Abstract

In single machine scheduling with release dates and job delivery, jobs are processed on a single machine and then delivered by a capacitated vehicle to a single customer. Only one vehicle is employed to deliver these jobs. The vehicle can deliver at most \( c \) jobs at a shipment. The delivery completion time of a job is defined as the time at which the delivery batch containing the job is delivered to the customer and the vehicle returns to the machine. The objective is to minimize the makespan, i.e., the maximum delivery completion time of the jobs. When preemption is allowed to all jobs, we give a polynomial-time algorithm for this problem. When preemption is not allowed, we show that this problem is strongly NP-hard for each fixed \( c \geq 1 \). We also provide a \( \frac{5}{3} \)-approximation algorithm for this problem, and the bound is tight.

© 2008 Published by Elsevier B.V.

Keywords: Scheduling; Job delivery; Approximation algorithm; Approximation ratio

1. Introduction and problem formulation

Single machine scheduling with release dates and job delivery to minimize makespan can be described as follows. There are \( n \) jobs \( J_1, \ldots, J_n \) to be first processed by a single machine and then delivered by a capacitated vehicle to a single customer. Each job \( J_j \) has a processing time \( p_j \) and a release date \( r_j \). Only one vehicle is employed to deliver these jobs. The vehicle can delivery at most \( c \) jobs at a shipment. The set of all jobs delivered together in one shipment forms a delivery batch. The round-trip transportation time between the machine and customer is a constant \( T \). The delivery completion time of \( J_j \) is defined as the time at which the delivery batch containing \( J_j \) is delivered to the customer and the vehicle returns to the machine. The objective is to minimize the makespan, i.e., the maximum delivery completion time of the jobs. By using the general notation for a schedule problem, introduced by Graham et al. [6], this problem is denoted by \( 1 \rightarrow D|r_j, c \geq 1|C_{\text{max}} \). If preemption is allowed to all jobs, the corresponding scheduling problem is denoted by \( 1 \rightarrow D|r_j, \text{pmtn}, c \geq 1|C_{\text{max}} \).

---

Research supported by NSFC (10671183) and NFSC-RGC (70731160633).

* Corresponding author. Tel.: +86 371 67767835.
E-mail address: yuanjj@zzu.edu.cn (J. Yuan).

0304-3975/ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.tcs.2007.11.008
The machine scheduling problem with job delivery has been widely discussed in manufacturing research over the last decade. The earliest scheduling paper with job delivery is probably the one by Maggu and Das [10]. They considered a two-machine flow shop problem to minimize the makespan. The jobs completed on the first machine need to be delivered to the second machine. Herrmann and Lee [8], Yuan [15], Chen [2], Yang [14] and Cheng et al. [4] have considered several batch scheduling problem with due date related measures; each delivery batch occurs a delivery cost. Lee and Chen [9] considered another coordination of production scheduling and transportation (subject to delivery time and vehicle capacity) to minimize the makespan without considering delivery cost. This problem has been extended by Chang and Lee [1] by considering the situation where each job might occupy a different amount of physical space in a vehicle. Zhong, Dosa and Tan [16] present some improved approximation results for the problems considered by Chang and Lee [1]. Wang and Cheng [13] introduced the machine availability constraint into Lee and Chen’s model. That is, the machine is available only in some time intervals. Recent development of this topic can also be find in Chen and Vairaktarakis [3], Hall and Potts [7], Pundoor and Chen [11], and Wang and Lee [12].

In this paper, we consider the single machine scheduling problem with release dates and job delivery to minimize the makespan. When preemption is not allowed, we give a polynomial-time algorithm for this problem. When preemption is not allowed, we show that this problem is strongly NP-hard for each fixed $c \geq 1$. We also provide a $\frac{5}{3}$-approximation algorithm for this problem, and the bound is tight.

2. Notations and preliminaries

For a job $J_j$, its processing time and release date are denoted by $p_j$ and $r_j$, respectively. $T$ is used to denote the round-trip transportation time between the machine and customer. For a job set $B$, we define $\pi(B) = \sum_{j \in B} p_j$, and call it the processing time of $B$. Let $C^*(I)$ and $C^H(I)$ be the makespans for an instance $I$ given by an optimum algorithm and an approximation algorithm $H$, respectively. If $C^H(I) \leq rC^*(I)$ holds for each instance $I$, we say that $H$ is $r$-approximate for this problem. The minimum value of $r$ is defined as the approximation ratio of $H$. Unless ambiguity would result, we simplify $C^*(I)$ and $C^H(I)$ by $C^*$ and $C^H$, respectively.

Let $\pi$ be a feasible schedule for the scheduling problem. In the schedule $\pi$, we define the following notation:

- $S_j$, the starting time of $J_j$ in $\pi$.
- $\rho_j$, the ready time of $J_j$, which represents the processing completion time of $J_j$ on the machine.
- $B_k$, the $k$-th delivery batch in $\pi$. Batches delivered earlier have smaller indices.
- $\delta(B_k)$, the departure time from the machine for the vehicle to deliver $B_k$. Note that $\delta(B_k) \geq \rho(B_k)$ in any feasible solution.
- $C_j$, the delivery completion time of $J_j$, which is the time at which the delivery batch containing $J_j$ is delivered to the customer and the vehicle returns to the machine.

Since $T$ is the round-trip transportation time between the machine and customer, for every delivery batch $B$ and every job $J_j \in B$ we have $\rho_j \leq \delta(B)$ and $C_j = \delta(B) + T$. Furthermore, for every job $J_j$, we must have $r_j \leq S_j$ and $\rho_j = S_j + p_j$.

Assume that $\pi^*$ is an optimal schedule for the scheduling problem. If a notation $a$ is defined for $\pi$ as above, then the responding notation for $\pi^*$ is denoted by $a^*$. That is, we use the notations $S_j^*, \rho_j^*, B_k^*, \rho^*(B_k^*), \delta^*(B_k^*), C_j^*$ for the optimal schedule $\pi^*$.

In the optimal schedule $\pi^*$, the last delivery batch can be delivered only if all jobs have completed their processing on the machine. Furthermore, since we have $n$ jobs and the capacity of the vehicle is $c$, there are at least $\lceil \frac{n}{c} \rceil$ delivery batches in $\pi^*$. Hence, we have the following proposition.

**Proposition 2.1.** For the problem $1 \rightarrow D|r_j, c \geq 1|C_{\text{max}}$, we have $C^* \geq \max\{P + T, \lceil \frac{n}{c} \rceil T\}$, where $P = \sum_{j=1}^{n} p_j$ is total sum of the processing times of all jobs.
3. Scheduling with preemption

We show in this section that the problem $1 \rightarrow D|r_j, \text{pmtn}, c \geq 1|C_{\text{max}}$ can be solved in polynomial time. The following proposition is critical.

**Proposition 3.1.** For the problem $1 \rightarrow D|r_j, \text{pmtn}, c \geq 1|C_{\text{max}}$, there exists an optimal schedule $\pi$ with the following properties.

1. All jobs are processed on the machine by the SRPT-rule. That is, at any time $t$, the job with the smallest remaining processing time is scheduled.
2. A job with an earlier ready time is delivered no later than that with a later ready time.
3. Each delivery batch in $\pi$, apart from the first delivery batch, contains exactly $c$ jobs.

**Proof.** (1) Let $\pi$ be an optimal schedule for the problem $1 \rightarrow D|r_j, \text{pmtn}, c \geq 1|C_{\text{max}}$. If possible, let $t$ be the minimum time such that $\pi$ does not coincide with the SRPT-rule. Then, there are two jobs $J_i$ and $J_j$ such that both $J_i$ and $J_j$ are available at time $t$; the remaining processing time of $J_j$ is less than the remaining processing time of $J_i$, but a part of $J_j$ is processing in a time slot starting at $t$. Let $S$ be the interval set consisting of all time slots starting after or at $t$ in which $J_i$ and $J_j$ are processing. Define a new schedule $\pi'$ for $1 \rightarrow D|r_j, \text{pmtn}, c \geq 1|C_{\text{max}}$ from $\pi$ by rescheduling the remaining parts of $J_i$ and $J_j$ in such a way that, in the time slots in $S$, the remaining parts of $J_j$ are processing before the remaining parts of $J_i$. One can easily see that $\rho_i(\pi') < \min(\rho_i(\pi), \rho_j(\pi))$, $\rho_j(\pi') = \max(\rho_i(\pi), \rho_j(\pi))$ and $\rho_k(\pi') = \rho_k(\pi)$ for all $k \neq i, j$. An early finished job is delivered no later than a later finished job, and the number of jobs in each delivery batch is the same as that in $\pi$. Then, we obtain a feasible schedule $\pi'$ with the makespan of at most $C^*$. A finite number of repetitions of this procedure yields an optimal schedule of the required form.

(2) Can be proved by a pair-wise interchange in two delivery batches.

(3) If a delivery batch, with the exception of the first delivery batch, contains less than $c$ jobs, we can always fill the delivery batch with more jobs from earlier delivery batches without increasing the objective value. □

By Proposition 3.1, it is easy to see that the following algorithm $H_1$ can obtain an optimal schedule for $1 \rightarrow D|r_j, \text{pmtn}, c \geq 1|C_{\text{max}}$ in $O(n)$ time.

**Polynomial-time algorithm $H_1$**

**Step 1:** All jobs are processed on the machine by the SRPT-rule. That is, at any time $t$, the job with the smallest remaining processing time is scheduled.

**Step 2:** Assign all jobs into delivery batches such that

1. A job with an earlier ready time is delivered no later than that with a later ready time.
2. Each delivery batch in $\pi$, except the first delivery batch, contains exactly $c$ jobs.

**Step 3:** Whenever the vehicle and a delivery batch are available, transport the delivery batch with the lowest ready time.

4. Scheduling without preemption

4.1. NP-hardness proof

We show in this subsection that the problem $1 \rightarrow D|c \geq 1, r_j|C_{\text{max}}$ is strongly NP-hard for each fixed $c \geq 1$.

**Theorem 4.1.1.** The problem $1 \rightarrow D|c \geq 1, r_j|C_{\text{max}}$ is strongly NP-hard for each fixed positive integer $c$.

**Proof.** The decision version of the problem is clearly in NP. We use the strongly NP-complete 3-Partition problem (Garey and Johnson [5]) for the reduction.

3-Partition problem: Given a set of $3t$ integers $a_1, a_2, \ldots, a_{3t}$ such that $\sum_{i=1}^{3t} a_i = tB$ and $\frac{B}{3} < a_i < \frac{2B}{3}$ for $1 \leq i \leq 3t$, is there a partition of the $a_i$’s into $t$ groups of 3, each summing exactly to $B$?

For a given instance of the 3-Partition problem, we construct an instance of the decision version of the problem $1 \rightarrow D|c \geq 1, r_j|C_{\text{max}}$ as follows.

- We have $n = (4t + 1)c$ jobs of three types.
We distinguish the following two cases.

**Case 1.** $a_{\pi(3k-2)} + a_{\pi(3k-1)} + a_{\pi(3k)} < B$. In this case, the processing completion time of $J_{\pi(3k)}$ is $\rho_{\pi(3k)} = 4B(k - 1) + p_{\pi(3k-2)} + p_{\pi(3k-1)} + p_{\pi(3k)} < 4kB$ and the processing completion time of $J_{\pi(3k+1)}$ is $\rho_{\pi(3k+1)} = 4B(k - 1) + p_{\pi(3k-2)} + p_{\pi(3k-1)} + p_{\pi(3k)} + p_{\pi(3k+1)} > 4kB + B$, where the last inequality follows from the fact that $p_j > 5B/4$ for each normal job $J_j$. This means that the $4c - 3$ separation jobs released at time $4kB$ cannot complete their processing by time $(4k+1)B$. Hence, the jobs which have completed their processing by time $(4k+1)B$ consist of...
at most \( c \) initial jobs, at most \((4c-3)k\) separation jobs and \(3k\) normal jobs. Consequently, at most \((4k+1)c < (4k+2)c\) jobs have completed their processing by time \((4k+1)B\). This contradicts the Full Delivery Batch Property.

**Case 2.** \( a_{\pi(3k-2)} + a_{\pi(3k-1)} + a_{\pi(3k)} > B \). In this case, the processing completion time of \( J_{\pi(3k-1)} \) is \( \rho_{\pi(3k-1)} = 4B(k-1) + p_{\pi(3k-2)} + p_{\pi(3k-1)} < 4kB \) and the processing completion time of \( J_{\pi(3k)} \) is \( \rho_{\pi(3k)} = 4B(k-1) + p_{\pi(3k-2)} + p_{\pi(3k-1)} + p_{\pi(3k)} > 4kB \), where the first inequality follows from the fact that \( p_j < 3B/2 \) for each normal job \( J_j \). This also means that the \( 4c-3 \) separation jobs released at time \( 4kB \) cannot complete their processing by time \( 4kB \). Hence, the jobs which have completed their processing by time \( 4kB \) consist of at most \( c \) initial jobs, at most \((4c-3)k\) separation jobs and \(3k-1\) normal jobs. Consequently, at most \((4k+1)c-1 < (4k+1)c\) jobs have completed their processing by time \( 4kB \). Again, this contradicts the Full Delivery Batch Property. The result follows. \( \square \)

The above theorem means that, even when \( c = 1 \), the scheduling problem \( 1 \rightarrow D|c \geq 1, r_j|C_{\text{max}} \) is strongly NP-hard.

### 4.2. Approximation algorithm

The following approximation algorithm \( H_2 \) is proposed here to solve the scheduling problem \( 1 \rightarrow D|c \geq 1, r_j|C_{\text{max}} \). For a time moment \( t \), we use \( U(t) \) to denote the set of all jobs which have been released but have not started their processing by time \( t \).

**Approximation algorithm \( H_2 \)**

- **Step 1:** Set \( t = 0 \).
- **Step 2:** Set \( V(t) = \{ J_j \in U(t) : p_j \leq 2t \} \).
  1. (1) If \( V(t) \neq \emptyset \), find the job \( J_k \) such that \( p_k = \min\{ p_j : J_j \in U(t) \} \) and schedule \( J_k \) at time \( t \). Set \( t = t + p_k \), go to step 2.
  2. (2) If \( V(t) = \emptyset \), wait until a new time \( t' \) such that \( V(t') \neq \emptyset \). Set \( t = t' \), go to step 2.
- **Step 3:** Assign the jobs into \( \lceil \frac{t}{2} \rceil \) delivery batches such that
  1. a job with a earlier ready time is delivered no later than that with a later ready time; and
  2. each delivery batch, apart from the first delivery batch, contains exactly 3 jobs.
- **Step 4:** Whenever the vehicle and a delivery batch are available, transport the delivery batch with the lowest ready time. If all jobs are delivered to the customer and the vehicle returns to the machine, then stop the algorithm.

In the above algorithm, Step 2 is used to arrange the processing of jobs, in which we accept an on-line version with the jobs arriving over time. Step 3 is used to form the delivery batches, in which we accept an off-line version, since the number \( n \) of jobs is assumed to be known. Step 4 arranges the delivery of the delivery batches formed in Step 3 in a greedy way. For convenience, we use \( J_j \) to denote the \( j \)-th job processed on the machine in \( H_2 \). Then \( i < j \) means that \( J_j \) is processed after \( J_i \).

The following two observations about \( H_2 \) will be used in the proof of Theorem 4.2.1.

**Observation 1.** \( S_1 = \min\{ \max\{ r_j, \frac{1}{2} p_j \} \} \).

**Observation 2.** If \( J_j \) is processed after \( J_i \) and \( p_j < p_i \), then \( r_j > S_1 \).

**Theorem 4.2.1.** \( C^{H_2} \leq \frac{5}{4} C^*, \) and the bound is tight.

**Proof.** Suppose that \( I \) is a smallest counterexample such that \( C^{H_2} > \frac{5}{4} C^* \), where “smallest” means that \( I \) has the smallest number of jobs among all counterexamples. Let \( \pi \) be the schedule obtained from \( H_2 \). Then there are \( d = \lceil \frac{n}{2} \rceil \) delivery batches, say \( B_1, \ldots, B_d \), in \( \pi \). Note that \( \delta(B_i) = \rho(B_i) \) and \( \delta(B_i) = \max\{ \rho(B_j), \delta(B_{j-1}) + T \} \) for \( 2 \leq j \leq d \). Denote by \( B_k \) the last delivery batch satisfying \( \rho(B_k) = \delta(B_k) \) in \( \pi \). Then we have \( C^{H_2} = \rho(B_k) + (d-k+1)T \).

Let \( J_h \) be the last processing completed job in \( B_k \). Then \( \rho_h = \rho(B_k) \).

**Claim 1.** \( J_1, \ldots, J_h \) are processed contiguously in \( \pi \), i.e., there is no idle time between these jobs on the machine.

Otherwise, let \( J_z \) (\( 2 \leq z \leq h \)) be the last job such that \( J_z, \ldots, J_h \) are processed contiguously in \( \pi \). We remove \( J_1, \ldots, J_{z-1} \) from \( I \). This does not change the starting times of \( J_z, \ldots, J_h \) and the departure times of \( B_k, \ldots, B_d \), and so does not change the value of \( C^{H_2} \). Furthermore, the value of \( C^* \) is not increased. Thus, we obtain a smaller counterexample than \( I \), a contradiction. Claim 1 follows.
By Claim 1, we have \( C^{H_2} = S_1 + p_1 + \cdots + p_h + (d - k + 1)T \).

Let \( \pi^* \) be an optimal schedule for \( I \). We also assume that each delivery batch in \( \pi^* \), apart from the first delivery batch, contains exactly \( c \) jobs, since, otherwise, we can always fill these delivery batches with more jobs from earlier delivery batches without increasing the objective value.

Let \( J_j \) be the longest job in \( I \), i.e., \( p_j = \max \{ p_j : J_j \in I \} \). We modify the release dates of each job \( J_j \in I \) in the following way.

If \( r_j < S_1 \), we define \( r_j' = S_1 \). Note that, in this case, by Observation 1, we have \( \frac{1}{2}p_j \geq S_1 \), and so \( r_j' - r_j \leq S_1 \leq p_j \).

If \( r_j \geq S_1 \) and \( J_j \) is the longest job in \( \{ J_1, J_2, \ldots, J_j \} \), we define \( r_j' = r_j \).

If \( r_j \geq S_1 \) and \( J_j \) is not the longest job in \( \{ J_1, J_2, \ldots, J_j \} \), then let \( i < j \) be maximum such that \( p_j < p_i \). Then we define \( r_j' = \max \{ r_j, S_1 + p_i \} \). Note that, in this case, by Observation 2, we have \( r_j > S_1 \), and so \( r_j' - r_j \leq p_i \leq p_1 \).

The new instance obtained from \( I \) by the above modification of release dates is denoted by \( I' \). It can be observed that \( r_j' - r_j \leq p_j \) for each job \( J_j \). By delaying the processing and delivery of each job in \( \pi^* \) by a time length \( p_j \), we obtain a feasible schedule for \( I' \) with the makespan \( C^* + p_j \). Thus, we have \( C^*(I') \leq C^* + p_j \).

Clearly, \( \pi \) is a feasible schedule for \( I' \). Note that the job with the smallest processing time is always scheduled in \( \pi \) and no smaller job in \( I' \) is released until its completion. If preemption is allowed to all jobs, these does not increase the objective value. It is easy to see that \( \pi \) is the same as the schedule obtained by the algorithm \( H_1 \). Thus, we have \( C^*(I') = \rho(B_2) + (d - k + 1)T \).

**Claim 2.** \( p_l \geq \frac{2}{3}C^* \) and \( J_1 \) is the unique longest job in \( I \). Further, we have \( r_l \leq S_1^* < \frac{1}{2}p_l \).

If \( p_l \geq \frac{2}{3}C^* \), then \( C^{H_2} = C^*(I') \leq C^* + p_l \leq \frac{4}{3}C^* \), a contradiction. Thus, we have \( p_l > \frac{2}{3}C^* \). If there are at least two longest jobs in \( I \), then \( C^* \geq 2p_l > \frac{4}{3}C^* \), a contradiction. If \( S_1^* \geq \frac{1}{2}p_l \), then \( C^* \geq \frac{1}{2}p_l + p_l = \frac{3}{2}p_l \), a contradiction. **Claim 2** follows.

We distinguish two cases in the following discussion.

**Case 1:** \( J_l \notin \{ J_1, \ldots, J_h \} \).

By **Claim 2** and Proposition 2.1, we have \( p_l > \frac{2}{3}C^* \geq \frac{2}{3}(p_1 + \cdots + p_h + p_l) \). Thus, we have \( p_l > 2(p_1 + \cdots + p_h) \).

By Observation 1 and **Claim 2**, we have \( S_1 \leq \max \{ r_l, \frac{1}{2}p_l \} = \frac{1}{2}p_l \). Thus, we have

\[
\begin{align*}
C^{H_2} & = S_1 + p_1 + \cdots + p_h + (d - k + 1)T \\
& \leq \frac{1}{2}p_l + p_1 + \cdots + p_h + (d - k + 1)T \\
& \leq \frac{1}{2}p_l + \frac{2}{3}(p_1 + \cdots + p_h) + \frac{1}{3}(p_1 + \cdots + p_h) + (d - k + 1)T \\
& \leq \frac{1}{2}p_l + \frac{2}{3}(p_1 + \cdots + p_h) + \frac{1}{6}p_1 + (d - k + 1)T \\
& = \frac{2}{3}(p_1 + \cdots + p_h + p_l) + (d - k + 1)T \\
& \leq \frac{2}{3}C^* + C^* \\
& = \frac{5}{3}C^*,
\end{align*}
\]

a contradiction. This completes the discussion of Case 1.

**Case 2:** \( J_1 \in \{ J_1, \ldots, J_h \} \).

By **Claim 2** and Proposition 2.1, we have \( p_l > \frac{2}{3}C^* \geq \frac{2}{3}(p_1 + \cdots + p_h) \).

**Claim 3.** \( U(S_l) = \{ J_1 \} \) and at most \( h \) jobs arrive at or before \( S_l \).

Since \( J_1 \) is the shortest job in \( U(S_l) \) and \( J_1 \) is unique, we have \( U(S_l) = \{ J_1 \} \). Since at most \( h - 1 \) jobs have been processed before \( S_l \), there are at most \( h \) jobs which arrive at or before \( S_l \). **Claim 3** follows.

**Claim 4.** \( J_1 \in \{ J_1^*, \ldots, J_h^* \} \).
Otherwise, $J_l \notin \{J_1^*, \ldots, J_h^*\}$. By Claim 3, $\{J_1^*, \ldots, J_h^*\}$ contains at least one job arriving after $S_l \geq \frac{1}{2}p_l$. Thus, we have $S_l \geq p_l^* \geq \frac{1}{2}p_l$, a contradiction. Claim 4 follows.

From Claim 4, we have $C^* \geq p_l + (d - k + 1)T$. Furthermore, we have

$$C^{H_2} = S_1 + p_1 + \cdots + p_h + (d - k + 1)T$$

$$\leq \frac{1}{2}p_1 + p_1 + \cdots + p_h + (d - k + 1)T$$

$$= \frac{1}{2}(p_1 + \cdots + p_h) - \frac{1}{2}p_1 + p_1 + (d - k + 1)T$$

$$\leq (p_1 + \cdots + p_h) - \frac{1}{3}(p_1 + \cdots + p_h) + p_l + (d - k + 1)T$$

$$= \frac{2}{3}(p_1 + \cdots + p_h) + p_l + (d - k + 1)T$$

$$\leq \frac{2}{3}C^* + C^*$$

$$= \frac{5}{3}C^*,$$

a contradiction. This completes the discussion of Case 2.

From the above discussions, we conclude that there is no counterexample to Theorem 4.2.1. It follows that $C^{H_2} \leq \frac{5}{3}C^*$.

In order to show that the bound $C^{H_2} \leq \frac{5}{3}C^*$ is tight, we consider an instance as follows. Let $c = 1$. A job $J_1$ with the processing time $p_1 = 2(n - 1)T$ is released at time 0; a job $J_2$ with the processing time $p_2 = (n - 1)T$ is released at time $(n - 1)T$; $n - 2$ jobs $J_3, \ldots, J_h$ with the zero processing time are released at time $(2n - 1)T + \epsilon$. Algorithm $H_2$ processes the jobs in the order $J_2, J_1, J_3, J_4, \ldots, J_h$. The departure times of the delivery batches are $2(n - 1)T, 4(n - 1)T, 4(n - 1)T + T, 4(n - 1)T + 2T, \ldots, 4(n - 1)T + (n - 2)T = (5n - 6)T$. Then we have $C^{H_2} = (5n - 5)T$. However, the optimal schedule is to process and deliver these jobs as early as possible in the order $J_1, J_3, \ldots, J_h, J_2$, which results in $C^* = (3n - 2)T + \epsilon$. Thus, we have $\frac{C^{H_2}}{C^*} = \frac{(5n-5)T}{(3n-2)T+\epsilon} \to \frac{5}{3}$ when $n \to +\infty$ and $\epsilon \to 0$. The result follows. \(\Box\)

References


