Cloud Scheduling with Discrete Charging Units

Ming Ming Tan, Runtian Ren, and Xueyan Tang

Abstract—We consider a scheduling problem for running jobs on machines rented from the cloud. Cloud service providers such as Amazon EC2 and Google Cloud offer machines to rent on demand, and charge the rental usage by a specific interval of time, say at an hourly rate. This pricing model creates an interesting optimization problem called Interval Scheduling with Discrete Charging Units (ISDCU) which assigns jobs to run on the machines with the objective of minimizing the rental cost. In this paper, we study the problem of ISDCU where each machine can process a maximum of g jobs simultaneously. We focus on interval jobs where each job must be assigned to a machine upon its arrival and run for a required processing length. We show that ISDCU is NP-hard even for the case of $g = 1$. We also show that no deterministic online algorithm can achieve a competitive ratio better than $\max\{2, g\}$ in the non-clairvoyant setting, and better than $\max\{3/2, g\}$ in the clairvoyant setting. Lastly, we develop and analyze several online algorithms, most of which achieve a competitive ratio of $O(g)$.

Index Terms—Cloud scheduling, interval scheduling, online algorithm.

1 Introduction

"Pay-as-you-go" billing is a most salient feature of clouds. In practice, many clouds charge the rental of machines by a discrete unit, such as by the hour in Amazon EC2 [4] or by the minute in Google Cloud [3]. Every *partially* used unit is charged as a *full* unit. In this paper, we conduct a theoretical study on job scheduling for optimizing the rental cost with discrete charging units. We focus on a fundamental form of job scheduling – to schedule interval jobs with fixed starting and ending times. The objective is to rent machines and assign jobs to run on the machines so as to minimize the rental cost. We call this problem Interval Scheduling with Discrete Charging Units (ISDCU).

1.1 Problem Description

Formally, the input to the ISDCU problem is a set of jobs each defined by a processing interval $[a, d)$ where a and d are the arrival and departure times of the job respectively, and a positive integer τ specifying the length of a charging unit. The jobs need to be scheduled on identical machines where each machine can simultaneously process at most a fixed number of g jobs at any time. A job must remain in the same machine throughout its lifetime, unless explicitly stated otherwise. Machines may be rented for any length of time. The cost of renting a machine for a duration T is $\lceil \frac{T}{\tau} \rceil$. The goal is to schedule all the jobs in a way that minimizes the total rental cost incurred.

Essentially, if a machine is launched at time s and terminated at time t, then it is charged for every interval of $[s, s+\tau)$, $[s+\tau, s+2\tau)$, $\ldots$, $[s+(i-1)\tau, s+i\tau)$ that $[s, t)$ overlaps, where $i = \lceil \frac{t-s}{\tau} \rceil$. We refer to each of these length-τ intervals as a *charging unit*. The target to minimize the total rental cost is the same as to minimize the total number of charging units required to process all jobs.

• *The authors are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: mmtan830@gmail.com, {RENR0002, asxytang}@ntu.edu.sg.*

1.2 Related Work

In a special case of ISDCU where the length τ of a charging unit is sufficiently large (longer than the period from the arrival of the first job to the departure of the last job), the number of charging units is equivalent to the number of machines used. Thus, ISDCU degenerates to the basic interval scheduling problem [10], which aims at minimizing the number of machines used for processing a set of interval jobs. The basic interval scheduling problem is polynomially solvable and the optimal number of machines used to process all jobs is $\lceil \frac{D}{g} \rceil$, where D is the maximum number of jobs that pairwise overlap.

In another special case of ISDCU where the length τ of a charging unit is infinitely small, the total rental cost is equivalent to the total busy time of machines, where a machine is considered busy when there is at least one job running on it. In this case, ISDCU degenerates to interval scheduling with bounded parallelism [2], [6], [9], [12], [16], [17], which aims at minimizing the total busy time of the machines used for processing a set of interval jobs. Winkler *et al.* [17] proved the NP-completeness of this problem when $g \geq 2$. Shalom *et al.* [16] showed that any online algorithm for interval scheduling with bounded parallelism is g-competitive and the competitive ratio is tight in terms of g. However, as shall be shown, not all online algorithms are g-competitive for ISDCU. Thus, it is important to design good online scheduling strategies.

Other related problems that share aspects with our model include scheduling with calibrations [1], [5] and rent minimization [15]. Both problems consider machines that can process only one job at a time and flexible jobs that have laxity in starting. Each job has a release time, a processing length and a deadline. The job needs to run for a consecutive period equal to its processing length between its release time and deadline. The input to the problem of scheduling with calibrations includes a set of flexible jobs, a fixed number of identical machines, and an integer ℓ specifying a calibration length. Every job is scheduled without preemption and completely within a single calibrated interval. The goal is to find a schedule that minimizes the number of calibrations performed. If an instance of scheduling with calibrations has the input jobs being interval jobs, then the problem looks similar to an instance of ISDCU, by viewing each calibration as a charging unit. However, our ISDCU problem has no restriction that every job must lie within a single charging unit. Moreover, the problem of scheduling with calibrations assumes a fixed number of machines (each machine can accommodate just one job at any time), while ISDCU can open as many machines as needed and each machine can run multiple jobs in parallel. The rent minimization problem extends the classical machine minimization problem [7] to optimize the rental cost. Saha [15] developed a constant factor offline algorithm and an $O(\log \mu)$-competitive online algorithm for rent minimization where μ is the max/min job processing length ratio. Different from the above two problems, we focus on interval jobs. We show that it is NP-hard to optimize the rental cost for interval jobs with fixed starting and ending times even on machines that can process only one job at a time.

1.3 Our Results

We show that ISDCU is NP-hard even when the machine capacity $g = 1$ (i.e., each job must occupy a machine exclusively). We establish a lower bound of $\max\{2, g\}$ on the competitiveness of any deterministic online algorithm in the non-clairvoyant setting, and a lower bound of $\max\{3/2, g\}$ in the clairvoyant setting. We present and analyze a range of online algorithms for ISDCU in both the non-clairvoyant and the clairvoyant settings. The best competitive ratio of the online algorithms presented achieves a factor of g for $g > 2$ which matches the lower bound established.

2 Complexity

We begin our study by investigating the complexity of ISDCU. We consider the decision problem of ISDCU: Given an instance of ISDCU, is there a feasible schedule that incurs at most K charging units? The decision problem of ISDCU is NP-hard for $g \geq 2$. This follows from the NP-hardness of the problem of interval scheduling with bounded parallelism [17]. We prove that the decision problem of ISDCU is NP-hard even for the case of $g = 1$. This can be derived via reduction from the problem of circular arc coloring (CAC). Garey *et al.* [8] proved that CAC is NP-complete. A formal description of CAC is as follows.

Problem: Circular Arc Coloring (CAC)
Instance: A circle of length L, n circular arcs each defined by a pair of distinct positive integers $\{s_i, f_i\}$ on the circle, and a positive integer K.
Question: Does there exist a partition of the circular arcs into at most K sets such that arcs in each set are disjoint?

Theorem 1. *The ISDCU problem with $g = 1$ is NP-hard.*

Proof. We establish a reduction from CAC to ISDCU with $g = 1$. Given an instance of CAC, we build an instance of ISDCU as follows. We set the length of a charging unit $\tau = L$. For each arc $\{s_i, f_i\}$, if $s_i < f_i$ (the arc does not contain the point L), we create a job $[s_i, f_i)$; otherwise, we create a job $[s_i, f_i + L)$ (see Figure 1 for illustration). If a feasible coloring exists for the CAC instance, we turn it into a feasible schedule for the ISDCU instance by assigning the corresponding jobs of all the arcs with the same color to one machine. It is apparent that each machine needs only one charging unit to process all its jobs. Thus, the total number of charging units required by this schedule is K.

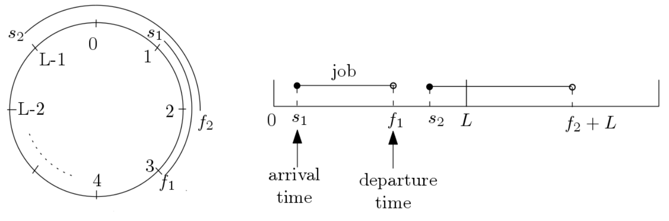

Fig. 1. An instance of CAC and its corresponding job representation in ISDCU.

Finally, observe that all jobs in the ISDCU instance arrive before time L and we only need at most two charging units in each machine to process all the jobs. Given a feasible schedule of the ISDCU instance, we can partition the machines into K charging units. Each charging unit will be assigned a different color and a job that falls within a charging unit will have its corresponding arc assigned the same color as the charging unit. If a job has its processing interval crossing two charging units R_1 and R_2 where R_1 precedes R_2, we assign the color of R_2 to its corresponding arc. It can be shown that such a coloring is feasible. First, since all jobs within a single charging unit are disjoint, their corresponding arcs are disjoint. Second, if a job crosses two charging units R_1 and R_2, then the job must end after time L. Since no job arrives after L, it must be the only job in R_2. Thus, it is feasible to color the corresponding arc with the color of R_2. □

3 Online algorithms

Now, we investigate algorithms for ISDCU in the online setting where the jobs are released one at a time and each job has to be assigned before the next one is released. We shall consider both the non-clairvoyant and the clairvoyant settings of ISDCU. In the non-clairvoyant setting, the departure time of each job is not known at its arrival and thus cannot be used for scheduling purposes. In the clairvoyant setting, the departure time of each job is known at its arrival and can be used for scheduling purposes.

Throughout the paper, we shall use the following definitions. A charging unit represented by $[s, s + \tau)$ starts at time s and expires at time $s + \tau$. A charging unit $[s, s + \tau)$ is said to be *open* at time t if $s \leq t < s + \tau$. Each charging unit is associated with a machine. A machine is said to be *open* at time t if it has a charging unit open at t. An open machine is said to be *full* if there are g jobs running on it, otherwise it is said to be *available* if there are less than g jobs running on it. An open machine is said to be *idle* if there is no job running on it, otherwise the machine is said to be *non-idle*. A machine is said to be *initiated* at time t if its first charging unit starts at t. In general, a machine is said to expire at time

t if its ongoing charging unit expires at t. When the ongoing charging unit of a machine expires, the machine will have its service extended with a new charging unit only if it is non-idle.

3.1 A Variant of ISDCU with Job Migration Allowed

We first present an online greedy algorithm for solving a variant of ISDCU with job migration allowed. In this variant, a job running on a machine can be interrupted and migrated to continue running on another machine instantaneously.

Our greedy algorithm works as follows. When a job J arrives, the algorithm checks whether there exists any available machine. If at least one machine is available, J is placed on the available machine with the latest expiration time. Ties are broken by giving preference to non-idle machines. Otherwise, if there is no available machine, a new machine is initiated to accommodate J.

The greedy algorithm carries out job migration only at job departures. When a job departs, the algorithm migrates the remaining active jobs to fill up the open machines in decreasing order of their expiration times. Suppose there remain n active jobs and let M_1, M_2, M_3, $\ldots$ be the list of all the open machines sorted in decreasing order of their expiration times. Then, by job migration, the algorithm ensures that the first $\lceil \frac{n}{g} \rceil - 1$ machines M_1, M_2, $\ldots$, $M_{\lceil \frac{n}{g} \rceil - 1}$ are all full and the $\lceil \frac{n}{g} \rceil$-th machine $M_{\lceil \frac{n}{g} \rceil}$ has $n - (\lceil \frac{n}{g} \rceil - 1)g$ jobs running on it. A pseudo code description of the greedy algorithm is given in Algorithm 1.

Algorithm 1 A greedy algorithm for the variant of ISDCU with job migration allowed

```
1:  while true do
2:      if a job arrives then
3:          if there exists at least one available machine then
4:              assign the job to the available machine with the latest expiration time
5:          else
6:              initiate a new machine with a charging unit
7:              assign the job to the new machine
8:      if a job departs then
9:          sort all the open machines in decreasing order of their expiration times
10:         let $M_1$, $M_2$, ..., $M_m$ be the sorted list of open machines
11:         $i \leftarrow 1$
12:         $j \leftarrow m$
13:         while $i \neq j$ do
14:             while $M_i$ is full do
15:                 $i \leftarrow i + 1$
16:             while $M_j$ is idle do
17:                 $j \leftarrow j - 1$
18:             if $i \neq j$ then
19:                 migrate a job from $M_j$ to $M_i$
```

Obviously, the greedy algorithm works in both the non-clairvoyant and clairvoyant settings. The job assignment at each job arrival and the job migration at each job departure both have time complexities polynomial in the number of open machines. Since the number of open machines at any time is bounded by the total number of jobs to schedule, the time complexity of Algorithm 1 is polynomial in the total number of jobs.

By definition, if an incoming job is placed on an idle machine, all the open machines with earlier expiration times must be full. Together with the job migration at job departures, it is straightforward that the schedule produced by Algorithm 1 has the following property.

Proposition 1. *At any time, non-idle machines must have equal or later expiration times than idle machines.*

Proposition 1 implies that when a new charging unit is started to extend the service of a machine, all the open machines must be non-idle. Remember that a charging unit is started to initiate a new machine only if all the open machines are full. Thus, we can infer the following property.

Proposition 2. *At the starting time of any charging unit, all the open machines are non-idle.*

By induction, we can also prove the following property.

Proposition 3. *At any time, if there are m non-idle machines, then there are at least $(m-1)g + 1$ active jobs.*

The above claim is trivial at the first job arrival. By the definition of the algorithm, the claim holds after carrying out job migration at each job departure. If the claim holds prior to a job arrival, it apparently remains true after the job arrival if the new job is not placed on an idle machine or a new machine. In the case that the new job is placed on an idle machine, it follows from Proposition 1 that all the non-idle machines must be full. In the case that the new job is placed on a new machine, all the open machines must be full. Thus, in both cases, the claim must still hold after the new job is placed.

Now, we show that Algorithm 1 is indeed optimal. Let R_1, R_2, R_3, $\ldots$ be the list of all the charging units used by Algorithm 1 for scheduling a set of jobs, sorted in increasing order of their starting times. We show by induction that for any $k \geq 1$, there exists an optimal schedule with the first k charging units starting at the same times as R_1, R_2, $\ldots$, R_k.

For $k = 1$, Algorithm 1 initiates the first machine at the first job arrival. Obviously, the first charging unit of an optimal schedule cannot start later than the first job arrival. If it starts earlier than the first job arrival, we can delay its starting to the first job arrival without violating the feasibility of the schedule. Thus, the claim holds for the first charging unit. Now, suppose the claim holds for the first $k-1$ charging units. Let t be the time when the k-th charging unit R_k used by Algorithm 1 starts. Let h be the number of charging units among R_1, R_2, $\ldots$, R_{k-1} that are still open at time t. By Proposition 2, the machines of all these h charging units as well as R_k are non-idle at time t. By Proposition 3, there are at least $hg + 1$ active jobs at t. Since the first $k-1$ charging units of the optimal schedule start at the same times as R_1, R_2, $\ldots$, R_{k-1}, the optimal schedule must also have only h charging units remaining open at time t. Therefore, to accommodate all the active jobs at t, the k-th charging unit of the optimal schedule cannot start later than t. On the other hand, R_1, R_2, $\ldots$, R_{k-1} used by Algorithm 1 can accommodate all the active jobs up till

t. If the k-th charging unit of the optimal schedule starts earlier than t, we can delay its starting to t. As a result, the claim holds for the first k charging units.

Theorem 2. *Algorithm 1 produces an optimal schedule for the variant of ISDCU with job migration allowed.*

3.2 Lower Bounds

For the rest of the paper, we study online algorithms for ISDCU where each job must run on the same machine throughout its execution. First, we investigate the lower bounds on the competitiveness of ISDCU.

We start by deriving the lower bounds on the competitive ratio of any deterministic online algorithm for ISDCU with $g = 1$.

Theorem 3. *In the non-clairvoyant setting, no deterministic online algorithm for ISDCU with $g = 1$ can achieve a competitive ratio better than 2.*

Proof. Consider any deterministic online algorithm O. Let ϵ be a small positive value and n be a positive integer where $(n+1)\epsilon < \tau$. Let a job arrive at time $i\epsilon$ for each $i = 0, 1, \cdots, n-1$ (see Figure 2 for illustration). These n jobs all depart at time $n\epsilon$. Hence, these n jobs overlap with one another and will be assigned to n different machines. We label these n machines $M_1, M_2, \cdots, M_n$, sorted in increasing order of their initiation times. Let $R_i = [(i-1)\epsilon, (i-1)\epsilon + \tau)$ denote the charging unit of M_i which initiates the machine. At time $(n+1)\epsilon$, let a set $\mathcal{J}$ of n new jobs arrive. These n new jobs will be scheduled to run on n different machines. In the non-clairvoyant setting, the algorithm O has no information on the departure time of a job when the job is scheduled. If O schedules a job of $\mathcal{J}$ to run on machine M_i, then we let the job depart at $i\epsilon + \tau$. A new charging unit on M_i is required since the job cannot fit into the charging unit R_i. If O schedules a job of $\mathcal{J}$ by initiating a new machine, then we let the job depart at τ. In both cases, a new charging unit is required for each of the n new jobs in $\mathcal{J}$. Hence, the total number of charging units needed is $2n$.

Suppose a machine M_i has two charging units. Let J_i denote the job in $\mathcal{J}$ assigned to M_i. If $i < n$, then J_i can fit into R_{i+1}. If $i = n$, then J_i cannot fit into any R_j for $1 \le j \le n$. Lastly, a job in $\mathcal{J}$ that was assigned to a new machine by O can fit into any R_j for $1 \le j \le n$. From these observations, we can deduce an optimal schedule for this instance in the following way. For each $i < n$ where M_i has two charging units in the schedule of O, we assign J_i to M_{i+1}. If M_n has two charging units in the schedule of O, we assign J_n to M_1. The remaining jobs in $\mathcal{J}$ can be assigned to the rest of the available machines arbitrarily. We can see that all jobs in $\mathcal{J}$ would fit into the charging units that initiate the machines $M_1, M_2, \cdots, M_n$, except job J_n which departs at $n\epsilon + \tau$ (if it exists). Hence, in an optimal schedule, each machine M_i for $2 \le i \le n$ needs not extend its service upon the expiration of R_i. Machine M_1 will need to extend its service after R_1 expires if there is a job which departs at $n\epsilon + \tau$. Consequently, the optimal number of charging units needed is at most $n + 1$. This suggests that no deterministic online algorithm can achieve a competitive ratio less than $\frac{2n}{n+1}$, which can be made arbitrarily close to 2 as n approaches infinity. □

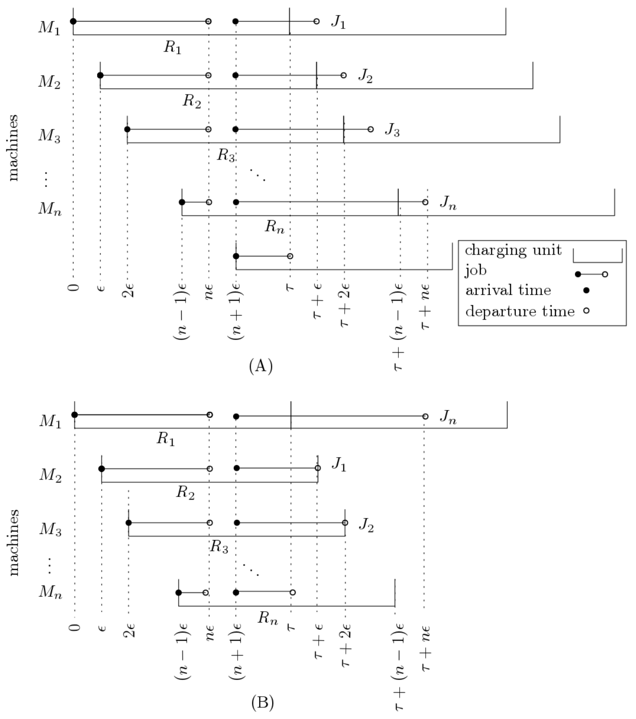

Fig. 2. (A) An online schedule in the non-clairvoyant setting. (B) An optimal schedule for the instance of (A).

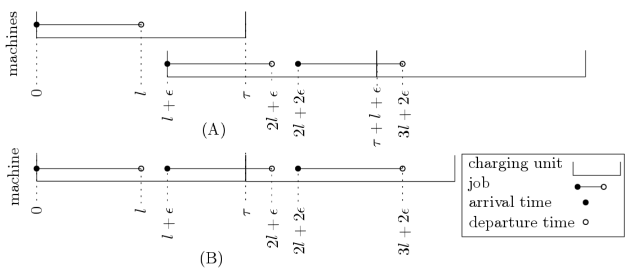

Fig. 3. (A) An online schedule in the clairvoyant setting. (B) An optimal schedule for the instance of (A).

Theorem 4. *In the clairvoyant setting, no deterministic online algorithm for ISDCU with $g = 1$ can achieve a competitive ratio better than $\frac{3}{2}$.*

Proof. Let $l = \frac{\tau}{2}$ and ϵ be a small positive value satisfying $3l + 2\epsilon \le 2\tau$. Consider any deterministic online algorithm O. Let the first job arrive at time 0 and depart at time l. At time $l + \epsilon$, let the second job arrive and the departure time of this job is set to $2l + \epsilon > \tau$. If O schedules the second job by initiating a new machine, then we let the third job arrive at time $2l + 2\epsilon$ and depart at time $3l + 2\epsilon \le 2\tau$ (see Figure 3 for illustration). In total, three charging units are needed. However, in the optimal offline solution, all three jobs would be assigned to the same machine, and only two charging units are required.

On the other hand, if O schedules the second job on the first machine, then we let the third job arrive at $l + 2\epsilon$ and depart at τ (see Figure 4 for illustration). Since the third job overlaps with the second job, the third job must be assigned to a new machine. In total, three charging units are required. However, in the optimal offline solution, the

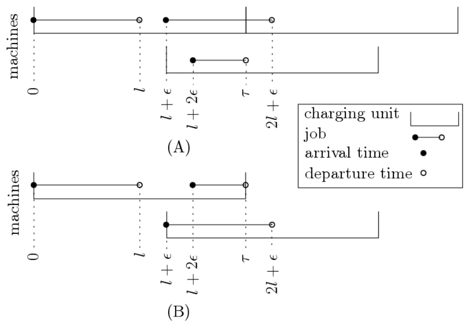

Fig. 4. (A) Another online schedule in the clairvoyant setting. (B) An optimal schedule for the instance of (A).

second job would be assigned to a new machine, and the third job would be assigned to the first machine. Then, only two charging units are needed. □

Recall from Section 1.2 that ISDCU with an infinitely small length of charging unit is essentially a problem of interval scheduling with bounded parallelism. The latter problem has a lower bound of g on the competitiveness of any deterministic online algorithm [16]. Thus, we have the following corollaries.

Corollary 1. *In the non-clairvoyant setting, no deterministic online algorithm for ISDCU can achieve a competitive ratio better than* $\max\{2, g\}$.

Corollary 2. *In the clairvoyant setting, no deterministic online algorithm for ISDCU can achieve a competitive ratio better than* $\max\{\frac{3}{2}, g\}$.

3.3 Upper Bounds

Now, we investigate the upper bounds on the competitiveness of ISDCU.

While our ISDCU problem degenerates to Interval Scheduling with Bounded Parallelism (ISBP) [6] by ignoring the discrete charging unit, an optimal schedule for ISBP may not work well for ISDCU in general. In ISBP, idle machines are indistinguishable. But in ISDCU, idle machines with different expiration times can give rise to different rental costs. Consider the following instance. Let n be an integer and ϵ be a small positive value such that $2n\epsilon < \tau$. We release n groups of jobs as follows. At each time $2(i-1)\epsilon$ for $i = 1, 2, \ldots, n$, we release g jobs which would all depart at time $(2i-1)\epsilon$. In an optimal ISBP schedule, the g jobs in each group can be placed on a separate machine, which gives a total busy time of $n\epsilon$. However, in ISDCU, such a schedule incurs a total rental cost of n charging units. The optimal ISDCU schedule is to place all the jobs on one machine with a charging unit $[0, \tau)$ so that the rental cost is 1. This example shows that an optimal ISBP schedule is not necessarily good for ISDCU. Moreover, any online algorithm is g-competitive for ISBP and thus optimal since it matches the lower bound [16]. But this is not true for ISDCU as we discuss next.

An intuitive strategy to address the issue in the above instance is to favor open machines in job placement and avoid initiating new machines unless necessary. We refer to such an online algorithm as a *rational* algorithm. That is, a rational algorithm will only initiate a new machine

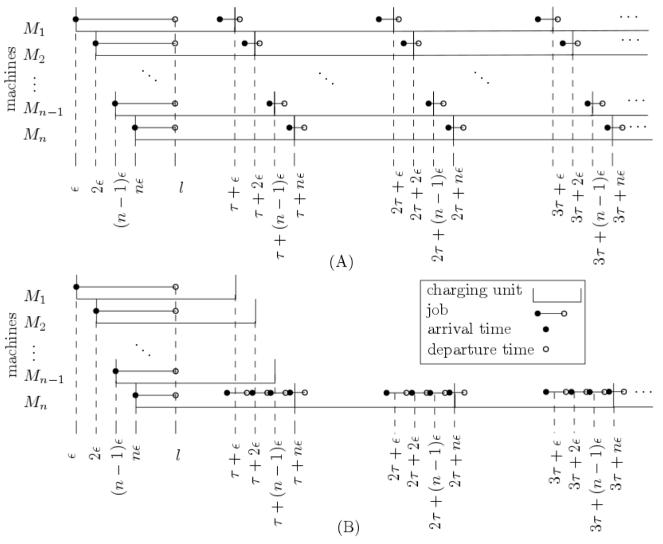

Fig. 5. (A) A rational schedule. (B) An optimal schedule for the instance of (A).

if no open machine is available to place an incoming job. However, simply following this rule is still not adequate. As the following theorem shows, not all rational algorithms can achieve bounded competitive ratios.

Theorem 5. *There exists a rational algorithm where the competitive ratio is unbounded.*

Proof. Consider the following instance of ISDCU with $g = 1$. Let n be a positive integer and ϵ be a small positive value. The input jobs of the instance are specified by the following processing intervals (see Figure 5 for illustration):

- $[i\epsilon, l)$ for $i = 1, 2, \ldots, n$;
- $[a_{i,j}, d_{i,j})$ for $i, j = 1, 2, \ldots, n$,

where $a_{i,j}$, $d_{i,j}$ and l are positive values satisfying $a_{1,1} > l > n\epsilon$ and $a_{i,j} < j\tau + i\epsilon < d_{i,j} < a_{i+1,j}$.

Since the jobs $[i\epsilon, l)$ for $i = 1, 2, \ldots, n$ overlap at time $n\epsilon$, a rational algorithm will assign these n jobs to different machines. Let M_i be the machine to which the job $[i\epsilon, l)$ is assigned. The rest of the jobs $[a_{i,j}, d_{i,j})$ for $i, j = 1, 2, \ldots, n$ are mutually disjoint. As a result, they can be placed on any of the machines $M_1, M_2, \cdots, M_n$.

Suppose a rational algorithm always assigns the job $[a_{i,j}, d_{i,j})$ to machine M_i. Since the charging units of M_i would always expire at times of the form $j\tau + i\epsilon$ where $j \geq 1$, the processing interval of the job $[a_{i,j}, d_{i,j})$ exceeds the expiration time of M_i's ongoing charging unit and results in the starting of a new charging unit. In total, the rental cost is $n^2 + n$. Note that the jobs $[a_{i,j}, d_{i,j})$ for $i, j = 1, 2, \ldots, n$ can actually be all placed on one machine, say M_n. Hence, the total rental cost of an optimal schedule is $2n$. Therefore, the competitive ratio of this rational algorithm is at least $\frac{n^2+n}{2n} = \frac{n+1}{2}$, which can be made arbitrarily large as n approaches infinity. □

The above theorem indicates that the main challenge in designing a decent algorithm is to decide which machine to assign a job to when there is more than one machine available to process the job. In the following, we propose several rational algorithms and study their competitive ratios. Before presenting these algorithms, we first introduce some preliminaries necessary for the competitive analysis.

3.3.1 Preliminaries

Consider an instance of ISDCU. Without loss of generality, we may assume that the first job arrives after time 0 and no machine is initiated before the arrival of the first job. Let $t_0 = 0$ and $t_i = t_{i-1} + \tau$ for $i \geq 1$. Let x_i^* be the number of machines open at t_i in an optimal schedule of the instance. Then, the optimal rental cost $\text{OPT} = \sum_{i \geq 1} x_i^*$, since the sets of charging units counted by different x_i^*'s are disjoint and $x_0^* = 0$. Let y_i be the maximum number of concurrent jobs in $[t_{i-1}, t_i]$.

Consider any schedule S of the instance. Let z_i be the number of machines that are open at t_i in S and contain at least one job $[a, d)$ where $a < t_i < d$. By definition, we have $z_0 = 0$. Since the number of concurrent jobs at time t_{i-1} is at least z_{i-1}, it follows that $z_{i-1} \leq y_i$. We have the following fundamental result to establish lower bounds on OPT.

Lemma 1. *Given any instance of ISDCU and any schedule of the instance, we have*

$$\forall i \geq 1, \quad x_{i-1}^* + x_i^* \geq \left\lceil \frac{y_i}{g} \right\rceil, \tag{1}$$

$$\forall i \geq 1, \quad x_{i-1}^* \geq \left\lceil \frac{z_{i-1}}{g} \right\rceil, \tag{2}$$

$$\text{OPT} \geq \frac{1}{2} \sum_{i \geq 1} \left\lceil \frac{y_i}{g} \right\rceil, \tag{3}$$

and

$$\text{OPT} \geq \sum_{i \geq 1} \left\lceil \frac{z_{i-1}}{g} \right\rceil. \tag{4}$$

Proof. To prove (1), note that the set of y_i jobs that are concurrent at some time in $[t_{i-1}, t_i]$ must be scheduled on machines that are open at either t_{i-1} or t_i. Since each machine can accommodate up to g jobs, we deduce that there must be at least $\lceil \frac{y_i}{g} \rceil$ machines open at either t_{i-1} or t_i. To prove (2), note that by definition, there are at least z_{i-1} jobs concurrent at t_{i-1}. These z_{i-1} jobs must be scheduled on machines open at t_{i-1}. Since each machine can accommodate at most g jobs, we must have at least $\lceil \frac{z_{i-1}}{g} \rceil$ machines open at t_{i-1}. Lastly, (3) and (4) follow from (1) and (2) since $\text{OPT} = \sum_{i \geq 1} x_i^*$ and $x_0^* = 0$. □

Let x_i be the number of machines open at t_i in schedule S. Similar to the optimal rental cost, it is easy to infer that the total rental cost of S is $\sum_{i \geq 1} x_i$. We shall analyze the competitive ratios of various algorithms by establishing relations between x_i and z_i as well as y_i and using Lemma 1 to bound the total rental cost with OPT.

In many of the proofs, we divide the analysis of x_i into two cases. If there exists a machine initiated in $(t_{i-1}, t_i]$, we make use of Lemma 2 below. If there is no new machine initiated in $(t_{i-1}, t_i]$, we identify a special job in each charging unit to study the relations between x_i, z_i and y_i according to the definition of the specific algorithm.

Lemma 2. *In a schedule produced by a rational algorithm, if there is at least one machine initiated in $(t_{i-1}, t_i]$, we have $x_i \leq \lceil \frac{y_i}{g} \rceil$.*

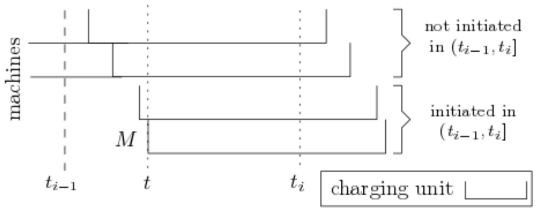
machines
not initiated in $(t_{i-1}, t_i]$
initiated in $(t_{i-1}, t_i]$
M
t_{i-1}
t
t_i
charging unit

Fig. 6. A rational schedule where there is at least one machine initiated in $(t_{i-1}, t_i]$.

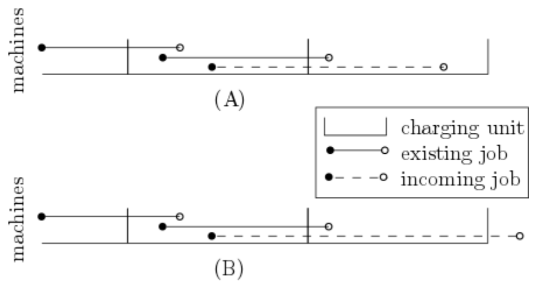

Fig. 7. (A) A machine that can fit an incoming job. (B) A machine that cannot fit an incoming job.

Proof. Consider the machine M which is initiated the latest in $(t_{i-1}, t_i]$ (see Figure 6 for illustration). Let t be the initiation time of machine M. Note that a machine which is open at t_i is either initiated in $(t_{i-1}, t_i]$ or has its service extended from a charging unit which was open at t_{i-1}. In either case, the machine has a charging unit containing t. Thus, all machines which are open at t_i must also be open at t. Since the algorithm is rational, all other machines which are open at t must be fully occupied in order for the new machine M to be initiated at time t. Therefore, the number of active jobs at time t is at least $(x_i - 1)g + 1$. This gives $y_i \geq (x_i - 1)g + 1$. Consequently, $x_i \leq \frac{y_i}{g} + \frac{g-1}{g}$. Since x_i is an integer, we must have $x_i \leq \lceil \frac{y_i}{g} \rceil$. □

3.3.2 A simple strategy for clairvoyant setting

In the clairvoyant setting, when a job arrives, the departure time of the job is known. Thus, we know how long a machine has to run to complete processing the job at its arrival. Intuitively, we can avoid starting a new charging unit if we place a job on an available machine that can fit the job. Formally, a machine is said to be able to *fit* a job if the job can be completed within the period spanned by the open charging unit and the subsequent charging units (if any) necessitated by the existing jobs running on the machine (see Figure 7 for illustration). Motivated by this idea, we propose a FitFirst strategy for online scheduling in the clairvoyant setting. The FitFirst strategy always gives preference to a machine which can fit an incoming job when there are multiple machines available to place the job. As shall be shown, the competitive ratio of any rational algorithm adopting the FitFirst strategy is at most $2g + 2$.

Recall that if a charging unit R is not the initiating charging unit of a machine, there must be at least one job crossing the starting time of R. Let $J(R)$ be the earliest arriving job that crosses the starting time of R. We call $J(R)$ the *defining job* of R. Essentially, a defining job of a charging

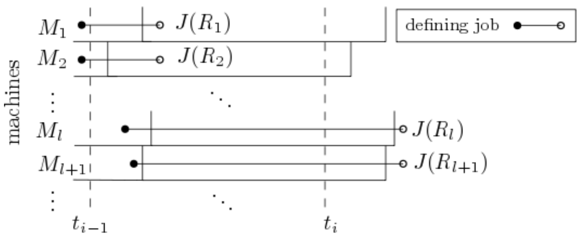

Fig. 8. A rational schedule in the clairvoyant setting where each defining job either arrives before t_{i-1} or departs after t_i.

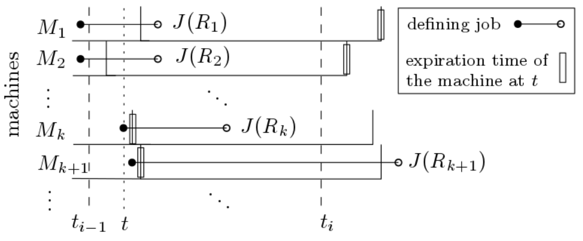

Fig. 9. A rational schedule in the clairvoyant setting where M_k is the machine with highest index whose defining job arrives after t_{i-1} and departs before t_i.

unit R is the job that is responsible for extending the service of the machine to start the charging unit R.

Theorem 6. *In the clairvoyant setting, any rational algorithm adopting the FitFirst strategy is $(2g + 2)$-competitive.*

Proof. Let x_i and z_i be defined on a schedule produced by a rational algorithm adopting the FitFirst strategy. Recall from Section 3.3.1 that the total rental cost of the schedule is given by $\sum_{i\geq 1} x_i$. We divide the analysis of x_i into two cases. If there is at least one machine initiated in $(t_{i-1}, t_i]$, by Lemma 2, we have

$$x_i \leq \left\lceil \frac{y_i}{g} \right\rceil . \tag{5}$$

Now suppose there is no new machine initiated in $(t_{i-1}, t_i]$. Then each charging unit R that is open at t_i has a defining job $J(R)$. Let $R_1, R_2, \cdots, R_{x_i}$ be the list of charging units open at t_i, such that $J(R_1), J(R_2), \cdots, J(R_{x_i})$ are sorted in non-decreasing order of their arrival times. We consider the following two cases:

Case 1: If $J(R_1), J(R_2), \cdots,\ J(R_{x_i})$ all arrive before t_{i-1}, then

$$x_i \leq z_{i-1}. \tag{6}$$

Case 2: If at least one defining job arrives at or after t_{i-1}, let l be the lowest index where $J(R_l)$'s arrival time is at least t_{i-1}. Then, the jobs $J(R_1), J(R_2), \cdots, J(R_{l-1})$ have arrival times before t_{i-1}. So, we have

$$l - 1 \leq z_{i-1}. \tag{7}$$

We further consider two sub-cases:

Case 2a: If the jobs $J(R_l), J(R_{l+1}), \cdots, J(R_{x_i})$ all depart after t_i (see Figure 8 for illustration), then they all cross t_i. Thus, we have

$$x_i - l + 1 \leq z_i. \tag{8}$$

In view of (7) and (8), we deduce

$$x_i \leq z_{i-1} + z_i. \tag{9}$$

Case 2b: If there exists one or more defining jobs whose arrival time is at least t_{i-1} and departure time is before t_i, let k be the highest index where $J(R_k)$ satisfies these properties (see Figure 9 for illustration). Let M_k be the machine of R_k. Let t be the arrival time of $J(R_k)$. Consider each M_j where $j < k$. Since this is the clairvoyant setting and since $J(R_j)$ arrives earlier than t, we know that M_j will have a charging unit open at t_i in order to process $J(R_j)$. Now, since $J(R_k)$ departs before t_i, $J(R_k)$ can actually fit into M_j upon its arrival unless M_j is fully occupied. Hence, if the FitFirst strategy is applied, we must have $R_1, R_2, \ldots, R_{k-1}$ all fully occupied at time t. By further taking into account of the job $J(R_k)$, we get $y_i \geq (k-1)g + 1$. Since k is an integer, we obtain

$$k \leq \left\lceil \frac{y_i}{g} \right\rceil . \tag{10}$$

If $k = x_i$, then (10) implies that

$$x_i \leq \left\lceil \frac{y_i}{g} \right\rceil . \tag{11}$$

Otherwise, if $k < x_i$, then for all $j > k$, $J(R_j)$'s departure time is after t_i. This means the jobs $J(R_{k+1})$, $J(R_{k+2})$, $\ldots$, $J(R_{x_i})$ all cross t_i. Hence, $x_i - k \leq z_i$. By (10), we have

$$x_i \leq z_i + \left\lceil \frac{y_i}{g} \right\rceil . \tag{12}$$

Overall, by (5), (6), (9), (11), (12), we always have

$$x_i \leq z_{i-1} + z_i + \left\lceil \frac{y_i}{g} \right\rceil .$$

Since $z_0 = 0$, we have $\sum_{i\geq 1} z_i = \sum_{i\geq 1} z_{i-1}$. Consequently, by Lemma 1,

$$\sum_{i\geq 1} x_i \leq 2 \sum_{i\geq 1} z_{i-1} + \sum_{i\geq 1} \left\lceil \frac{y_i}{g} \right\rceil \leq (2g+2) \cdot \mathrm{OPT}\,.$$

□

3.3.3 *ExpireLatest Algorithm*

Motivated by the optimal algorithm in Section 3.1 for the variant of ISDCU with job migration allowed, a natural idea to reuse open charging units as much as possible is to favor those with later expiration times. Thus, we propose a rational algorithm called ExpireLatest. When multiple machines are available to process an incoming job, the ExpireLatest algorithm assigns the job to the available machine which expires the latest. If there are multiple machines expiring at the same latest time, preference is given to the machines which are non-idle. Ties among non-idle machines or among idle machines (if any) may be broken arbitrarily.

The expiration time of a machine might differ between the clairvoyant and the non-clairvoyant settings. In the non-clairvoyant setting, when a job arrives, it is placed on a machine without any information of its departure time. Hence, the machine's expiration time is always the expiration time of the open charging unit on the machine. In other words, if a job is assigned to a machine M with an open charging unit $[s, s + \tau)$, the expiration time of M after job placement remains $s + \tau$. However, in the clairvoyant setting, when a job arrives and is placed on a machine, the departure time of the job is known. Thus, the number of

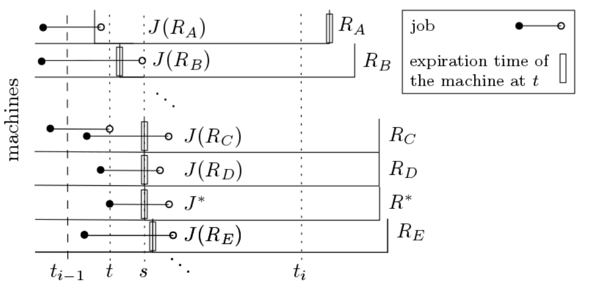

Fig. 10. An ExpireLatest schedule in the non-clairvoyant setting.

charging units required to run the job can be computed at its arrival. As a result, the expiration time of the machine depends on the departure time of the job. Specifically, if a job $[a, d)$ is assigned to a machine M with an open charging unit $[s, s + \tau)$ where $s \le a < s + \tau$, then the expiration time of M after job placement is at least $s + i\tau$ where $i \ge 1$ is the largest integer satisfying $s + (i-1)\tau < d$. Note that by choosing the available machine which expires the latest, the ExpireLatest algorithm implicitly adopts the FitFirst strategy in the clairvoyant setting. In view of these differences, the ExpireLatest algorithm needs to be analyzed separately for the clairvoyant and non-clairvoyant settings.

Theorem 7. *In the non-clairvoyant setting, the ExpireLatest algorithm is* 2*-competitive for* $g = 1$*, and is* $(g + 2)$*-competitive otherwise.*

Proof. Let x_i and z_i be defined on a schedule produced by the ExpireLatest algorithm. By Lemma 2, if there is at least one machine initiated in $(t_{i-1}, t_i]$, we have

$$x_i \le \left\lceil \frac{y_i}{g} \right\rceil. \tag{13}$$

Now, we suppose no machine is initiated in $(t_{i-1}, t_i]$ (see Figure 10 for illustration). Then each charging unit R open at t_i has a defining job $J(R)$. We partition the set of charging units open at t_i into two subsets $\mathcal{S}$ and $\mathcal{T}$ where $\mathcal{S}$ consists of those charging units whose defining jobs arrive before t_{i-1}, and $\mathcal{T}$ consists of those charging units whose defining jobs arrive at or after t_{i-1}. If the set $\mathcal{T}$ is empty, then all charging units open at t_i have their defining jobs arrived before t_{i-1}. Hence, we have

$$x_i \le z_{i-1} \le y_i. \tag{14}$$

On the other hand, if the set $\mathcal{T}$ is non-empty, then let s be the earliest starting time among all charging units in $\mathcal{T}$. We further partition the set $\mathcal{T}$ into three subsets $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ as follows: $\mathcal{T}_1$ consists of those charging units with starting time s and whose machine contains a job crossing t_{i-1} and the machine is never idle from t_{i-1} onwards until at least after s (such as R_C in Figure 10). $\mathcal{T}_2$ consists of those charging units with starting time s that are not in $\mathcal{T}_1$ (such as R_D in Figure 10). $\mathcal{T}_3$ consists of those charging units with starting time after s (such as R_E in Figure 10).

Suppose $\mathcal{T}_2$ is non-empty. Note that by definition, for the machine of each charging unit in $\mathcal{T}_2$, there must be a time $t' \in [t_{i-1}, s]$ where the machine is idle immediately before a job is placed at t'. Since all machines of the charging units in $\mathcal{T}_2$ are non-idle at time s, there must exists a job J^* that arrives in $[t_{i-1}, s]$ such that the assignment of J^* renders all machines of the charging units in $\mathcal{T}_2$ non-idle. In other words, at the arrival time of J^*, prior to its assignment, all other machines of the charging units in $\mathcal{T}_2$ are non-idle except J^*'s machine.

Next, we are going to prove the following claims. Let t be the arrival time of J^* and let R^* be the charging unit open at t_i in J^*'s machine.

Claim 1: The machine of each charging unit in $\mathcal{T}_2 \setminus \{R^*\}$ must be fully occupied at time t.
Claim 2: The machine of each charging unit in $\mathcal{S} \cup \mathcal{T}_1$ has a job active at t.
Claim 3: The machine of each charging unit in $\mathcal{T}_3$ is fully occupied at t.

Claim 1 is true because the ExpireLatest algorithm never assigns a job to an available machine which is idle if there is also an available machine which is non-idle in the event that there is more than one machine with the latest expiration time.

To prove Claim 2, note that since $t \in [t_{i-1}, s]$, the machine of each charging unit in $\mathcal{T}_1$ has a job active at t by definition. Hence, we only need to consider each machine of a charging unit $R \in \mathcal{S}$. If the starting time of R is before t (such as R_A in Figure 10), then R is already open at t and the expiration time of the machine of R is at least t_i. Note that at time t, prior to the start of R^*, the expiration time of the machine of R^* is before t_i. Since the machine of R expires later than the machine of R^* at t, the machine of R must be fully occupied at t, for otherwise, J^* shall be placed on the machine of R according to the ExpireLatest algorithm. On the other hand, if the starting time of R is at least t (such as R_B in Figure 10), then by definition, $J(R)$ is still active at t. As such, regardless of the starting time of R, there is a job active on the machine of R at time t.

To prove Claim 3, we consider each charging unit $R \in \mathcal{T}_3$. By definition, the machine of R expires later than the machine of R^* at time t. Hence, following the same argument as before, the machine of R must be fully occupied at time t.

Suppose $\mathcal{T}_2$ is empty. Then we let J^* be the defining job of any charging unit in $\mathcal{T}_1$. By definition, J^* arrives in $[t_{i-1}, s]$. It is straightforward to verify that Claim 2 and Claim 3 still hold using the same arguments.

To summarize, we have shown that there is an arrival time t of a job J^* placed on the machine of a charging unit R^* where all machines of the charging units in $\mathcal{S} \cup \mathcal{T}_1$ have a job active at t and all machines of the charging units in $\mathcal{T}_2 \cup \mathcal{T}_3 \setminus \{R^*\}$ are fully occupied at t. Suppose there are k charging units in $\mathcal{S} \cup \mathcal{T}_1$. Then, by the definition of $\mathcal{S}$ and $\mathcal{T}_1$, we have

$$k \le z_{i-1}. \tag{15}$$

By taking into account of J^*, we see that there are at least $(x_i - k - 1)g + k + 1$ concurrent jobs at time t. This gives

$$y_i \ge (x_i - k - 1)g + k + 1. \tag{16}$$

From (16), we deduce that $x_i \le y_i$ if $g = 1$, and that $x_i \le \frac{y_i - k - 1}{g} + k + 1$ if $g > 1$. For $g > 1$, since $k \ge 0$ by definition and $k \le z_{i-1}$ by (15), we further derive that $x_i \le \frac{y_i - 1}{g} + z_{i-1} + 1$. Now, since x_i is an integer, we have $x_i \le \lceil \frac{y_i}{g} \rceil +$

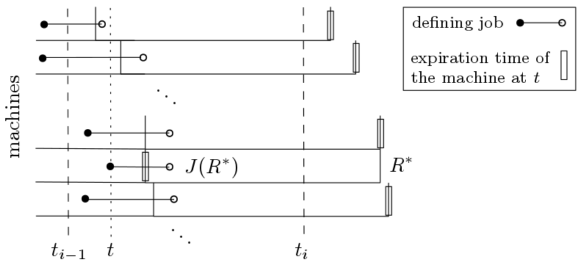

Fig. 11. An ExpireLatest schedule in the clairvoyant setting.

z_{i-1}. Together with (14), we conclude that when no machine is initiated in $(t_{i-1}, t_i]$,

$$x_i \le y_i \text{ if } g = 1,$$

and

$$x_i \le \left\lceil \frac{y_i}{g} \right\rceil + z_{i-1} \text{ otherwise.}$$

Note that the above two relations also hold by following (13) in the case that at least one machine is initiated in $(t_{i-1}, t_i]$. By Lemma 1, we have $\sum_{i\ge1} x_i \le \sum_{i\ge1} y_i \le 2\cdot\text{OPT}$ for $g=1$, and $\sum_{i\ge1} x_i \le \sum_{i\ge1}\lceil \frac{y_i}{g}\rceil + \sum_{i\ge1} z_{i-1} \le (g+2)\cdot\text{OPT}$ for $g>1$. ☐

Theorem 8. *In the clairvoyant setting, the ExpireLatest algorithm is 2-competitive for $g=1$, $\frac{5}{2}$-competitive for $g=2$, and g-competitive for $g>2$.*

Proof. Again, by Lemma 2, if there is a machine initiated in $(t_{i-1}, t_i]$, then we have $x_i \le \lceil \frac{y_i}{g}\rceil$. Let $\mathcal{T}$ be as defined in Theorem 7. If $\mathcal{T}$ is empty, then following the same argument as in Theorem 7, we have $x_i \le z_{i-1}$.

If $\mathcal{T}$ is non-empty, then let R^* be the charging unit in $\mathcal{T}$ whose defining job $J(R^*)$ arrives the latest (see Figure 11 for illustration). Let t be the arrival time of $J(R^*)$. By definition, all other charging units open at t_i have their respective defining jobs arrived earlier than t. Since this is a clairvoyant setting, it is known at the arrival of each defining job that its departure time will extend beyond the expiration time of the ongoing charging unit of the machine. Hence, for any charging unit $R \ne R^*$ open at t_i, after the placement of $J(R)$, the expiration time of the machine of R will be at least the expiration time of R, which is at least t_i. At time t, prior to the start of R^*, the expiration time of the machine of R^* is the starting time of R^*, which is before t_i. Therefore, the machine of R expires later than the machine of R^* at time t. By the definition of the ExpireLatest algorithm, R must be fully occupied at time t, for otherwise, $J(R^*)$ shall be placed on the machine of R.

To summarize, we have established that if there is a machine initiated in $(t_{i-1}, t_i]$, then $x_i \le \lceil \frac{y_i}{g}\rceil$; otherwise, either all defining jobs of the charging units open at t_i cross t_{i-1} which gives $x_i \le z_{i-1}$, or all machines of the charging units open at t_i other than R^* must be fully occupied at time t which gives $y_i \le (x_i-1)g+1$ and hence $x_i \le \lceil \frac{y_i}{g}\rceil$.

In other words, we have shown that for each x_i, either $x_i \le \lceil \frac{y_i}{g}\rceil$ or $x_i \le z_{i-1}$ holds. In the following, we use the relations in Lemma 1 to show that $\sum_{i\ge1} x_i \le 2\cdot\text{OPT}$ for $g=1$ and $\sum_{i\ge1} x_i \le \max\{\frac{5}{2}, g\}\cdot\text{OPT}$ for $g>1$. For $g=1$, since $z_{i-1} \le y_i$ by definition, we always have $x_i \le y_i$ and hence

$$\sum_{i\ge1} x_i \le \sum_{i\ge1} y_i \le 2\cdot\text{OPT}\,.$$

For $g>1$, recall that by assumption, the earliest job arrives after time 0. Hence, any machine open at t_1 is initiated in $(0, t_1]$. As such, we have $x_1 \le \lceil \frac{y_1}{g}\rceil$. We see that $\sum_{i\ge1} x_i$ can be partitioned into a sequence of series $S_{1,m_1}, S_{m_1+1,m_2}, \ldots$, where each $S_{m,n}$ is of the form $S_{m,n} = \sum_{i=m}^{n} x_i$ and there exists a u ($m \le u < n$) such that $x_i \le \lceil \frac{y_i}{g}\rceil$ for each $m \le i \le u$ and $x_i \le z_{i-1}$ for each $u+1 \le i \le n$. As a result, each $S_{m,n}$ satisfies

$$S_{m,n} \le \left\lceil \frac{y_m}{g}\right\rceil + \left\lceil \frac{y_{m+1}}{g}\right\rceil + \cdots + \left\lceil \frac{y_u}{g}\right\rceil + z_u + \cdots + z_{n-1}.$$

Note that $z_u \le x_u$ by definition and $x_u \le \lceil \frac{y_u}{g}\rceil$ by assumption. Using Lemma 1, we have

$$z_u \le \left\lceil \frac{y_u}{g}\right\rceil \le x^*_{u-1} + x^*_u,$$

and

$$z_u \le g x^*_u.$$

This implies that

$$z_u \le \frac{1}{2}x^*_{u-1} + \frac{g+1}{2}x^*_u. \tag{17}$$

Also by Lemma 1, for each $m \le i \le u-1$, we have $\lceil \frac{y_i}{g}\rceil \le x^*_{i-1} + x^*_i$ and for each $u+1 \le i \le n$, we have $z_i \le g x^*_i$. Together with (17), we get

$$S_{m,n} \le x^*_{m-1} + 2x^*_m + \cdots + 2x^*_{u-2} + \frac{5}{2}x^*_{u-1} + \frac{g+3}{2}x^*_u$$
$$+ g x^*_{u+1} + \cdots + g x^*_{n-1}.$$

When $g=2$, we have $\frac{g+3}{2} = \frac{5}{2}$. When $g \ge 3$, we have $\frac{g+3}{2} \le g$. Hence,

$$S_{m,n} \le \max\left\{\frac{5}{2}, g\right\}\cdot \sum_{i=m-1}^{n-1} x^*_i.$$

Thus, by adding all such series $S_{m,n}$ and noting that $x^*_0 = 0$, we get

$$\sum_{i\ge1} x_i \le \max\left\{\frac{5}{2}, g\right\}\cdot\sum_{i\ge1} x^*_i = \max\left\{\frac{5}{2}, g\right\}\cdot\text{OPT}\,.$$

☐

Theorem 8 implies that ExpireLatest is an optimal online algorithm for $g > 2$ in the clairvoyant setting due to the lower bound established in Corollary 2. Moreover, we remark that the competitive ratio for $g = 1$ in Theorems 7 and 8 is tight. Figure 12 shows a tight example. First, a sequence of n jobs arrive and give rise to the initiations of n machines. After the first n jobs depart, another sequence of n jobs arrive and are placed on the machines in the reverse order of their initiations by the ExpireLatest algorithm. Each job leads to the extension of a machine's service, resulting in a total of $2n$ charging units. An optimal schedule, on the other hand, would assign $n-1$ jobs of the second sequence to machines that can fit them and thus need only $n+1$ charging units in total. As n approaches infinity, the ratio between ExpireLatest and the optimal schedule can be made arbitrarily close to 2.

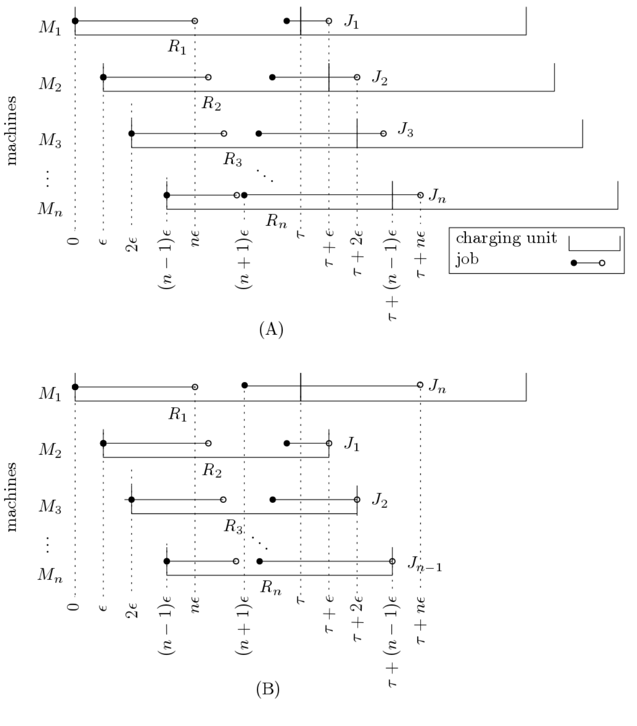

Fig. 12. (A) An ExpireLatest schedule with $g = 1$. (B) An optimal schedule for the instance of (A).

3.3.4 LeastIdle Algorithm

Finally, we introduce a rational algorithm called LeastIdle that can achieve near-optimal competitive ratios in both the non-clairvoyant and clairvoyant settings. The idea of the LeastIdle algorithm is to improve the utilization of open charging units by reducing the idle periods on the machines. The algorithm prefers to place a job on the machine with the least idle period if there are multiple machines available. Ties may be broken arbitrarily. The idle period of an open machine at time t is defined as $t - s$ where $s \le t$ is the latest time when the machine is non-idle. If the machine is non-idle at time t, then the idle period of the machine at t is 0. Since the idle period of a machine is independent of the departure time of the incoming job to be placed, the LeastIdle algorithm produces the same schedule in the non-clairvoyant and clairvoyant settings.

Theorem 9. *The LeastIdle algorithm is 2-competitive for $g = 1$, $\frac{5}{2}$-competitive for $g = 2$, and g-competitive for $g > 2$.*

Proof. Let x_i and z_i be defined on a schedule produced by the LeastIdle algorithm. Consider the set of machines $\mathcal{M}$ that are open at t_i. We define the flag job of a machine to be the earliest arriving job that runs on the machine during the interval $(t_{i-1}, t_i]$. Consider the set of machines $\mathcal{T} \subseteq \mathcal{M}$ whose flag jobs arrive at or after t_{i-1}. If the set $\mathcal{T}$ is non-empty, consider the machine M in this set whose the flag job arrives the latest. Denote the flag job of this machine as J, and its arrival time as t. Then, all other machines in $\mathcal{M} \setminus \{M\}$ have their respective flag jobs arrived before t. Hence, each machine in $\mathcal{M} \setminus \{M\}$ is non-idle at some time between t_{i-1} and t. If machine M is initiated in $(t_{i-1}, t_i]$, by Lemma 2, we have $x_i \le \lceil \frac{y_i}{g} \rceil$. Otherwise, machine M is idle from t_{i-1} onwards. Then, at time t, the idle periods of the machines in $\mathcal{M} \setminus \{M\}$ are less than the idle period of machine M. By the definition of the LeastIdle algorithm, all machines in $\mathcal{M} \setminus \{M\}$ must be fully occupied at time t, for otherwise, J would be placed on one of the machines in $\mathcal{M} \setminus \{M\}$ at time t. Thus, there are at least $(x_i - 1)g + 1$ active jobs at time t. Therefore, $y_i \ge (x_i - 1)g + 1$. Since x_i is an integer, we have $x_i \le \lceil \frac{y_i}{g} \rceil$. On the other hand, if the set $\mathcal{T}$ is empty, then all machines that are open at t_i have their flag jobs arrived before t_{i-1}, which gives $x_i \le z_{i-1}$.

We have shown that for each x_i, either $x_i \le \lceil \frac{y_i}{g} \rceil$ or $x_i \le z_{i-1}$ holds. Now, the theorem can be proved using the same argument as in Theorem 8. ☐

Theorem 9 indicates that LeastIdle is an optimal online algorithm for $g > 2$ in both the non-clairvoyant and clairvoyant settings due to the lower bounds in Corollaries 1 and 2. Moreover, following the instance in Figure 12, the competitive ratio for $g = 1$ in Theorem 9 is tight.

We further consider a special case of ISDCU in which each job has a processing interval no longer than a charging unit. In this case, we show that the LeastIdle algorithm can achieve a constant competitive ratio if ties among non-idle machines are broken with a static ordering of machines.

Theorem 10. *When the processing interval of each job is no longer than τ, the LeastIdle algorithm is 3-competitive if ties among non-idle machines are broken with a static ordering of machines.*

Proof. The main idea for proving this theorem is to show that for each $i \ge 1$, either $x_i \le \lceil \frac{y_{i-1}}{g} \rceil$ or $x_i \le \lceil \frac{y_i}{g} \rceil$ holds. Then, by (1) of Lemma 1, we have $\sum_{i \ge 1} x_i \le 3 \cdot \sum_{i \ge 1} x_i^* = 3 \cdot \mathrm{OPT}$.

Let $\mathcal{M}$ be the set of machines open at t_i. Let $\mathcal{T}$ be the set of machines in $\mathcal{M}$ whose flag jobs arrive at or after t_{i-1}. Following the argument in Theorem 9, when $\mathcal{T} \neq \emptyset$, we have $x_i \le \lceil \frac{y_i}{g} \rceil$. In what follows, we show that $x_i \le \lceil \frac{y_{i-1}}{g} \rceil$ when $\mathcal{T} = \emptyset$.

If $\mathcal{T} = \emptyset$, each machine open at t_i must have its flag job arrived before t_{i-1}. This suggests that all the machines open at t_i are non-idle at t_{i-1} and thus $x_i \le x_{i-1}$. Since LeastIdle is a rational algorithm, by Lemma 2, if any machine in $\mathcal{M}$ is initiated in $(t_{i-2}, t_{i-1}]$, we have $x_i \le x_{i-1} \le \lceil \frac{y_{i-1}}{g} \rceil$.

Now, suppose all the machines in $\mathcal{M}$ are initiated earlier than t_{i-2}. Since the processing interval of each job is no longer than τ, the flag job of each machine in $\mathcal{M}$ must arrive in $(t_{i-2}, t_{i-1}]$. This implies that there is at least one new job placed on each machine in $(t_{i-2}, t_{i-1}]$. If any machine in $\mathcal{M}$ is idle at any time in $(t_{i-2}, t_{i-1}]$, there must exist a job arrival that makes all the machines in $\mathcal{M}$ non-idle, since all the machines in $\mathcal{M}$ are non-idle at t_{i-1}. That is, before this job arrival, there is only one idle machine, and the arriving job is placed on this idle machine to make it non-idle. By the definition of LeastIdle, there are at least $(x_i - 1)g + 1$ active jobs after this job arrival, which implies $x_i \le \lceil \frac{y_{i-1}}{g} \rceil$. Suppose, on the other hand, that each machine in $\mathcal{M}$ is non-idle throughout $(t_{i-2}, t_{i-1}]$. If there is a static ordering of machines (e.g., by the order of their initiations) to break ties among non-idle machines, let's consider the last ordered machine M in $\mathcal{M}$. When the flag job of M is placed, all the machines in $\mathcal{M} \setminus \{M\}$ must be fully occupied.

This suggests that there are at least $(x_i - 1)g + 1$ active jobs at that time, which again indicates $x_i \leq \lceil \frac{y_{i-1}}{g} \rceil$. $\square$

4 Concluding Remarks

Our analysis shows that in the non-clairvoyant setting, the LeastIdle algorithm is optimal for $g = 1$ and $g > 2$ since it matches the lower bounds for any deterministic online algorithm. In the clairvoyant setting, both the ExpireLatest and LeastIdle algorithms are optimal for $g > 2$. On the other hand, our analysis does not appear tight for $g = 2$ in both the non-clairvoyant and clairvoyant settings. There is a gap of at least $\frac{1}{2}$ in these cases. A better analysis may be conducted or a better algorithm may be designed to close these gaps.

For future work, it would be interesting to study an even more general model of ISDCU where the input jobs have different demands for machine capacity (for example by using the dynamic bin packing model in [11], [13], [14]).

Acknowledgments

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 1 under Grant 2018-T1-002-063. The authors would like to thank anonymous reviewers for their valuable suggestions to improve this paper.

References

[1] M. A. Bender, D. P. Bunde, V. J. Leung, S. McCauley, and C. A. Phillips. Efficient scheduling to minimize calibrations. In *Proceedings of the 25th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 280–287. ACM, 2013.

[2] J. Chang, S. Khuller, and K. Mukherjee. LP rounding and combinatorial algorithms for minimizing active and busy time. *Journal of Scheduling*, 20(6):657–680, 2017.

[3] G. Cloud. https://cloud.google.com/.

[4] A. EC2. https://aws.amazon.com/de/ec2/.

[5] J. T. Fineman and B. Sheridan. Scheduling non-unit jobs to minimize calibrations. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 161–170. ACM, 2015.

[6] M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks. Minimizing total busy time in parallel scheduling with application to optical networks. In *Proceedings of IEEE International Symposium on Parallel & Distributed Processing*, pages 1–12. IEEE, 2009.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[8] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.

[9] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir. Real-time scheduling to minimize machine busy times. *Journal of Scheduling*, 18(6):561–573, 2015.

[10] A. W. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54(5):530–543, 2007.

[11] Y. Li, X. Tang, and W. Cai. Dynamic bin packing for on-demand cloud resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 27(1):157–170, 2016.

[12] G. B. Mertzios, M. Shalom, A. Voloshin, P. W. Wong, and S. Zaks. Optimizing busy time on parallel machines. *Theoretical Computer Science*, 562:524–541, 2015.

[13] R. Ren and X. Tang. Clairvoyant dynamic bin packing for job scheduling with minimum server usage time. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 227–237. ACM, 2016.

[14] R. Ren, X. Tang, Y. Li, and W. Cai. Competitiveness of dynamic bin packing for online cloud server allocation. *IEEE/ACM Transactions on Networking*, 25(3):1324–1331, 2017.

[15] B. Saha. Renting a cloud. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 24. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[16] M. Shalom, A. Voloshin, P. W. Wong, F. C. Yung, and S. Zaks. Online optimization of busy time on parallel machines. *Theoretical Computer Science*, 560:190–206, 2014.

[17] P. Winkler and L. Zhang. Wavelength assignment and generalized interval graph coloring. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 830–831. Society for Industrial and Applied Mathematics, 2003.

PLACE PHOTO HERE

Ming Ming Tan received her BSc and PhD in Mathematics from Nanyang Technological University, Singapore. She is currently a research fellow in the School of Computer Science and Engineering at Nanyang Technological University.

PLACE PHOTO HERE

Runtian Ren received the BSc degree in mathematics and applied mathematics from University of Science and Technology of China in 2014. He is currently a PhD student in the School of Computer Science and Engineering at Nanyang Technological University, Singapore.

PLACE PHOTO HERE

Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University in 1998, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an associate professor in the School of Computer Science and Engineering at Nanyang Technological University, Singapore. His research interests include distributed systems, cloud computing, mobile and pervasive computing, and wireless sensor networks. He has served as an associate editor of IEEE Transactions on Parallel and Distributed Systems, and a program co-chair of IEEE ICPADS 2012 and CloudCom 2014. He is a senior member of the IEEE.