In the single machine scheduling problem with job delivery to minimize makespan, jobs are processed on a single machine and delivered by a capacitated vehicle to their respective customers. We first consider the special case with a single customer, that is, all jobs have the same transportation time. Chang and Lee (2004) proved that this case is strongly NP-hard. They also provided a heuristic with the worst-case performance ratio \( \frac{5}{3} \), and pointed out that no heuristic can have a worst-case performance ratio less than \( \frac{3}{2} \) unless \( P = NP \). In this paper, we provide a new heuristic which has the best possible worst-case performance ratio \( \frac{3}{2} \). We also consider an extended version in which the jobs have non-identical transportation times and the transportation time of a delivery batch is defined as the maximum transportation time of the jobs contained in it. We provide a heuristic with the worst-case performance ratio 2 for the extended version, and show that this bound is tight.

**Keywords**: Scheduling; job delivery; heuristic; worst-case performance ratio.

1. **Introduction and Problem Formulation**

As introduced by Chang and Lee (2004), the single machine scheduling problem with job delivery can be described as follows. There are \( n \) jobs \( J_1, \ldots, J_n \) with each job \( J_j \) having a processing time \( p_j \). These jobs are first processed by a single machine, and then the finished jobs are delivered to their respective customers by a capacitated vehicle. Let \( s_j \) be the size of \( J_j \), which represents the physical space \( J_j \) occupies when it is loaded in the vehicle. Each job \( J_j \) also has a transportation time \( t_j \), which is the time needed by the vehicle to delivery to its customer and return to the machine. Only one vehicle is employed to deliver all jobs. This vehicle has a capacity \( z \). That is, the total physical space of jobs loaded into the vehicle at a time is not allowed to exceed \( z \). We assume that \( s_j \leq z \) for each job \( J_j \).
A set of jobs delivered together in one shipment is defined as a delivery batch. The transportation time of a delivery batch is defined as the maximum transportation time of the jobs contained in it. This transportation model can be applied to the version that the jobs have different customers lying on a line; or the version that the jobs have a common customer but have different transportation times since some jobs are fragile or urgent.

Let \( P = \sum_{j=1}^{n} p_j \) be the sum of processing times of all jobs in the manufacturing system. In a schedule \( \pi \), according to Chang and Lee (2004), we have the following notation:

- \( b(\pi) \), the total number of delivery batches in schedule \( \pi \).
- For each \( r = 1, \ldots, b(\pi) \), \( B_r(\pi) \), the \( r \)th delivery batch in schedule \( \pi \). Batches delivered earlier have smaller indices.
- \( P_r(\pi) = \sum_{j \in B_r(\pi)} p_j \), the processing time of the \( r \)th delivery batch \( B_r(\pi) \).
- \( \delta_r(\pi) \), the departure time from the machine for the vehicle to deliver \( B_r(\pi) \).
- \( \rho_r(\pi) \), the ready time of \( B_r(\pi) \), which represents the latest completion time on the machine of the jobs assigned to \( B_r(\pi) \). Clearly, we have \( \delta_r(\pi) \geq \rho_r(\pi) \) in any feasible solution.

We simplify \( B_r(\pi), P_r(\pi), \delta_r(\pi), \) and \( \rho_r(\pi) \) to \( B_r, P_r, \delta_r, \) and \( \rho_r \), respectively, when \( \pi \) is known.

The makespan of schedule \( \pi \), denoted by \( C_{\text{max}}(\pi) \), is defined as the time when the vehicle finishes delivering the last batch to the customers and returns to the machine. We also simplify \( C_{\text{max}}(\pi) \) to \( C_{\text{max}} \) if no confusion.

According to the notation of Chang and Lee (2004), when there is only a customer, all jobs have the same transportation time \( T \). Then the scheduling problem is denoted by \( 1 \rightarrow D, k = 1 \mid v = 1, c = z \mid C_{\text{max}} \), where “1 \( \rightarrow D, k = 1 \)” is used to represent problems in which jobs are first processed on a single machine, and then delivered to a single customer; and “\( v = 1, c = z \)” means that only a vehicle of capacity \( z \) is employed to deliver all jobs. When the jobs have non-identical transportation times, the scheduling problem is denoted by \( 1 \rightarrow D \mid v = 1, c = z, t_j \mid C_{\text{max}} \).

The machine scheduling problems with job delivery have been widely studied over the last decade. Herrmann and Lee (1993), Yuan (1996), Chen (1996), Yang (2000), and Cheng et al. (1996) considered several scheduling problems with jobs being delivered in batches and each delivery batch having a delivery cost. Lee and Chen (2001) studied another coordination of production scheduling and transportation (subject to delivery time and vehicle capacity) to minimize makespan without considering delivery cost. This problem was extended by Chang and Lee (2004) by considering the situation where each job might occupy a different amount of physical space in a vehicle. They assumed that all jobs have the same transportation time and proved that this problem is strongly NP-hard. They also provided a heuristic with the worst-case performance ratio \( \frac{3}{2} \), and pointed out that no heuristic can have a worst-case performance ratio less than \( \frac{3}{2} \) unless \( P = NP \). He et al. (2005)
presented an improved heuristic with the worst-case performance ratio $\frac{53}{35}$. Recent development of this topic can be found in Chen and Vairaktarakis (2005), Hall and Potts (2005), Li et al. (2005), Pundoor and Chen (2005), and Wang and Lee (2005).

In this paper, when all jobs have the same transportation time $T$, we provide a new heuristic which has the best possible worst-case performance ratio $\frac{3}{2}$. We also consider an extended version in which the jobs have non-identical transportation times. We provide a heuristic with the worst-case performance ratio 2 for the extended version, and show that the bound is tight.

2. $1 \rightarrow D$, $k = 1 | v = 1, c = z | C_{\text{max}}$

Before proposing the heuristic, we recall the following useful lemmas obtained by Chang and Lee (2004).

**Lemma 2.1 (Chang and Lee, 2004).** There exists an optimal schedule for the problem $1 \rightarrow D$, $k = 1 | v = 1, c = z | C_{\text{max}}$ such that

1. jobs are processed on the machine without idle time;
2. jobs assigned to one batch are processed consecutively on the machine;
3. jobs assigned to one batch are processed on the machine in any order;
4. batches are delivered in non-decreasing order of $P_r$, $\forall r$.

**Lemma 2.2 (Chang and Lee, 2004).** For any solution satisfying the conditions in Lemma 2.1, if $\delta_r = \rho_r$ for some $r$ with $2 \leq r \leq b$, then $\delta_i = \rho_i$ for $i = r, r + 1, \ldots, b$, and $C_{\text{max}} = P + T$, where $b$ is the total number of batches in the solution.

**Lemma 2.3 (Chang and Lee, 2004).** For any solution satisfying the conditions in Lemma 2.1, if $C_{\text{max}} > P + T$, then $P_1 < T$ and $C_{\text{max}} = P_1 + bT$.

First, we provide an **NF-LP/S** (Next Fit-Longest Processing time/Size ratio) algorithm for solving the batch partition problem:

**NF-LP/S algorithm:**

1. Re-index the jobs such that $\frac{p_1}{s_1} \geq \frac{p_2}{s_2} \geq \cdots \geq \frac{p_n}{s_n}$.
2. Assign job $J_1$ to $B_1$.
3. When the $j$th job $J_j$ is considered, if it can be assigned to the last opened batch, that is, the total job size of the last opened batch after $J_j$ is added to it does not exceed $z$, then assign it to the last one; otherwise open a new batch and assign the job $J_j$ to it.

For bin-packing problem, Garey and Johnson (1995) proposed an algorithm called Modified First Fit Decreasing (**MFFD**) algorithm, whose running time is $O(n \log n)$. Yue and Zhang (1995) showed that **MFFD** algorithm can find a packing
using at most \( \frac{71}{60} b^*_L + 1 \) bins, where \( b^*_L \) is the optimal solution for the bin-packing problem. Note that the MFFD algorithm will be used in our discussion.

Now we propose the following heuristic \( H_1 \) for \( 1 \rightarrow D, k = 1 | v = 1, c = z | C_{\text{max}} \).

**Heuristic \( H_1 \).**

**Step 1:** If \( \sum_{j=1}^{n} s_j \leq z \), assign all jobs to one batch. If \( z < \sum_{j=1}^{n} s_j \leq 2z \), assign jobs to batches by the NF-LP/S algorithm. Otherwise, assign jobs to batches by the MFFD algorithm. Let \( b_{H_1} \) be the total number of resulting batches.

**Step 2:** Define \( P_r \) as the total processing time of the jobs in the \( r \)th batch, \( r = 1, 2, \ldots, b_{H_1} \). Re-index these batches such that \( P_1 \leq P_2 \leq \cdots \leq P_{b_{H_1}} \), and denote the \( r \)th batch by \( B_r \).

**Step 3:** For \( r \) from 1 to \( b_{H_1} \), assign jobs in \( B_r \) to the machine. Jobs within each batch can be sequenced in an arbitrary order.

**Step 4:** Dispatch each completed but undelivered batch whenever the vehicle becomes available. If there are more than one batch which have been completed when the vehicle becomes available, dispatch the batch with the smallest index.

Note that, for the batches obtained by the NF-LP/S algorithm, the total size of the jobs in two consecutive delivery batches is greater than \( z \). Hence, we have the following three observations about Heuristic \( H_1 \).

**Observation 1.** If \( \sum_{j=1}^{n} s_j \leq z \), then \( b_{H_1} = 1 \).

**Observation 2.** If \( z < \sum_{j=1}^{n} s_j \leq 2z \), then \( 2 \leq b_{H_1} \leq 3 \), and the delivery batches are determined by the NF-LP/S algorithm.

**Observation 3.** If \( b_{H_1} \geq 4 \), then \( \sum_{j=1}^{n} s_j > 2z \), and the delivery batches are determined by the MFFD algorithm.

Let \( C_{H_1} \) be the makespan obtained from Heuristic \( H_1 \) and let \( C^* \) be the optimal makespan. We also assume that \( \pi^* \) is an optimal solution for the scheduling problem with \( b^* \) batches. Let \( b^*_L \) be the number of batches if jobs are assigned to batches by an optimal bin-packing method. Clearly, we have \( b^*_L \leq b^* \). Suppose that \( P_1 \) is the processing time of the first batch obtained by Heuristic \( H_1 \), and \( P^*_1 \) is the processing time of the first batch in the optimal solution \( \pi^* \).

In order to analyse the bound of Heuristic \( H_1 \), we need the following lemmas, where Lemma 2.4 was obtained by Chang and Lee (2004) (see Case 1 in the proof of Theorem 1).

**Lemma 2.4 (Chang and Lee, 2004).** If \( C_{H_1} = P_1 + b_{H_1} T \) and \( C^* = P^*_1 + b^* T \), we have

\[
\frac{C_{H_1}}{C^*} \leq \frac{P^*_1 + (b^* - 1)T}{P_1 + b^* T} + b_{H_1} T.
\]
Lemma 2.5 If $3b^* \geq 2b^{H_1} + 2$, then $\frac{C^{H_1}}{C^*} \leq \frac{3}{2}$.

Proof. We distinguish the following two cases.

Case 1: $C^{H_1} = P + T$. Note that $C^* \geq P + T$. Then we have $\frac{C^{H_1}}{C^*} = 1$.

Case 2: $C^{H_1} > P + T$. By Lemma 2.3, $C^{H_1} = P_1 + b^{H_1}T$ and $P_1 < T$. Since $C^* \geq b^*T$, we have $\frac{C^{H_1}}{C^*} \leq \frac{T + b^{H_1}T}{b^*T} \leq \frac{3}{2}$. This completes the proof. □

Theorem 2.6 $\frac{C^{r_1}}{C^*} \leq \frac{3}{2}$.

Proof. If $C^{H_1} = P + T$, then we have $\frac{C^{H_1}}{C^*} = 1$. Hence, we suppose that $C^{H_1} > P + T$. By Lemma 2.3, $C^{H_1} = P_1 + b^{H_1}T$ and $P_1 < T$.

If $b^* = 1$, then $\sum_{j=1}^{n} s_j \leq z$ and the only possibility is $b^{H_1} = 1$. Hence, we have $\frac{C^{r_1}}{C^*} = 1$.

If $b^* = 2$, then $\sum_{j=1}^{n} s_j \leq 2z$. By Observation 1 and Observation 2, we have $b^{H_1} \leq 3$. When $b^{H_1} \leq 2$, by Lemma 2.5, we have $\frac{C^{r_1}}{C^*} \leq \frac{3}{2}$. When $b^{H_1} = 3$, we assume that $P_1'$ and $P_2'$ are the processing times of the first batch $B_1'$ and the second batch $B_2'$ obtained by the NF-LP/S algorithm, respectively. Denote by $P^*_2$ the processing time of the second batch $B_2'$ in the optimal schedule $\pi^*$. Let $r = \min\{j \in \{1, 2, \ldots, n\} : \sum_{i=1}^{j} s_i > z\}$. Then $B_1' = \{J_1, \ldots, J_{r-1}\}$, $J_r \in B_2'$.

Recall that $\frac{p_{s_1}}{s_1} \geq \frac{p_{s_2}}{s_2} \geq \cdots \geq \frac{p_{s_n}}{s_n}$. Then we have

$$P^*_2 = \sum_{J_r \in B_2' \cap \{J_1, \ldots, J_r\}} p_j + \sum_{J_j \in B_2' \setminus \{J_1, \ldots, J_r\}} \frac{p_j}{s_j} s_j$$

$$\leq \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} p_j + \sum_{J_j \in B_2' \setminus \{J_1, \ldots, J_r\}} \frac{p_j}{s_r} s_j$$

$$\leq \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} p_j + \frac{p_r}{s_r} \left( z - \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} s_j \right)$$

$$\leq \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} p_j + \frac{p_r}{s_r} \left( \sum_{1 \leq j \leq r} s_j - \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} s_j \right)$$

$$= \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} p_j + \frac{p_r}{s_r} \sum_{J_j \in \{J_1, \ldots, J_r\} \setminus B_2'} s_j$$

$$\leq \sum_{J_j \in B_2' \cap \{J_1, \ldots, J_r\}} p_j + \sum_{J_j \in \{J_1, \ldots, J_r\} \setminus B_2'} \frac{p_j}{s_j} s_j$$

$$= \sum_{1 \leq j \leq r} p_j$$

$$\leq P_1' + P_2'.$
Since $P - P'_1 - P'_2$ is the processing time of the third batch obtained by the NF-LP/S algorithm, we have $P_1 \leq P - P'_1 - P'_2 \leq P - P'_2 = P'$. Note that $C^* \geq P'_1 + 2T$. Then we have $\frac{C^*}{C'} \leq \frac{P'_1 + 3T}{P'_1 + 2T} = \frac{3}{2}$.

If $b^* = 3$ and $b^{H_1} \leq 3$, by Lemma 2.5, we have $\frac{C^{H_1}}{C'} \leq \frac{3}{2}$.

If $b^* = 3$ and $b^{H_1} = 4$, we distinguish the following two cases.

Case 1: $C^* = P'_1 + 3T$. By Lemma 2.4, we have

$$\frac{C^{H_1}}{C^*} \leq \frac{P'_1 + 2T + 4T}{P'_1 + 3T} = \frac{P'_1 + 18T}{4P'_1 + 12T} \leq \frac{3}{2}$$

Case 2: $C^* = P + T$.

(a) If $P_1 \leq \frac{3P - 5T}{2}$, we have $\frac{C^{H_1}}{C'} \leq \frac{3P - 5T}{2P + T} = \frac{3}{2}$.

(b) If $P_1 > \frac{3P - 5T}{2}$, then $P \geq 4P_1 > 6P - 10T$, and so $P < 2T$. But, since $b^* = 3$ and $C^* = P + T$, we have $P + T \geq 3T$, and so $P \geq 2T$, a contradiction. Hence, this case does not occur.

The case "$b^* = 3$ and $b^{H_1} \geq 5$" does not occur, since $b^{H_1} \leq \frac{71}{60}b^*_L + 1 \leq \frac{51}{2T}$.

Now suppose that $b^* \geq 4$. If $b^{H_1} \leq 5$, then by Lemma 2.5, we have $\frac{C^{H_1}}{C'} \leq \frac{3}{2}$. If $b^{H_1} \geq 6$, by Observation 3, we have $\frac{71}{60}b^*_L + 1 \geq b^{H_1} \geq 6$. Hence, we have $b^* \geq 5$, and so, $3b^* = 2 \times \frac{71}{60}b^* + \frac{38}{60}b^* > 2 \times \frac{71}{60}b^* + 3 = 2\left(\frac{71}{60}b^* + 1\right) + 1 \geq 2b^{H_1} + 1$. Since $b^*$ is an integer, we have $3b^* \geq 2b^{H_1} + 2$. By Lemma 2.5, we have $\frac{C^{H_1}}{C'} \leq \frac{3}{2}$. This completes the proof.

**Remark.** Chang and Lee (2004) pointed out that no heuristic can have a worst-case performance ratio less than $\frac{3}{2}$ unless $P = \text{NP}$. Thus, Heuristic $H_1$ has the best possible bound.

### 3. 1 → $D[v = 1, c = z, t_j]C_{\text{max}}$

Suppose that each job $J_j$ has a transportation time $t_j$ and a size $s_j$. First, we consider two small instances, which show that a simple sequence such as $\text{SPT}$, $\text{LPT}$, or in the decreasing/increasing order of $p_j/s_j$ value will lead to a worst-case performance ratio no less than 3.

If jobs are processed in the $\text{SPT}$ order or in the increasing order of $p_j/s_j$ value, we consider the following instance: There are 6 jobs with $(p_1, s_1, t_1) = (0, 3, 10), (p_2, s_2, t_2) = (0, 3, \epsilon), (p_3, s_3, t_3) = (\epsilon, 2, 10), (p_4, s_4, t_4) = (2\epsilon, 2, \epsilon), (p_5, s_5, t_5) = (3\epsilon, 2, \epsilon), (p_6, s_6, t_6) = (4\epsilon, 2, 10)$ and $z = 7$. Clearly, we have $p_1 \leq p_2 \leq \cdots \leq p_6$ and $p_1/s_1 \leq p_2/s_2 \leq \cdots \leq p_6/s_6$. Since early processed jobs should be delivered no later than those processed later, $J_1, J_3$ and $J_6$ with transportation time 10 must be contained in the distinct delivery batch. Thus, the resulted makespan $C^H$ is at least 30. However, the optimal solution would have only two batches with $B_1 = \{J_1, J_3, J_6\}$ and $B_2 = \{J_2, J_4, J_5\}$. The optimal value is given by $C^* = 10 + 6\epsilon$. Therefore, $\frac{C^H}{C'} \geq \frac{30}{10 + 6\epsilon} \rightarrow 3$ when $\epsilon \rightarrow 0$. 

If jobs are processed in LPT or in the decreasing order of \( \frac{p_j}{s_j} \) value, we consider the following instance: There are 6 jobs with 
\((p_1, s_1, t_1) = (9\epsilon, 3, 10), (p_2, s_2, t_2) = (7\epsilon, 3, \epsilon), (p_3, s_3, t_3) = (4\epsilon, 2, 10), (p_4, s_4, t_4) = (3\epsilon, 2, \epsilon), (p_5, s_5, t_5) = (2\epsilon, 2, \epsilon), (p_6, s_6, t_6) = (\epsilon, 2, 10) \) and \( z = 7 \). Similar to the above instance, we have \( C_H \geq 30 \). However, the optimal solution would have only two batches with \( B_1 = \{J_1, J_3, J_6\} \) and \( B_2 = \{J_2, J_4, J_5\} \). The optimal value is given by \( C^* = 10 + 15\epsilon \). Therefore, \( \frac{C^*}{C_H} \geq \frac{30}{10 + 15\epsilon} \to 3 \) when \( \epsilon \to 0 \).

In order to obtain a better heuristic, we first assume that each job can be split in size and each split part of a job has the same transportation time with the original job. That is, if job \( J_j \) is split into two parts \( J_j^{(1)} \) and \( J_j^{(2)} \), then we have \( t_j^{(1)} = t_j^{(2)} = t_j \) and \( s_j^{(1)} + s_j^{(2)} = s_j \). A job is called a split job if it is split in size. Notice that “split job” refers to an original job which is split, and not to the two parts that are obtained from splitting job. Here “split” is just a special trick applied in the partition of delivery batches. When the delivery batches are formed, we return to the original problem again.

We now describe the full-batch-longest-transportation-time (FBLTT) rule for the version that all jobs can be split in size.

**FBLTT rule.**

Step 1: Re-index the jobs such that \( t_1 \geq t_2 \geq \cdots \geq t_n \).

Step 2: Open a batch with the first remaining job, and then fill the present batch by the jobs one by one from the head of the unbatched job list. When meeting a job such that the last opened batch has not enough room for it, place a part of the job into the batch such that the present batch is completely full and put the remaining part of the job at the head of the remaining unbatched job list.

Step 3: Repeat Step 2 until the job list is empty.

We use \( T_{opt} \) to denote the sum of the transportation times of the batches obtained by FBLTT rule. Let \( C^* \) be the optimal makespan of the considered scheduling problem. Then we have \( C^* \geq \max\{P, T_{opt}\} \), where \( P = \sum_{j=1}^{n} p_j \) is the sum of the processing times of all jobs.

Next, we propose a No-Split rule for the version that no job can be split in size.

**No-Split rule.**

Step 1: Assign jobs to batches by FBLTT rule. The resulting batches are denoted by \( B_1', B_2', \ldots, B_r' \), and the split jobs are denoted by \( J_1', J_2', \ldots, J_s' \). Suppose that the second part of the split job \( J_j', 1 \leq j \leq s \), is included in the batch \( B_{i_j}' \), \( 2 \leq i_j \leq r \).

Step 2: Move out all split jobs and open a new batch for each of them. The resulting batches are denoted by \( B_1, B_2, \ldots, B_{r+s} \), where \( B_j, 1 \leq j \leq r \), is the remaining batch by removing the split jobs from the batch \( B_j' \) and \( B_{r+j} = \{J_j'\}, 1 \leq j \leq s \), are the new opened batches.
Lemma 3.1  Let \( t_{B_j} \) be the transportation time of batch \( B_j \), \( 1 \leq j \leq r + s \). Write \( t_{\text{max}} = \max\{t_{B_j} : 1 \leq j \leq r + s\} \). Then we have \( t_{\text{max}} + \sum_{j=1}^{r+s} t_{B_j} \leq 2C^* \).

Proof. By the implementation of the FBLTT rule and the No-Split rule, we have \( t_{\text{max}} = t_{B'_1} \), \( t_{B_j} \leq t_{B'_j} \) for \( 1 \leq j \leq r \) and \( t_{B_{r+j}} = t_{B'_{r+j}} \) for \( 1 \leq j \leq s \). Then we have

\[
t_{\text{max}} + \sum_{j=1}^{r+s} t_{B_j} \leq t_{B'_1} + \sum_{j=1}^{r} t_{B'_j} + \sum_{j=1}^{s} t_{B'_{r+j}} \leq 2 \sum_{j=1}^{r} t_{B'_j} = 2T_{\text{opt}} \leq 2C^*.
\]

Now we provide a heuristic for the scheduling problem under consideration.

Heuristic \( H_2 \).

Step 1: Assign jobs to batches by the No-Split rule. Let \( b^{H_2} \) be the total number of resulting batches.

Step 2: Calculate the processing time \( P_j \) of \( B_j \), \( j = 1, 2, \ldots, b^{H_2} \).

Step 3: Apply Johnson’s rule (Johnson, 1954) to determine the sequence of batches:
Let \( B_1 = \{B_j : P_j < t_{B_j}\} \) and \( B_2 = \{B_j : P_j \geq t_{B_j}\} \). Sort the batches in \( B_1 \) in a non-decreasing order of their processing times and then sort the batches in \( B_2 \) in a non-increasing order of their transportation times. Re-index the batches based on the obtained sequence.

Step 4: For \( j \) from 1 to \( b^{H_2} \), assign jobs in \( B_j \) to the machine. Jobs within each batch can be sequenced in an arbitrary order.

Step 5: Dispatch each completed but undelivered batch whenever the vehicle becomes available. If more than one batch have been completed when the vehicle becomes available, dispatch the batch with the smallest index.

Let \( C^{H_2} \) be the makespan obtained from Heuristic \( H_2 \).

Theorem 3.2  \( C^{H_2} / C^* \leq 2 \).

Proof. Suppose that \( B_q \) is the last batch with \( \delta_q = \rho_q \). Then we have \( \delta_j > \rho_j \) for \( q + 1 \leq j \leq b \). Hence, \( C^{H_2} = \sum_{j=1}^{q} P_j + \sum_{j=q}^{b} t_{B_j} \). We distinguish the following two cases.

Case 1: \( B_q \in B_1 \). By Lemma 3.1, we have

\[
C^{H_2} = \sum_{j=1}^{q} P_j + \sum_{j=q}^{b} t_{B_j} \leq \sum_{j=1}^{q} t_{B_j} + \sum_{j=q}^{b} t_{B_j} \leq t_{\text{max}} + \sum_{j=1}^{b} t_{B_j} \leq 2C^*.
\]
Case 2: $B_q \in B_2$. Then

$$C^{H_2} = \sum_{j=1}^{q} P_j + \sum_{j=q}^{b} t_{B_j} \leq \sum_{j=1}^{q} P_j + \sum_{j=q}^{b} P_j \leq P + P_q \leq 2C^*.$$ 

In order to show that the bound is tight, we consider an instance with 6 jobs below: $z = 7$, $(p_1, s_1, t_1) = (\epsilon, 3, 1), (p_2, s_2, t_2) = (1 - 2\epsilon, 3, 2\epsilon), (p_3, s_3, t_3) = (2\epsilon, 2, 2\epsilon), (p_4, s_4, t_4) = (p_5, s_5, t_5) = (0, 2, \epsilon), (p_6, s_6, t_6) = (2\epsilon, 2, \epsilon)$, where $\epsilon$ is a sufficiently small positive number. By $H_2$, we have $B_1 = \{J_1, J_2\}, B_2 = \{J_3\}, B_3 = \{J_4, J_5, J_6\}$ and $C^{H_2} = 2 + 2\epsilon$. However, the optimal solution would have only two batches $B_1 = \{J_1, J_3, J_4\}$ and $B_2 = \{J_2, J_5, J_6\}$ with $C^* = 1 + 5\epsilon$. Therefore, $\frac{C^{H_2}}{C^*} = \frac{2 + 2\epsilon}{1 + 5\epsilon} \to 2$ when $\epsilon \to 0$. 

Acknowledgments

This research was supported by NSFC (10671183) and NSFC-RGC (70731160633). The authors are grateful to the anonymous referees for the helpful comments.

References


Lingfa Lu is a PhD student in the Department of Mathematics, Zhengzhou University, Zhengzhou, Henan, People’s Republic of China. His research interests are in combinatorial optimization and scheduling theory. His papers have been published in *European Journal of Operational Research*, *OR Transaction*, and other journals.

Jinjiang Yuan is a Professor in the Department of Mathematics, Zhengzhou University, Zhengzhou, Henan, People’s Republic of China. His principal research interests are in graph theory, combinatorial optimization and scheduling theory. He has published a number of papers in *European Journal of Operational Research*, *Journal of Scheduling*, *Operations Research Letters*, *Theoretical Computer Science*, and other journals.