'}))$.

Output: another feasible assignment such that the processing time in each batch is similar if possible.

1: Divide all jobs into large jobs and small jobs. Job j is *large* if its size $s \geq \frac{1}{16}$ or its processing time $p \geq \frac{\varepsilon \bar{C}}{32}$ for a given small ε and $1/\epsilon$ is integer; other it is called *small*.
2: Guess all possible assignments of large jobs into $\bar{b}$ batches, i.e., exhaust all the possible cases.
3: Check the feasibility for each case, i.e., the total size in each batch is at most 1 and the total processing time in batch B_i is at most $\bar{p}(B_i^{'})$.
4: Ignore all infeasible cases and keep all feasible cases. For each feasible case, we run the following steps to get a feasible assignment for small jobs.
5: Fractionally assign all small jobs into $\bar{b}$ batches by solving a linear programming which will be defined later.
6: If there is a basic feasible solution, then we stop and output an assignment as follows: for large jobs, we assign them into batches as we guess; for small jobs with the integral assignment, we assign them by the solution of linear programming; the left jobs with the fractional assignment are packed in an extra batch. We will prove that one batch is enough later.
7: Otherwise output there is no feasible solution.

Variable $x_{ij} = 1$ if job i is assigned into batch j.

The equations (2) show that each small job must be assigned on some machine. The equations (3) show that the total processing time of jobs assigned in batch j is bounded by $\bar{p}(B_j^{'})$ respectively. The equations (4) show that the total size of the jobs assigned in each batch is bounded by 1 respectively.

Lemma 7 *If $\bar{b} \leq 7$ there are at most $64/\epsilon + 112$ large jobs.*

Proof If there are more than 112 large jobs with size at least 1/16, then $\bar{b}$ would have been larger than 7. If there are more than $64/\epsilon$ large jobs with processing time at least $\epsilon \bar{C}/32$, then the makespan of an optimal solution would have been larger than $\bar{C}$. □

Lemma 8 *In a basic feasible solution of the above LP, there are at most $2\bar{b}$ small jobs such that $0 < x_{ij} < 1$ for some j.*

Proof In a basic feasible solution, let x be the number of small jobs such that $x_{ij} = 1$ and let y be the number of small jobs such that $0 < x_{ij} < 1$ for some j. It is not difficult to see that

$$x + y = n^{'}.$$

On the other side, the linear program contains at most $n^{'} + 2\bar{b}$ constraints and $\bar{b}n^{'}$ variables. So in a basic feasible solution, there are at most $n^{'} + 2\bar{b}$ no-zero variables, i.e.,

$$x + 2y \leq n^{'} + 2\bar{b}.$$

Hence we have $y \leq 2\bar{b}$. □

Lemma 9 *If there are some fractional jobs in a feasible solution of the linear programming, then an extra batch is enough and $p(B_b) \leq \frac{\varepsilon \bar{C}}{2}$.*

Proof By Lemma 8, we have at most $2\bar{b}$ small jobs with $0 < x_{ij} < 1$ for some j. Since each small job has size at most 1/16 and $\bar{b} \leq 7$, all such small jobs can be packed into one batch. Hence an extra bin for fractional small items is enough.

By the definition of small jobs and Lemma 8, $\bar{b} \leq 7$, we have

$$p(B_b) \leq 2\bar{b} \cdot \frac{\varepsilon \bar{C}}{32} < \frac{\varepsilon \bar{C}}{2}.$$

Hence we have this lemma. □

4.2 A new upper bound $1.5 + \epsilon$

Algorithm 3 : new framework for scheduling jobs for small b^*

Input: the number b^* of batches in an optimal solution.
Output: a feasible schedule with at most $(b^* + 1)$ batches and the makespan at most $(1.5 + O(\epsilon))C^*$.
1: Find an upper bound of C^* by running a 2-approximation algorithm and setting $\bar{C}$ as the value of the makespan.
2: Set $\bar{p}(B_i^{'}) = k \cdot \frac{\varepsilon \bar{C}}{32}$ for all $1 \leq k \leq 64/\epsilon$ and $1 \leq i \leq b^*$.
3: For each possible vector $\bar{p} = (\bar{p}(B_1^{'}), \bar{p}(B_2^{'}), \ldots)$, we call Algorithm $A_2(\bar{C}, b^*, \bar{p})$. If there is a feasible assignment by Algorithm A_2, then call Algorithm A_1 to schedule all the batches.
4: Output the assignment with the minimal makespan.

Lemma 10 *There is at least one feasible solution returned by the above algorithm.*

Proof It is not difficult to see that an optimal solution with b^* batches and $(p(B_1^*), p(B_2^*), \ldots, p(B_{b^*}^*))$ is one of feasible solutions of LP($\bar{C}, b^*, \bar{p}$), where $\bar{p}$ is vector and its i-th component value is $(k+1)\frac{\epsilon}{32}\bar{C}$, where $k \cdot \frac{\epsilon}{32}\bar{C} < p(B_i^*) \leq (k+1)\frac{\epsilon}{32}\bar{C}$. Since in our algorithm, we try all possible values for each component of $\bar{p}$, the optimal solution must be included in feasible solutions of LP(*,*,*) defined in the last subsection. □

Lemma 11 *The above algorithm is run in polynomial time if $b^* \leq 7$.*

Proof By Lemma 7, the number of large jobs is upper bounded by $O(1/\epsilon)$. Given a vector $\bar{p}$, the number of enumerating all possible cases for large jobs is at most $(b^*)^{O(1/\epsilon)} \leq 7^{O(1/\epsilon)}$. The number of all possible vectors $\bar{p}$ is at most $O((1/\epsilon)^{b^*}) \leq O(1/\epsilon^7)$. And it takes time $O(n^c)$ to solve a linear program for some constant c. Hence we have this lemma. □

Lemma 12 *If $2 \leq b^* \leq 7$, then the makespan by the above algorithm is at most $(1.5 + \epsilon)C^*$.*

Proof Given an optimal solution with batching vector $(p(B_1^*), p(B_2^*), \ldots, p(B_{b^*}^*))$, we can consider the linear programm LP($\bar{C}, b^*, \bar{p}$), where $\bar{C} \leq 2C^*$ and the i-th

component $\bar{p}_i$ is at least $p(B_i^*)$ and at most $p(B_i^*) + \epsilon\bar{C}/32$. Then we consider a feasible solution associated with the above LP($\bar{C}, b^*, \bar{p}$). Let C be the makespan generated by the feasible solution. By Lemma 2, at least one of the following equations holds.

- $C = C(B_b) + T$
- $C = C(B_{b-1}) + 2T$
- $C = C(B_1) + b \cdot T$
- $C = C(B_{b-2}) + 3T$
- $C = C(B_2) + (b-1) \cdot T$

Case 1: $C = C(B_b) + T$. We can prove $C = \frac{3}{2}C^*$ using the way similar with Case 1 in Lemma 6.

Case 2: $C = C(B_{b-1}) + 2T$. We can also prove $C = \frac{3}{2}C^*$ using the way similar with Case 2 in Lemma 6.

Case 3: $C = C(B_1) + b \cdot T$. In this case, we can assume $b \geq 3$, otherwise Case 1 or Case 2 will occur. Then we have

$$C(B_1) = p(B_1) \leq \frac{P}{3} \leq \frac{2C_M^*}{3} \leq \frac{2C^*}{3}$$

and $C^* \geq \frac{p(B_1^*)}{2} + b^* \cdot T$ since $C^*(B_1^*) \geq \frac{p(B_1^*)}{2}$. If $b = b^*$, it means that we don't use an extra batch in Algorithm A_2. Due to $\bar{C} \leq 2C^*$,

$$C(B_1) = p(B_1) \leq p(B_1^*) + \epsilon\bar{C}/32 < p(B_1^*) + \epsilon C^*,$$

due to $C^* \geq \frac{p(B_1^*)}{2} + b^* \cdot T$, we have

$$\frac{C(B_1) + b^*T}{C^*} \leq \frac{p(B_1)/2 - \epsilon C^* + b^*T}{\frac{p(B_1^*)}{2} + b^* \cdot T} + \frac{p(B_1)/2 + \epsilon C^*}{C^*} = 1 + \epsilon + 1/3.$$

Otherwise by Lemma 9, we have $b = b^* + 1$. By Lemma 9 and Algorithm A_1, $\bar{C} \leq 2C^*$, we have

$$C(B_1) = p(B_1) \leq \epsilon C^*.$$

Due to $b^* \geq 2$ and $C^* \geq b^*T$, we have

$$\frac{C(B_1) + (b^* + 1)T}{C^*} \leq \epsilon + 1.5.$$

Case 4: $C = C(B_{b-2}) + 3T$. In this case, we have $b \geq 4$, otherwise one of the above three cases will happen. If $b^* \geq 4$, then we can prove $C/C^* \leq 1.5$ using the way similar with Case 3 in Lemma 6. Otherwise $b^* = 3$. By Lemma 9, we have $b = 4$.

By the algorithm we use, $C(B_2) = p(B_2) \leq \frac{2C_M^*}{3}$. Hence we have:

$$\frac{C(B_2)+3T}{C^*} \leq \frac{2C_M^*/3+2T/3}{C_M^*+T} + \frac{7T/3}{b^*T} \leq 2/3+7/9 = \frac{3}{2},$$

since $b^* = 3$.

Case 5: $C = C(B_2) + (b-1)\cdot T$. In this case we have $b \geq 5$, otherwise one of Case 1, Case 2 and Case 4 will happen. By Lemma 3, we have $p(B_2) \leq 2T$.

By Lemma 9, we can get:

$$\frac{C(B_2)+(b-1)T}{C^*} \leq \frac{(b+1)T}{b^*T} \leq \frac{b^*+2}{b^*} \leq \frac{3}{2},$$

where $b^* \geq b-1 \geq 4$.

From the previous analysis, we can see that the lemma holds. □

Theorem 1 *There is a polynomial algorithm with makespan at most* $(1.5+\epsilon)C^*$, *where* $\epsilon > 0$ *is sufficient small real.*

Proof Consider the following algorithm.

Input: a set of n jobs and a parameter ϵ.
Output: a feasible schedule with makespan $(1.5+\epsilon)C^*$.
1: Run Algorithm $A_1(FFD)$ and get a schedule δ_1.
2: If the total size of jobs is at most 7, run Algorithm A_3 with b^* varying from 2 to 7, get the corresponding schedule δ_i for $2 \leq i \leq 7$. (If some schedule δ_i does not exist, its makespan is ∞)
3: Output schedule δ_i for $1 \leq i \leq 7$ with the minimal makespan.

To get schedule δ_1, it takes $O(n \log n)$ time steps. By Lemma 11, it takes polynomial time to get δ_i for $2 \leq i \leq 7$. So the above algorithm takes polynomial time.

By Lemmas 5, 6, and 12, we have the makespan by the above algorithm is at most $(1.5+\epsilon)C^*$. □

5 Concluding remarks

In this paper, we find that the approximation ratio of old algorithms is 1.5 for the case where there are at least eight batches, then we give a new algorithm with approximation ratio $1.5+\varepsilon$ for the other case. We know that there is no algorithm with approximation ratio less than 1.5 unless P=NP. Whether there is 1.5-approximation algorithm for the case where there are at most seven batches in an optimal solution leaves as an open question.

Acknowledgements This research was partially supported by NSFC(11971091,11701062), Liaoning Natural Science Foundation(2019-MS-062).

References

Chang Y-C, Lee C-Y (2004) Machine scheduling with job delivery coordination. Eur J Oper Res 158(2):470–487

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math 5(1):287–326

Dósa G, Li R, Han X, Tuza Z (2013) The tight bound of first fit decreasing bin-packing algorithm is FFD(I) $< = 11/9OPT(I) + 6/9$. Theo Comput Sci 510:13–61

Hall LA, Shmoys DB (1992) Jackson's rule for single-machine scheduling: making a good heuristic better. Math Oper Res 17(1):22–35

Hurink J, Knust S (2001) Makespan minimization for flow-shop problems with transportation times and a single robot. Discrete Appl Math 112(1–3):199–216

Johnson DS (2008) Bin packing. Springer, Boston, pp 1–99

Johnson DS, Demers AJ, Ullman JD, Garey MR, Graham RL (1974) Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J Comput 3(4):299–325

Kise H (1991) On an automated two-machine flowshop scheduling problem with infinite buffer. J Oper Res Soc Jpn 34:354–361

Lan Y, Han X, Zongtao W, Guo H, Chen X (2016) Complexity of problem TF2 $|v = 1, c = 2|c_{\max}$. Inf Process Lett 116(1):65–69

Lee C-Y, Chen Z-L (2001) Machine scheduling with transportation considerations. J Sched 4(1):3–24

Li C-L, Vairaktarakis G, Lee C-Y (2005) Machine scheduling with deliveries to multiple customer locations. Eur J Oper Res 164(1):39–51

Potts CN (1980) Analysis of a heuristic for one machine sequencing with release dates and delivery times. Oper Res 28(6):1436–1441

Potts CN, Kovalyov MY (2000) Scheduling with batching: a review. Eur J Oper Res 120(2):228–249

Potts CN, Van Wassenhove LN (1992) Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. J Oper Res Soc 43(5):395–406

Su CS, Pan CH, Hsu TS (2009) A new heuristic algorithm for the machine scheduling problem with job delivery coordination. Theor Comput Sci 410(27):2581–2591

Wang L, Liu Z (2013) An improved algorithm for scheduling two identical machines with batch delivery consideration. Oper Res Trans 17(1):38–43

Webster S, Baker KR (1995) Scheduling groups of jobs on a single machine. Oper Res 43(4):692–703

Woeginger GJ (1994) Heuristics for parallel machine scheduling with delivery times. Acta Inf 31(6):503–512

Zhang Y, Zheng Q, Ren J, Zhang L (2015) An improved algorithm for the machine scheduling problem with job delivery coordination. In *12th International symposium on operations research and its applications in engineering, technology and management (ISORA 2015)*, pp 167–172

Zhong W, Dósa G, Tan Z (2007) On the machine scheduling problem with job delivery coordination. Eur J Oper Res 182(3):1057–1072

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.