Interval Job Scheduling with Machine Launch Cost

Runtian Ren, Yuqing Zhu, Chuanyou Li and Xueyan Tang

Abstract—We study an interval job scheduling problem in distributed systems. We are given a set of interval jobs, with each job specified by a size, an arrival time and a processing length. Once a job arrives, it must be placed on a machine immediately and run for a period of its processing length without interruption. The homogeneous machines to run jobs have the same capacity limits such that at any time, the total size of the jobs running on any machine cannot exceed its capacity. Launching each machine incurs a fixed cost. After launch, a machine is charged a constant cost per time unit until it is terminated. The problem targets to minimize the total cost incurred by the machines for processing the given set of interval jobs. We focus on the algorithmic aspects of the problem in this paper. For the special case where all the jobs have a unit size equal to the machine capacity, we propose an optimal offline algorithm and an optimal 2-competitive online algorithm. For the general case where jobs can have arbitrary sizes, we establish a non-trivial lower bound on the optimal solution. Based on this lower bound, we propose a 5-approximation algorithm in the offline setting. In the non-clairvoyant online setting, we design a $O(\mu)$-competitive Modified First-Fit algorithm which is near optimal (μ is the max/min job processing length ratio). In the clairvoyant online setting, we propose an asymptotically optimal $O(\sqrt{\log \mu})$-competitive algorithm based on our Modified First-Fit strategy.

Index Terms—Job Scheduling; Online Algorithm; Approximation Algorithm.

1 Introduction

In this paper, we study an *Interval Job Scheduling with Machine Launch Cost* problem (abbreviated as ISL) defined as follows. The input is a set of interval jobs, with each job specified by a size and an active interval delimited by its arrival and departure times. Homogeneous machines of uniform capacity are available to process jobs.[1] Each job must be placed onto a machine to run over its active interval without interruption. At any time, the total size of the active jobs placed on any machine cannot exceed the machine capacity. Suppose a machine M is launched at time t_1 and terminated at time t_2 for running a subset of the jobs. Then, the cost for using this machine is $C + (t_2 - t_1)$, where a cost C is incurred for launching M at time t_1 and a constant cost rate 1 is incurred for running M throughout the period $[t_1, t_2)$. The target of ISL is to minimize the total cost incurred for processing all the jobs.

Our ISL problem can model several practical scenarios in energy-efficient computing and cloud computing.

Energy-efficient computing: Energy management in computing servers of data centers is a key issue faced by industries. It is observed that up to 50% budget of a data center is invested on the electricity costs consumed by the servers, and about 1.5% of the total electricity worldwide is used by data centers nowadays [29]. Energy-efficient scheduling focuses on minimizing the energy consumed by a cluster of servers for supporting service level agreements [30]. A service level agreement is a commitment between service providers and customers, which includes service parameters such as the demand for computing resource and the usage period. Generally, the request of a customer can be modeled by a job specified by a size (e.g., the number of virtual machines to rent) and an active interval (i.e., a fixed period for renting the computing resources). For processing jobs, a typical computing server's energy consumption model is as follows. On one hand, according to [16], [24], [27], when a server is "on", its power usage rate at any time is a constant, regardless of the total size of the jobs running concurrently on the server; when a server is "off", its power usage rate is 0. On the other hand, powering a server up and down normally incur inevitable energy overheads. Let C_{up} and C_{down} denote these energy overheads. Since each server used must be powered down eventually, we can assume alternatively an initial energy overhead $C = C_{up} + C_{down}$ for powering a server up and no energy overhead for powering a server down. Without loss of generality, by assuming that the power usage rate is 1 when a server is "on", the energy consumption of a server for being "on" for a duration l becomes $C + l$. Note that a server can serve several requests concurrently as long as their aggregate size does not exceed the server's capacity. In this way, how to assign the customer requests onto servers such that the total energy consumption is minimized can be modeled by our ISL problem.

Cloud computing: A cloud service customer may rent virtual machines from cloud service providers for processing jobs and the rental is charged according to "pay-as-you-go" billing [1]. To launch (terminate) a virtual machine, there

- *Runtian Ren, Yuqing Zhu and Xueyan Tang are with School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: {RENR0002, yuqing002, asxytang}@ntu.edu.sg.*
- *Chuanyou Li is with School of Computer Science and Engineering, Southeast University, Nanjing, China 210096. E-mail: cyli@seu.edu.cn.*

1. The capacity of a machine refers to its capacity of computing resources such as CPU cores or time shares. The size of a job refers to its demand for computing resources such as CPU cores or time shares.

is generally a constant duration of setup[2] (shutdown) until the virtual machine is ready to process (turned off), which is also charged to the customer. In other words, each virtual machine used has a fixed launch cost, and the total cost paid by the customer includes both launch and running costs. Note that a virtual machine can run multiple jobs simultaneously as long as the computing resources required by the jobs do not exceed the virtual machine's capacity. Thus, a good strategy for renting virtual machines and allocating jobs onto the virtual machines can be useful for the customers to save their costs. Our ISL problem can be used to model such an issue as well.

In this paper, we focus on the algorithmic aspects of ISL and study the problem in the offline, non-clairvoyant online and clairvoyant online settings. In the offline setting, the information of all the jobs is known before the scheduling process. In the online settings, each job must be placed onto a machine when it arrives without any information of the jobs arriving in the future. In the non-clairvoyant online setting, the departure time of a job is not known at its arrival and thus cannot be used for the scheduling purpose. In the clairvoyant online setting, the departure time of a job is known at its arrival and can be used for the scheduling purpose.

As shall be elaborated in Section 3, our ISL problem generalizes the MinUsageTime Dynamic Bin Packing (DBP) problem [21]. By the results of existing studies [6], [21], we can easily conclude the following:

1. ISL is NP-hard.
2. For ISL in the non-clairvoyant online setting, the competitiveness of any deterministic online algorithm has a lower bound of $\Omega(\mu)$.
3. For ISL in the clairvoyant online setting, the competitiveness of any deterministic online algorithm has a lower bound of $\Omega(\sqrt{\log \mu})$.

Here, μ is the max/min length ratio among the active intervals of all the jobs to schedule. Consequently, the following three questions arise naturally:

1. Does there exist a $O(1)$-approximation algorithm for ISL in the offline setting?
2. Does there exist a $O(\mu)$-competitive algorithm for ISL in the non-clairvoyant online setting?
3. Does there exist a $O(\sqrt{\log \mu})$-competitive algorithm for ISL in the clairvoyant online setting?

We give an affirmative answer to each question in this paper.

Our main ideas on developing offline and online algorithms for ISL are as follows. First, we consider a special case where each job to be scheduled has a unit size equal to the machine capacity (abbreviated as ISL-Unit). We propose an optimal offline algorithm and an optimal 2-competitive online algorithm to solve ISL-Unit. Then, we consider ISL in the general case where each job can have an arbitrary size. Based on the strategies designed for ISL-Unit, an intuitive method to deal with a general ISL instance is to schedule the large jobs $\mathcal{J}^l$ (of size larger than half the machine capacity) and the small jobs $\mathcal{J}^s$ (of size at most half the machine capacity) separately. This is because no two large jobs active at the same time can be placed onto the same machine, which suggests that $\mathcal{J}^l$ can be seen as an ISL-Unit instance. Consequently, by adopting the ISL-Unit strategies for $\mathcal{J}^l$, the total cost incurred for scheduling $\mathcal{J}^l$ in the offline setting is no more than the optimal cost for scheduling $\mathcal{J}$; the total cost incurred for scheduling $\mathcal{J}^l$ in the online setting is no more than twice the optimal cost for scheduling $\mathcal{J}$. Then, we only need to focus on scheduling $\mathcal{J}^s$. We establish a non-trivial lower bound on the optimal cost of any ISL instance. Based on this lower bound and the previous works on MinUsageTime DBP [6], [25], [26], we propose a $O(1)$-approximation algorithm, a $O(\mu)$-competitive non-clairvoyant algorithm and a $O(\sqrt{\log \mu})$-competitive clairvoyant algorithm to schedule $\mathcal{J}^s$ in the three settings.

The rest of the paper is organized as follows. In Section 2, we introduce some notations and definitions. In Section 4, we present our optimal offline and online algorithms for ISL-Unit. In Section 5, we introduce the lower bound on the optimal cost of any ISL instance. In Sections 6, 7, 8, we study ISL in the offline, non-clairvoyant online and clairvoyant online settings respectively. Finally, in Section 9, we briefly discuss how these results can be improved in the case of uniform job sizes and introduce some future works.

2 Notations and Definitions

We introduce some key notations used in this paper. For any time interval I, we use I^- and I^+ to denote the left and right endpoints of I respectively. For technical reasons, we shall view intervals as half-open, i.e., $I = [I^-, I^+)$. Let $\mathrm{len}(I) = I^+ - I^-$ denote the length of interval I.

As introduced in Section 1, the input to ISL is a set of interval jobs $\mathcal{J}$ and each job $J \in \mathcal{J}$ is specified by a size $s(J)$ and an active interval $I(J) = [I(J)^-, I(J)^+)$. $I(J)^-$, $I(J)^+$ and $\mathrm{len}(I(J))$ are known as J's arrival time, departure time and processing length respectively. Given a set of jobs $\mathcal{J}$, we use $s(\mathcal{J}, t)$ to denote the total size of the jobs active at time t, i.e., $s(\mathcal{J}, t) = \sum_{J \in \mathcal{J}: t \in I(J)} s(J)$. Without loss of generality, we assume that all the machines have the same capacity 1 and each job has a size no larger than 1. Then, at any time, the total size of the active jobs placed on any machine cannot exceed 1.

To facilitate reasoning, we shall assume that there is a sufficiently large pool of machines and that once a machine used is terminated, the machine is never launched again. Such an assumption makes sense for our problem since after a machine used is terminated, it becomes indistinguishable from any unused machine. If a machine M is launched at time t_1 and terminated at time t_2, we say that M is in the "on" state during the interval $[t_1, t_2)$ and refer to the interval as the "on" interval of M. The cost for using this machine M is $C + (t_2 - t_1)$, where C is the cost for launching M, and $t_2 - t_1$ is the cost for keeping M "on" during $[t_1, t_2)$. Note that during its "on" interval, M may not always be processing jobs. When at least one job is running on M, M is said to be *busy*; when M is "on" but no job is running on it, M is said to be *idle*. If M is busy (idle) throughout an interval I, then I is referred to as a *busy (idle) interval* of M.

The target of ISL is to minimize the total cost incurred by the machines for processing all the jobs. We propose several offline and online algorithms and study their approximation and competitive ratios [7], [31], i.e., the worst-case ratio

2. Empirical studies show that the virtual machine start up time in the cloud is about several minutes [23].

between a solution constructed by the algorithm and an optimal offline solution.

3 Related Work

Our ISL problem generalizes several problems studied in recent years. Two representative problems are *the interval job scheduling problem with bounded parallelism* (ISBP) [3], [8], [9], [11], [18], [19], [24], [27], [32] and *the MinUsageTime Dynamic Bin Packing (DBP) problem* [6], [14], [16], [21], [22], [25], [26], [28]. Both problems target to minimize the total usage time of the machines for processing a set of interval jobs. They can be seen as special cases of our ISL problem by assuming no initial cost for launching a machine to process jobs (i.e., $C=0$). Our ISL problem considers a more general and practical model for the energy-efficient scheduling and cloud computing issues.

In the ISBP problem, all the jobs have the same size equal to a fraction $\frac{1}{g}$ of the machine capacity, such that a machine can run at most g ($g \geq 2$) jobs concurrently at any time. Winker and Zhang [32] first proved its NP-hardness through a reduction from the Circular Arc Coloring problem. In the offline setting, Alicherry and Bhatia [3] proposed a 2-approximation algorithm through a network flow formulation. Kumar and Rudra [19] proposed another 2-approximation algorithm based on the 2-allocation technique introduced by Gergov [12]. Flammini *et al.* [11] presented a greedy First-Fit algorithm which gives a 4-approximation. Chang *et al.* [9] proposed another 3-approximation algorithm called GreedyTracking. In the clairvoyant online setting where the departure time of a job is revealed when it arrives, Shalom *et al.* [27] established a tight bound g on the competitiveness of ISBP and also studied several special cases where better competitiveness can be achieved.

The MinUsageTime DBP problem generalizes the ISBP problem by allowing each job to have an arbitrary size. In the offline setting, Khandekar *et al.* [16] first proposed a 5-approximation algorithm. Later, Ren and Tang [25] gave a 4-approximation Dual Coloring algorithm by extending the algorithm of Kumar and Rudra [19]. In the non-clairvoyant online setting where the departure time of a job is not known when it arrives, Li *et al.* [21] established a lower bound of μ on the competitiveness of any deterministic online algorithm, where μ is the max/min length ratio among the active intervals of all the jobs to schedule. Ren *et al.* [26] showed that the First Fit packing algorithm achieves a competitive ratio of $\mu+3$, which is near optimal. In the clairvoyant online setting, Azar and Vainstein [6] established a lower bound $\Omega(\sqrt{\log \mu})$ on the competitiveness and proposed a matching $O(\sqrt{\log \mu})$-competitive algorithm.

Azar *et al.* [5] modeled the setup time for booting a virtual machine and developed bi-objective algorithms to minimize the maximum delay of job completion and the total dollar expenditure of job processing in the cloud. In contrast, we study a unified cost metric that combines the costs for machines to launch and process jobs. While the study of [5] was limited to unit-size jobs, each with a demand equal to the machine capacity, we consider the general case where each job can have an arbitrary demand for machine capacity.

Many previous works studied the energy conservation problems for a limited number of machines with the energy cost of power-up operations taken into consideration. Different from such cases, our ISL problem assumes that an unlimited number of machines are provided for running jobs. The rationale is that a data center is generally equipped with sufficiently large number of servers for cloud service renting, and a customer can also rent an arbitrary number of virtual machines at the same time for processing jobs.

For the energy conservation problem with a single machine, a classical optimization problem is to design the power-down strategy to minimize the energy consumed during the machine's idle periods when there is no computation demand. If the machine has an active state and a sleep state, such a problem is equivalent to the ski-rental problem, a famous rent-or-buy problem. Karlin *et al.* [15] proposed an optimal 2-competitive deterministic online algorithm and an improved $\frac{e}{e-1}$-competitive randomized online algorithm. Irani *et al.* [13] and Augustine *et al.* [4] studied the power-down strategies on a single machine with an active state and several low-power sleep states.

For the energy conservation problem with multiple machines, Albers [2] studied an offline problem to minimize the total energy consumption for a group of heterogeneous machines to satisfy the varying total demand for computing capacity over time, but the study ignored specific jobs. Khuller *et al.* [17], [20] introduced the machine activation problems, where an activation budget is given for activating machines. The target is to minimize the makespan for a batch of jobs on a set of heterogeneous machines. Different from [17], [20], we focus on minimizing the total launch and running cost for processing interval jobs.

4 ISL-Unit

We start by considering a special case of ISL where each job has a size 1 which is equal to the machine capacity. In this case, each machine can accommodate at most one job at any time. We refer to this special case as ISL-Unit. We propose an optimal algorithm in the offline setting and an optimal 2-competitive algorithm in the online settings based on the following unified strategy.

Our algorithms work by scheduling jobs in the chronological order of their arrivals. When a job arrives, if there exist one or multiple machines whose last job was completed less than C time units ago, then the incoming job is placed on the machine whose last job was completed the latest. Otherwise, a new machine is launched to run the incoming job. We remark that in our problem definition, "C" is defined as the launch cost of a machine. The "C" in the "C time units" of our scheduling algorithms refers to the same scalar value C as in the problem definition. That is, the running cost of a machine for C time units is the same as the cost to launch a machine, since we assume the running cost of a machine per time unit is 1.

In the offline setting, each machine is terminated when it completes its last job. In the online setting, whenever a machine completes a job and becomes idle, it is kept "on" for C time units for receiving new incoming jobs. If no new incoming job is placed on it, the machine is then terminated.

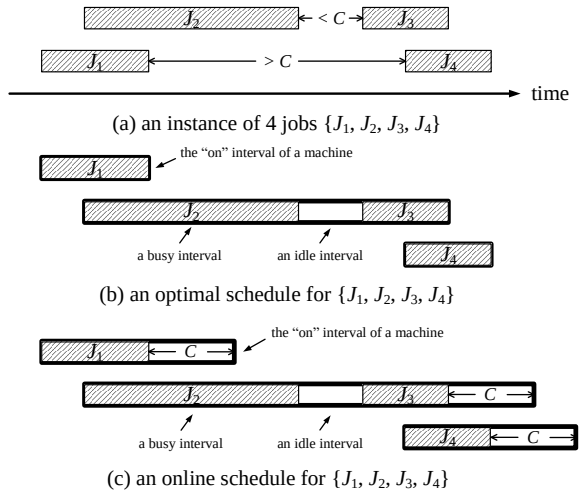


Fig. 1. Example schedules of an ISL-Unit instance with four jobs.

Examples of the offline and online schedules constructed are shown in Figure 1.

We first prove the optimality of the offline algorithm by converting any optimal schedule A for a set of jobs $\mathcal{J}$ into the schedule produced by our algorithm without incurring any extra cost during the conversion process. We index and examine the jobs in their arrival order. Assume that job J_j is the first job that is scheduled differently by the optimal schedule A and our algorithm. Since the jobs $\{J_1, J_2, \ldots, J_{j-1}\}$ are scheduled identically by schedule A and our algorithm, before job J_j arrives, the set of machines used by schedule A is identical to the set of machines used by our algorithm. Now, suppose schedule A places job J_j on a machine M', while our algorithm places job J_j on another machine M. We can convert schedule A to a new schedule that places J_j on M without increasing the cost incurred. Specifically, let $\mathcal{J}_M$ and $\mathcal{J}_{M'}$ denote the sets of jobs placed on machines M and M' respectively by schedule A. We swap the assignments of jobs from J_j onward between machines M and M':

1. Place jobs $\mathcal{J}_{M'} \cap \{J_1, J_2, \ldots, J_{j-1}\}$ and $\mathcal{J}_M - \{J_1, J_2, \ldots, J_{j-1}\}$ on machine M'.
2. Place jobs $\mathcal{J}_M \cap \{J_1, J_2, \ldots, J_{j-1}\}$ and $\mathcal{J}_{M'} - \{J_1, J_2, \ldots, J_{j-1}\}$ on machine M. In this way, job J_j is placed on machine M.

Since our schedule is feasible, the jobs in $\mathcal{J}_M \cap \{J_1, J_2, \ldots, J_{j-1}\}$ do not overlap with J_j and hence all the jobs in $\mathcal{J}_{M'} - \{J_1, J_2, \ldots, J_{j-1}\}$. Similarly, since schedule A is feasible, the jobs in $\mathcal{J}_{M'} \cap \{J_1, J_2, \ldots, J_{j-1}\}$ do not overlap with J_j and hence all the jobs in $\mathcal{J}_M - \{J_1, J_2, \ldots, J_{j-1}\}$. Therefore, the new schedule after the above conversion is feasible. It is easy to infer that the total length of M and M''s "on" intervals remains unchanged after conversion. Thus, no extra cost is incurred due to the conversion. So, we have obtained a new schedule in which jobs $\{J_1, J_2, \ldots, J_j\}$ are scheduled identically to our algorithm. By performing the similar conversions for every job in $\mathcal{J} - \{J_1, J_2, \ldots, J_j\}$, we can obtain an optimal schedule exactly the same as the schedule produced by our algorithm. Thus, we can conclude that our offline algorithm is optimal for solving ISL-Unit.

Next, we study the competitiveness of our online algorithm. In the online algorithm, if a machine is terminated at time t, it must be "on" and idle during the time interval $[t-C, t)$. Thus, each machine incurs a cost that is C higher than that in the offline algorithm. Note that the cost incurred by each machine is at least C in the offline algorithm since it needs to be launched initially. Therefore, our online algorithm is 2-competitive. We remark that our online algorithm does not make use of the processing length information of a job in scheduling it. Thus, it works for both the non-clairvoyant setting (where the processing length is not known at a job's arrival) and the clairvoyant setting (where the processing length is known at a job's arrival).

Finally, we construct an instance to show that no deterministic online algorithm can achieve a competitive ratio less than 2 for ISL-Unit. In our instance, we release at most n jobs, each with a processing length ε. First, a job J_1 is released at time 0. A new machine must be launched to run this job and this machine becomes idle at time ε. Obviously, an optimal schedule for this single job is to terminate the machine once J_1 is completed. Suppose an online algorithm terminates this machine at time t_1. If $t_1 \geq C$, we stop job releasing. Then, the cost ratio between the online algorithm and the optimal schedule is $\frac{C+t_1}{C+\varepsilon} \geq \frac{2C}{C+\varepsilon}$. Otherwise, if $t_1 < C$, we continue to release a new job J_2. In general, a job J_i ($i \geq 2$) is released at time $\sum_{j=1}^{i-1}(t_j + \frac{1}{2^j}\delta)$, after the machine used for J_{i-1} is terminated. Here, t_j denotes the "on" time of the machine for processing J_j and $\delta > 0$. Thus, a new machine must be launched to run J_i. Suppose the online algorithm terminates the new machine at time $\sum_{j=1}^{i-1}(t_j + \frac{1}{2^j}\delta) + t_i$. If $t_i \geq C$, we stop job releasing. The best strategy for scheduling jobs $\{J_1, \ldots, J_i\}$ is to place them on one machine and terminate this machine at time $\sum_{j=1}^{i-1}(t_j + \frac{1}{2^j}\delta) + \varepsilon$. So, the cost ratio between the online algorithm and the optimal schedule is $\frac{\sum_{j=1}^{i-1}(C+t_j)+(C+t_i)}{C+\sum_{j=1}^{i-1}(t_j+\frac{1}{2^j}\delta)+\varepsilon} > \frac{(i+1)C+\sum_{j=1}^{i-1}t_j}{C+\sum_{j=1}^{i-1}t_j+\delta+\varepsilon} > \frac{2i\cdot C}{i\cdot C+\delta+\varepsilon} > \frac{2C}{C+\delta+\varepsilon}$. Otherwise, job releasing continues. After n jobs are released, the cost of the online algorithm is at least $\sum_{j=1}^{n-1}(C+t_j) + (C+\varepsilon) = n \cdot C + \sum_{j=1}^{n-1} t_j + \varepsilon$, whereas the cost of the optimal schedule is $C + \sum_{j=1}^{n-1}(t_j + \frac{1}{2^j}\delta) + \varepsilon < C + \sum_{j=1}^{n-1} t_j + \delta + \varepsilon$. Thus, the cost ratio is at least $\frac{n\cdot C+\sum_{j=1}^{n-1}t_j+\varepsilon}{C+\sum_{j=1}^{n-1}t_j+\delta+\varepsilon} > \frac{(2n-1)\cdot C+\varepsilon}{n\cdot C+\delta+\varepsilon}$. Therefore, the competitive ratio by applying any deterministic online algorithm is at least

$$\min\Big\{\frac{2C}{C+\varepsilon}, \frac{2C}{C+\delta+\varepsilon}, \frac{(2n-1)\cdot C+\varepsilon}{n\cdot C+\delta+\varepsilon}\Big\}.$$

As ε and δ approach 0 and n goes towards infinity, the above bound can be made arbitrarily close to 2. Hence, our online algorithm is optimal for ISL-Unit.

5 A Lower Bound for ISL

Now, we study the ISL problem in the general case where jobs can have arbitrary sizes. We first investigate the lower bound on the optimal cost of any ISL instance, which shall be used later to design and analyze offline and online scheduling algorithms. We establish a lower bound by relaxing the requirement of ISL. Specifically, we consider a

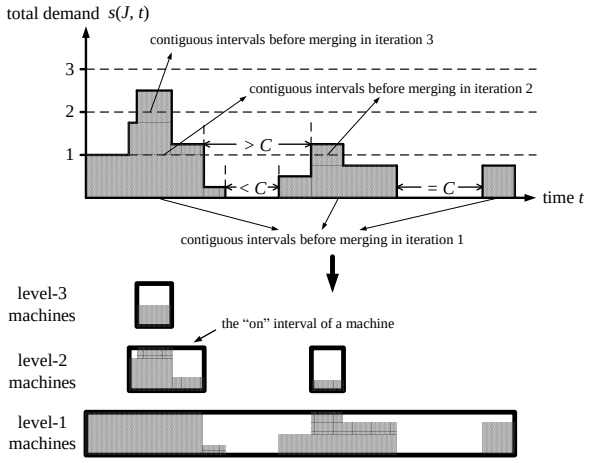


Fig. 2. Example of an optimal RISL schedule: the upper diagram illustrates the total size of the active jobs as a function of time, and the lower diagram shows the corresponding optimal RISL schedule.

Relaxed ISL (abbreviated as RISL) problem. Given a set of jobs $\mathcal{J}$, the objective of RISL is to find a minimum-cost schedule of machines such that at any time t, the aggregate capacity of "on" machines is at least $s(\mathcal{J},t)$, i.e., the total size of the jobs active at time t. Note that RISL does not impose the feasibility of actually placing the jobs on the "on" machines. Since each job has to be placed on a single machine throughout its active interval, a feasible placement may not always exist even if the aggregate capacity of "on" machines can meet the total demand $s(\mathcal{J},t)$ at all times. Thus, RISL is a relaxation of ISL and an optimal RISL schedule can be used to bound the optimal cost of the ISL problem.

We compute an optimal RISL schedule in an iterative manner. In each iteration k, we first determine all the contiguous intervals in which $s(\mathcal{J},t) > k-1$. Next, we merge every two consecutive intervals I_a and I_b into one interval if they are no further than C apart, i.e., we replace I_a and I_b with $[I_a^-, I_b^+)$ if $I_b^- - I_a^+ \le C$. This process is repeated recursively until no more intervals can be merged. Then, the resultant intervals are all further than C apart from each other. For each resultant interval I, we schedule a new machine with an "on" interval I. All the machines used in iteration k, denoted by $\mathcal{M}_k^{\mathrm{RISL}}$, are referred to as the machines of level k. It is easy to see that $\max_t\lceil s(\mathcal{J},t)\rceil$ iterations are needed by the scheduling process. The schedule produced is given by $\mathcal{M}^{\mathrm{RISL}} = \bigcup_{k=1}^{\max_t\lceil s(\mathcal{J},t)\rceil} \mathcal{M}_k^{\mathrm{RISL}}$. In this schedule, at any time t, there is a machine with an "on" interval covering t at each of the levels 1, 2, ..., $\lceil s(\mathcal{J},t)\rceil$. Thus, the schedule meets the capacity requirement of RISL. Figure 2 gives an example to illustrate the construction of an optimal RISL schedule.

Theorem 1. Our algorithm above produces an optimal RISL schedule that minimizes the total cost incurred by the machines used.

Proof: We prove it by converting any optimal RISL schedule for a set of jobs $\mathcal{J}$ into the schedule $\mathcal{M}^{\mathrm{RISL}}$ produced by our algorithm without incurring any extra cost.

Let $\mathcal{M}^{opt}$ denote the set of machines used by an optimal schedule for $\mathcal{J}$. Let $Q(M)$ denote the "on" interval of a machine M. Let $N(\mathcal{M}^{opt},t) = |\{M : M \in \mathcal{M}^{opt}, t \in Q(M)\}|$ denote the number of "on" machines at time t in the schedule of $\mathcal{M}^{opt}$. For each $k \in \{1,\ldots,\max_t\lceil s(\mathcal{J},t)\rceil\}$, we examine the times t when $N(\mathcal{M}^{opt},t) \ge k$. These times may constitute one or more contiguous intervals that are apart from each other. For each interval I, we construct a machine with an "on" interval I. All the machines constructed form a set $\mathcal{M}_k$. Then, we create a new schedule $\mathcal{M}^*$ by putting $\mathcal{M}_1$, $\mathcal{M}_2$, ..., $\mathcal{M}_{\max_t\lceil s(\mathcal{J},t)\rceil}$ together, i.e., $\mathcal{M}^* = \bigcup_{k=1}^{\max_t\lceil s(\mathcal{J},t)\rceil} \mathcal{M}_k$. It is easy to verify that the schedule $\mathcal{M}^*$ meets the capacity requirement of RISL and the total "on" time of the machines $\mathcal{M}^*$ cannot exceed that of the machines $\mathcal{M}^{opt}$. Moreover, every time a new machine in $\mathcal{M}^*$ is launched, the number of "on" machines in $\mathcal{M}^{opt}$ must have been increased, so there must also be a machine launched in $\mathcal{M}^{opt}$. Thus, $|\mathcal{M}^*| \le |\mathcal{M}^{opt}|$. As a result, the total cost of $\mathcal{M}^*$ is bounded by that of $\mathcal{M}^{opt}$, which suggests that $\mathcal{M}^*$ is also an optimal RISL schedule for $\mathcal{J}$.

Next, we prove that the machines in $\mathcal{M}^*$ have the same "on" intervals as the machines $\mathcal{M}^{\mathrm{RISL}}$ produced by our algorithm. For each $k \in \{1,\ldots,\max_t\lceil s(\mathcal{J},t)\rceil\}$, consider the machine set $\mathcal{M}_k \subseteq \mathcal{M}^*$ and the level-k machines $\mathcal{M}_k^{\mathrm{RISL}}$ produced by our algorithm. We sort the machines in $\mathcal{M}_k$ and $\mathcal{M}_k^{\mathrm{RISL}}$ in increasing order of their launch times, i.e., $\mathcal{M}_k = \{M_{k,1}, M_{k,2}, \ldots\}$ and $\mathcal{M}_k^{\mathrm{RISL}} = \{M_{k,1}^{\mathrm{RISL}}, M_{k,2}^{\mathrm{RISL}}, \ldots\}$. We first compare machine $M_{k,1}$ with machine $M_{k,1}^{\mathrm{RISL}}$. By our algorithm, the total demand $s(\mathcal{J},t)$ must be greater than $k-1$ at time $Q(M_{k,1}^{\mathrm{RISL}})^-$. Since $\mathcal{M}^{opt}$ meets the capacity requirement of RISL, we have $N(\mathcal{M}^{opt}, Q(M_{k,1}^{\mathrm{RISL}})^-) \ge k$, which suggests that machine $M_{k,1}$ is launched no later than $Q(M_{k,1}^{\mathrm{RISL}})^-$. Now, suppose $Q(M_{k,1})^- < Q(M_{k,1}^{\mathrm{RISL}})^-$. Then, according to our algorithm, the total demand $s(\mathcal{J},t)$ is at most $k-1$ during the interval $[Q(M_{k,1})^-, Q(M_{k,1}^{\mathrm{RISL}})^-)$. We can thus delay the launch of $M_{k,1}$ to reduce its cost, which contradicts the optimality of $\mathcal{M}^*$. Therefore, we have $Q(M_{k,1})^- = Q(M_{k,1}^{\mathrm{RISL}})^-$.

Now we prove that $Q(M_{k,1})^+ = Q(M_{k,1}^{\mathrm{RISL}})^+$. If $Q(M_{k,1})^+ > Q(M_{k,1}^{\mathrm{RISL}})^+$, the total demand $s(\mathcal{J},t)$ must exceed $k-1$ immediately before time $Q(M_{k,1})^+$. Otherwise, $M_{k,1}$ can be terminated earlier to save cost, which contradicts the optimality of $\mathcal{M}^*$. Since the total demand $s(\mathcal{J},t)$ exceeds $k-1$ immediately before $Q(M_{k,1})^+$, $Q(M_{k,1})^+$ must fall inside or on the right endpoint of the "on" interval of some machine in $\mathcal{M}_k^{\mathrm{RISL}}$. Suppose that this machine is $M_{k,j}^{\mathrm{RISL}}$ where $j > 1$. Note that by our algorithm, the total demand $s(\mathcal{J},t)$ is bounded by $k-1$ during the interval $[Q(M_{k,j-1}^{\mathrm{RISL}})^+, Q(M_{k,j}^{\mathrm{RISL}})^-)$ and $Q(M_{k,j}^{\mathrm{RISL}})^- - Q(M_{k,j-1}^{\mathrm{RISL}})^+ > C$. We can thus save cost by terminating $M_{k,1}$ earlier at time $Q(M_{k,j-1}^{\mathrm{RISL}})^+$ and launching a new machine from time $Q(M_{k,j}^{\mathrm{RISL}})^-$ to $Q(M_{k,1})^+$ while keeping the schedule $\mathcal{M}^*$ meeting the capacity requirement. This leads to a contradiction to the optimality of $\mathcal{M}^*$. Similarly, if $Q(M_{k,1})^+ < Q(M_{k,1}^{\mathrm{RISL}})^+$, machine $M_{k,2}$ must be launched before time $Q(M_{k,1}^{\mathrm{RISL}})^+$ as the total demand $s(\mathcal{J},t)$ exceeds $k-1$ immediately before time $Q(M_{k,1}^{\mathrm{RISL}})^+$. Since $\mathcal{M}^*$ meets the capacity requirement, the total demand $s(\mathcal{J},t)$ must be bounded by $k-1$ during the interval $[Q(M_{k,1})^+, Q(M_{k,2})^-)$. Since $[Q(M_{k,1})^+, Q(M_{k,2})^-) \subseteq Q(M_{k,1}^{\mathrm{RISL}})$, according to our algorithm, we have $Q(M_{k,2})^- - Q(M_{k,1})^+ \le C$. This

implies that machines $M_{k,1}$ and $M_{k,2}$ can be replaced by a machine with an "on" interval $[Q(M_{k,1})^-, Q(M_{k,2})^+)$ to save cost, which again contradicts the optimality of $\mathcal{M}^*$.[3] Therefore, we have $Q(M_{k,1})^+ = Q(M_{k,1}^{\mathrm{RISL}})^+$ and hence $Q(M_{k,1}) = Q(M_{k,1}^{\mathrm{RISL}})$.

Given $Q(M_{k,1}) = Q(M_{k,1}^{\mathrm{RISL}})$, by similar arguments, we can show that $Q(M_{k,2}) = Q(M_{k,2}^{\mathrm{RISL}})$. Inductively, it can be shown that $Q(M_{k,i}) = Q(M_{k,i}^{\mathrm{RISL}})$ for any i. Therefore, for each k, the machines in $\mathcal{M}_k$ and $\mathcal{M}_k^{\mathrm{RISL}}$ have the same "on" intervals, which implies that the schedules $\mathcal{M}^*$ and $\mathcal{M}^{\mathrm{RISL}}$ have the same costs. □

6 Offline Setting

We first consider ISL in the offline setting and propose a 5-approximation algorithm. All the jobs are first classified into two sets $\mathcal{J}^l$ and $\mathcal{J}^s$ according to their sizes, where $\mathcal{J}^l$ ($\mathcal{J}^s$) includes all the large (small) jobs with size larger (no larger) than $\frac{1}{2}$. The large jobs and the small jobs are scheduled separately. We apply our optimal offline algorithm proposed for ISL-Unit (introduced in Section 4) to schedule all the large jobs, since any two large jobs cannot run concurrently on the same machine. The small jobs are scheduled by the following steps. First, a demand chart is built to represent the total size of the active small jobs as a function of time (similar to the top diagram of Figure 2). The horizontal dimension of the demand chart represents the time. At any time t, the height of the demand chart is given by $s(\mathcal{J}^s, t)$. Then, all the small jobs are placed inside the demand chart such that no three jobs overlap together in their placement. Next, the jobs are assigned to the machines by slicing the demand chart. Finally, the machine schedules are derived by a similar algorithm to that for solving the RISL problem (introduced in Section 5). Figure 3 gives an example to illustrate the scheduling of small jobs. For ease of reference, Algorithm 1 shows the details of scheduling the small jobs $\mathcal{J}^s$.

The placement and slicing steps basically follow the Dual Coloring algorithm proposed in our earlier work for the MinUsageTime DBP problem [25]. In the placement step, each small job J is represented by a rectangle spanning its active interval $I(J)$ in the time dimension and having a height of its size $s(J)$ in the demand dimension (see Figure 3(b)). To place jobs, we examine a collection of altitudes from the top to the bottom in the demand chart. We gradually color the area of the demand chart as jobs are placed. A red color indicates that there is no overlap among jobs in the area, while a blue color indicates that jobs are likely to overlap in the area. Initially, the collection of altitudes to examine includes all the ceiling altitudes $s(\mathcal{J}^s, t)$ in the demand chart. When examining an altitude h, we divide the horizontal line at altitude h into three sets of intervals: red, blue and uncolored. For each uncolored interval I_u, we look for a yet-to-place job J whose active interval $I(J)$ intersects with I_u but does not intersect with any other uncolored interval and any red interval. If there is no such job, all the area below I_u in the demand chart is colored blue to allow for overlaps. If such a job J exists, J is placed at altitude h. The uncolored area covered by J is then colored red to prevent overlaps. The altitude of J's lower boundary, i.e., $h - s(J)$, is added as an altitude to examine. The examination of an altitude h proceeds until there is no more uncolored interval. The placement step completes when all the altitudes are examined. Upon completion, it can be shown that all the jobs are placed in the demand chart and no area of the demand chart is covered by three or more jobs simultaneously [25].

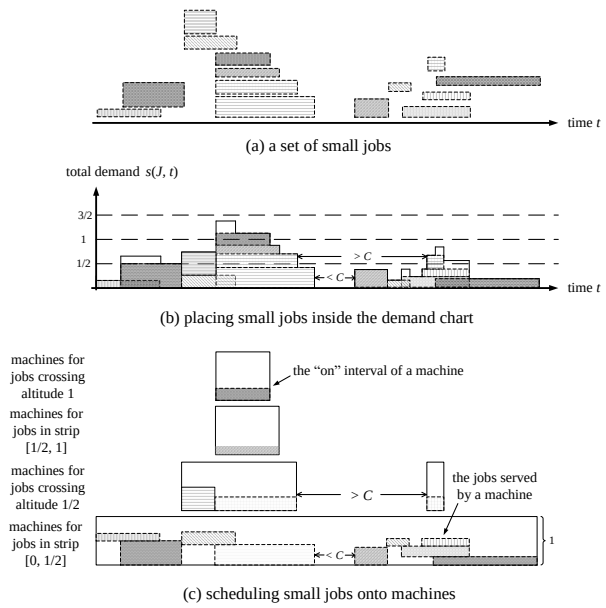


Fig. 3. An example for scheduling small jobs in the offline setting: the top diagram shows the set of small jobs to schedule, the middle diagram shows the job placement in the demand chart, and the bottom diagram shows the corresponding job assignments and machine schedules.

In the slicing step, the demand chart is sliced into horizontal strips of height $\frac{1}{2}$ each. Thanks to the overlap limit, all the small jobs placed completely within each strip have a total size at most 1 at any time and thus can be assigned to run on one machine. On the other hand, since the size of each small job is bounded by $\frac{1}{2}$, a job can span at most 2 strips in the placement. For each pair of neighboring strips, due to the overlap limit, at most two small jobs can cross them concurrently at any time, so their total size is bounded by 1. Therefore, all the jobs placed across a pair of neighboring strips can be assigned to run on one machine. Note that the number of strips produced by the slicing step is at most $\max_t \lceil \frac{s(\mathcal{J}^s,t)}{1/2} \rceil \leq 2 \cdot \max_t \lceil s(\mathcal{J}^s, t) \rceil$. Based on the above analysis, all the small jobs can be divided into at most $2 \cdot \max_t \lceil s(\mathcal{J}^s,t) \rceil + 2 \cdot \max_t \lceil s(\mathcal{J}^s,t) \rceil - 1 = 4 \cdot \max_t \lceil s(\mathcal{J}^s,t) \rceil - 1$ disjoint groups such that the jobs in each group have a total size bounded by 1 at any time. Finally, we can apply the optimal RISL algorithm in Section 5 to each group of small jobs to produce an optimal RISL schedule in just one iteration (see Figure 3(c)).

Now we prove that our algorithm achieves an approximation ratio of 5. Let $\mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$ denote the optimal ISL cost for processing all the jobs $\mathcal{J}$ and let $A(\mathcal{J})$ denote the total cost incurred by applying our algorithm to schedule

3. If $Q(M_{k,2})^- - Q(M_{k,1})^+ = C$, the replacement does not change the cost but it extends the termination time of $M_{k,1}$ to $Q(M_{k,2})^+$. We can repeat the comparison of the new $Q(M_{k,1})^+$ with $Q(M_{k,1}^{\mathrm{RISL}})^+$ until either the optimality of $\mathcal{M}^*$ is violated or $Q(M_{k,1})^+ = Q(M_{k,1}^{\mathrm{RISL}})^+$.

ALGORITHM 1: Scheduling Small Jobs for Offline ISL

Input: $\mathcal{J}^s = \{J \in \mathcal{J} : s(J) \le \frac{1}{2}\}$.
Output: A schedule for all the small jobs $\mathcal{J}^s$.
—Place the small jobs inside the demand chart—
$\mathcal{A}ltitudes = \{s(\mathcal{J}^s, t) : t \in \bigcup_{J \in \mathcal{J}^s} I(J)\}$;
while $\mathcal{A}ltitudes \neq \emptyset$ **do**
  Determine $h = \max_{h' \in \mathcal{A}ltitudes} h'$;
  Let $\mathcal{R}ed$, $\mathcal{B}lue$ and $\mathcal{U}ncolored$ be the respective sets of all maximal red, blue and uncolored intervals at altitude h;
  while $\mathcal{U}ncolored \neq \emptyset$ **do**
    Pick an uncolored interval $I_u \in \mathcal{U}ncolored$;
    if $\exists\ J \in \mathcal{J}^s$ *such that* $I(J) \cap I_u \neq \emptyset$ *and* $\forall\ I \in \mathcal{U}ncolored \cup \mathcal{R}ed \setminus \{I_u\}, I(J) \cap I = \emptyset$ **then**
      Place job J at altitude h and delete J from $\mathcal{J}^s$;
      Color the rectangle $\{I(J) \cap I_u\} \times (h - s(J), h]$ red;
      Delete I_u from $\mathcal{U}ncolored$;
      Add $[I_u^-, I(J)^-)$ and $[I(J)^+, I_u^+)$ into $\mathcal{U}ncolored$ if they are not empty;
      Add $h - s(J)$ into $\mathcal{A}ltitudes$ if $h > s(J)$ and $h - s(J) \notin \mathcal{A}ltitudes$;
    else
      Color the rectangle $I_u \times (0, h]$ blue;
      Delete I_u from $\mathcal{U}ncolored$;
    end
  end
  Delete h from $\mathcal{A}ltitudes$;
end
—Slice the demand chart and schedule small jobs—
Slice the demand chart into horizontal strips of height $\frac{1}{2}$ each;
for *each strip* **do**
  Schedule all the small jobs placed completely within the strip by applying the optimal RISL algorithm;
end
for *every two neighboring strips* **do**
  Schedule all the small jobs placed across the two strips by applying the optimal RISL algorithm;
end

$\mathcal{J}$. We have $A(\mathcal{J}) = A(\mathcal{J}^l) + A(\mathcal{J}^s)$. Since no two large jobs can run simultaneously on the same machine, by applying the optimal ISL-Unit algorithm, we have $A(\mathcal{J}^l) = \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^l) \le \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$. Next, we bound $A(\mathcal{J}^s)$. If we apply the optimal RISL algorithm in Section 5 to schedule $\mathcal{J}^s$, the machines used would consist of $\max_t \lceil s(\mathcal{J}^s, t) \rceil$ disjoint sets $\mathcal{M}_1^{\mathrm{RISL}}, \mathcal{M}_2^{\mathrm{RISL}}, \ldots$ where each set $\mathcal{M}_k^{\mathrm{RISL}}$ denotes the machines of level k. Let $\mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s)$ denote the cost of the optimal RISL schedule for $\mathcal{J}^s$ and let $E(\mathcal{M}_k^{\mathrm{RISL}})$ denote the total cost of the machines in $\mathcal{M}_k^{\mathrm{RISL}}$. Then, $\mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s) = \sum_{k=1}^{\max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}})$. Note that the machines $\mathcal{M}_k^{\mathrm{RISL}}$ actually serve the job demands between altitudes $k-1$ and k in the demand chart.

On the other hand, by applying our approximation algorithm, all the machines used to process the small jobs can be divided into at most $4 \cdot \max_t \lceil s(\mathcal{J}^s, t) \rceil - 1$ disjoint sets, among which there is one set for processing the jobs placed completely within each strip of the demand chart and one set for processing the jobs placed across each pair of neighboring strips. By the construction of the demand chart, the set of machines for the jobs placed in the strip from altitude $(h\text{–}1) \cdot \frac{1}{2}$ to $h \cdot \frac{1}{2}$ only needs to be "on" when the total size of active small jobs exceeds $(h-1) \cdot \frac{1}{2} \ge \lceil \frac{h}{2} \rceil - 1$. Thus, the "on" intervals of these machines must be completely contained in the "on" intervals of the machines $\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}$. This is true before as well as after the "on" intervals are merged by the optimal RISL algorithm. Similarly, the set of machines for the jobs placed across the strips from altitude $(h\text{–}2) \cdot \frac{1}{2}$ to $(h\text{–}1) \cdot \frac{1}{2}$ and from $(h\text{–}1) \cdot \frac{1}{2}$ to $h \cdot \frac{1}{2}$ only needs to be "on" when the total size of active small jobs exceeds $(h-1) \cdot \frac{1}{2} \ge \lceil \frac{h}{2} \rceil - 1$. So, they must also have their "on" intervals fully contained in the "on" intervals of the machines $\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}$. Therefore, the total cost incurred for processing $\mathcal{J}^s$ is bounded by

$$\begin{aligned}
&\sum_{h=1}^{2 \cdot \max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}) + \sum_{h=2}^{2 \cdot \max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}) \\
&= 2 \cdot \sum_{k=1}^{\max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}}) + E(\mathcal{M}_1^{\mathrm{RISL}}) + 2 \cdot \sum_{k=2}^{\max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}}) \\
&< 4 \cdot \sum_{k=1}^{\max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}}) \\
&= 4 \cdot \mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s).
\end{aligned}$$

Recall that the RISL problem is a relaxation of the ISL problem. It follows that $\mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s) \le \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^s) \le \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$. Thus, $A(\mathcal{J}^s) \le 4 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$. As a result, $A(\mathcal{J}) = A(\mathcal{J}^l) + A(\mathcal{J}^s) \le 5 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$.

Theorem 2. Our algorithm above is a 5-approximation algorithm for offline ISL.

Remark. In our offline algorithm, the large jobs and small jobs are scheduled separately. An intuitive alternative is to schedule the large jobs and small jobs together. However, if we schedule all the jobs together to deal with an ISL instance, the approximation ratio derived for the offline strategy would be larger than 5. Specifically, if we schedule all the jobs together, using the same placement step of Algorithm 1, all the jobs are placed inside the demand chart such that no three jobs overlap each other in their placement. Then in the slicing step of Algorithm 1, all the jobs placed completely within each strip of height $\frac{1}{2}$ can still be assigned to one machine since their total size is at most 1 at any time. However, we shall need up to two machines for processing the jobs crossing two neighboring strips. This is because there can be two jobs crossing the strips at the same time point and each job can have a size more than $\frac{1}{2}$. Since the number of strips is at most $2 \cdot \max_t \lceil s(\mathcal{J}, t) \rceil$, all the jobs are divided into at most $2 \cdot \max_t \lceil s(\mathcal{J}, t) \rceil + 2 \cdot (2 \cdot \max_t \lceil s(\mathcal{J}, t) \rceil - 1) = 6 \cdot \max_t \lceil s(\mathcal{J}, t) \rceil - 2$ disjoint groups with the total size of the jobs in each group bounded by 1 at any time. By the same arguments above, the set of machines for the jobs placed in the strip from altitude $(h-1) \cdot \frac{1}{2}$ to $h \cdot \frac{1}{2}$ must have their "on" intervals fully contained in the "on" intervals of the machines $\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}$, and so does the set of machines for each group of jobs placed across the strips from altitude $(h\text{–}2) \cdot \frac{1}{2}$ to $(h\text{–}1) \cdot \frac{1}{2}$ and from $(h\text{–}1) \cdot \frac{1}{2}$ to $h \cdot \frac{1}{2}$. Therefore, the total cost incurred for processing all the jobs $\mathcal{J}$ is bounded by

$$\begin{aligned}
&\sum_{h=1}^{2 \cdot \max_t \lceil s(\mathcal{J}, t) \rceil} E(\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}) + \sum_{h=2}^{2 \cdot \max_t \lceil s(\mathcal{J}, t) \rceil} 2 \cdot E(\mathcal{M}_{\lceil h/2 \rceil}^{\mathrm{RISL}}) \\
&= 2 \cdot \sum_{k=1}^{\max_t \lceil s(\mathcal{J}, t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}}) + 2 \cdot E(\mathcal{M}_1^{\mathrm{RISL}}) + 4 \cdot \sum_{k=2}^{\max_t \lceil s(\mathcal{J}, t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}})
\end{aligned}$$

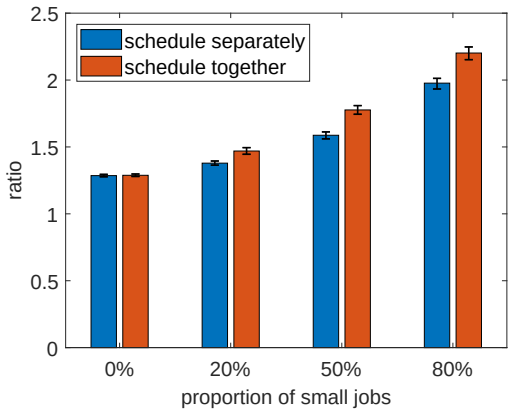
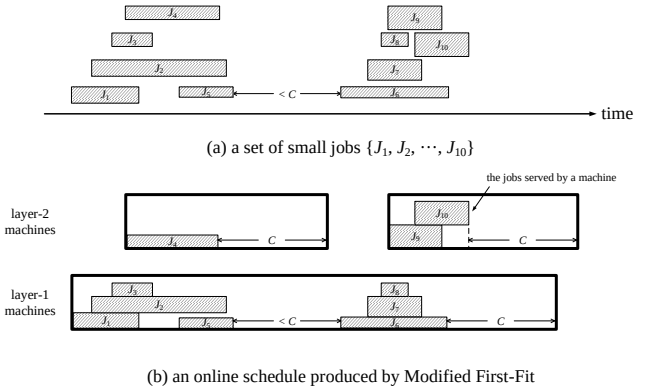


Fig. 4. Simulation results for the offline algorithms over 100 job sequences: the bars represent the average ratio between the total cost of a scheduling algorithm and the lower bound of the optimal cost; the lines over the bars represent the 90th and 10th percentile ratios.



(b) an online schedule produced by Modified First-Fit

Fig. 5. An example for scheduling small jobs by Modified First-Fit: the upper diagram shows the set of small jobs to schedule, and the lower diagram shows the corresponding job assignments and machine schedules produced by Modified First Fit.

$$
\begin{aligned}
&< 6 \cdot \sum_{k=1}^{\max_t \lceil s(\mathcal{J},t) \rceil} E(\mathcal{M}_k^{\mathrm{RISL}}) \\
&= 6 \cdot \mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s) \\
&\leq 6 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}).
\end{aligned}
$$

So, the algorithm is a 6-approximation algorithm.

We also conduct simulations to compare the strategies of scheduling large and small jobs together and separately using randomly generated job sequences. We assume that the jobs arrive following a Poisson process with a mean inter-arrival time of 1 time unit. A portion x of the jobs are small jobs and have sizes randomly generated from a uniform distribution $(0, \frac{1}{2}]$. The remaining portion $(1-x)$ of the jobs are large jobs and have sizes randomly generated from a uniform distribution $(\frac{1}{2}, 1]$. The processing lengths of all the jobs are randomly generated from a uniform distribution $[10, 100]$ time units. The launch cost of a machine is set to 1. For each setting of x, we randomly generate 100 job sequences. Each sequence contains 1000 jobs. We apply the scheduling algorithms to the sequence and compute the total cost for processing all the jobs. We also derive a lower bound on the optimal cost of the sequence as described in Section 5. We compute the ratio between the total cost of a scheduling algorithm and the lower bound for each sequence, and use the average ratio over the 100 job sequences as a performance measure. Figure 4 shows the average ratio together with the 90th and 10th percentile ratios for different proportions of small jobs. As can be seen, when there is a mix of large and small jobs, scheduling all the jobs together produces higher ratios than scheduling large and small jobs separately. This confirms that scheduling large and small jobs together is less efficient than our approach to schedule them separately. Moreover, Figure 4 also shows that the ratios produced by our offline algorithm are well below the derived approximation ratio of 5 for various settings. This demonstrates the practical effectiveness of our algorithm in the settings tested.

7 Non-clairvoyant Setting

Now we study ISL in the non-clairvoyant online setting, where each job must be placed on a machine at its arrival without knowing its departure time and any future job arrivals. We propose a Modified First-Fit algorithm and prove that it is $(\mu+9)$-competitive. Again, the Modified First-Fit algorithm schedules the large jobs $\mathcal{J}^l$ (of size larger than $\frac{1}{2}$) and the small jobs $\mathcal{J}^s$ (of size at most $\frac{1}{2}$) separately. The large jobs are scheduled by applying the 2-competitive online algorithm for ISL-Unit (introduced in Section 4), while the small jobs are scheduled based on the idea of First Fit. Specifically, each machine used for processing small jobs is given a layer label. The machines given the label k are referred to as the layer-k machines. At any time, at most one machine of each layer can be in the "on" state. To schedule an incoming job J, we find the lowest-indexed layer k such that either (1) no layer-k machine is in the "on" state or (2) the "on" machine of layer k has enough capacity available to accommodate job J. In the former case, a new layer-k machine is launched to run job J. In the latter case, the incoming job J is assigned to the "on" machine of layer k. Whenever an "on" machine for small jobs becomes idle, the machine waits for at most C time units for receiving new incoming jobs. If no incoming job is placed on it after C time units, the machine is terminated. Figure 5 shows an example of how Modified First-Fit schedules small jobs.

Let $\mathrm{MFF}(\mathcal{J})$ denote the total cost for processing all the jobs $\mathcal{J}$ by applying our Modified First-Fit algorithm. Then, we have $\mathrm{MFF}(\mathcal{J}) = \mathrm{MFF}(\mathcal{J}^l) + \mathrm{MFF}(\mathcal{J}^s)$. Since any two large jobs cannot run concurrently on the same machine, by applying the 2-competitive algorithm for ISL-Unit, we have $\mathrm{MFF}(\mathcal{J}^l) \leq 2 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^l) \leq 2 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$. To bound $\mathrm{MFF}(\mathcal{J}^s)$, we split the cost incurred for processing all the small jobs $\mathcal{J}^s$ into two parts: the cost when the machines are busy (referred to as the busy cost) and the cost for launching the machines and when they are idle (referred to as the auxiliary cost). We show that the busy cost of all the machines used for $\mathcal{J}^s$ is no more than $(\mu+3) \cdot \int_{\bigcup_{J \in \mathcal{J}^s} I(J)} \lceil s(\mathcal{J}^s, t) \rceil \mathrm{d}t$. Since the number of "on" machines for processing the small jobs $\mathcal{J}^s$ must be at least $\lceil s(\mathcal{J}^s, t) \rceil$ at any time t, $\int_{\bigcup_{J \in \mathcal{J}^s} I(J)} \lceil s(\mathcal{J}^s, t) \rceil \mathrm{d}t$ is a lower bound on the total "on" time and hence a lower bound on the optimal ISL cost for $\mathcal{J}^s$, i.e., $\mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^s)$. We also prove that the auxiliary cost of all the machines used for $\mathcal{J}^s$ is no more than four times the optimal RISL

cost for $\mathcal{J}^s$, i.e., $\mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s)$. In this way, $\mathrm{MFF}(\mathcal{J}^s)$ is bounded by $(\mu+3)\cdot\mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^s)+4\cdot\mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s)\le(\mu+7)\cdot\mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^s)\le(\mu+7)\cdot\mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$. Combining $\mathrm{MFF}(\mathcal{J}^l)$ and $\mathrm{MFF}(\mathcal{J}^s)$, we can then conclude that the Modified First-Fit algorithm is $(\mu+9)$-competitive.

Bounding the busy cost of machines used for $\mathcal{J}^s$: To bound the busy cost for small jobs, we apply a charging technique in a similar spirit to that in [16], [26]. Suppose the machines used for small jobs consist of m layers: $\mathcal{M}_1^{\mathrm{MFFs}},\mathcal{M}_2^{\mathrm{MFFs}},\ldots,\mathcal{M}_m^{\mathrm{MFFs}}$. It is easy to bound the total busy time of the layer-1 machines $\mathcal{M}_1^{\mathrm{MFFs}}$ by $\int_{\bigcup_{J\in\mathcal{J}^s}I(J)}\lceil s(\mathcal{J}^s,t)\rceil\,\mathrm{d}t$, since at most one machine in $\mathcal{M}_1^{\mathrm{MFFs}}$ is busy at any moment when there are small jobs active.

Next we bound the total busy time of the layer-k machines ($k\ge2$). Let $\mathcal{J}_k^{\mathrm{MFFs}}$ be the set of small jobs placed on the layer-k machines. By the Modified First-Fit algorithm, the busy time of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ is given by $\mathrm{len}\big(\bigcup_{J\in\mathcal{J}_k^{\mathrm{MFFs}}}I(J)\big)$. Let $\mathcal{J}_k$ be a maximal subset of $\mathcal{J}_k^{\mathrm{MFFs}}$ by deleting any job $J\in\mathcal{J}_k^{\mathrm{MFFs}}$ whose active interval is fully contained in another job's active interval. Then, by sorting the jobs in $\mathcal{J}_k$ in the order of their arrival times, i.e., $\mathcal{J}_k=\{J_{k,1},J_{k,2},\ldots,J_{k,z_k}\}$ where $I(J_{k,1})^-<I(J_{k,2})^-<\ldots<I(J_{k,z_k})^-$ and $z_k=|\mathcal{J}_k|$, we have $I(J_{k,1})^+<I(J_{k,2})^+<\ldots<I(J_{k,z_k})^+$. Besides, we also have $\bigcup_{J\in\mathcal{J}_k^{\mathrm{MFFs}}}I(J)=\bigcup_{J\in\mathcal{J}_k}I(J)$ after reducing $\mathcal{J}_k^{\mathrm{MFFs}}$ to $\mathcal{J}_k$.

We split $\bigcup_{J\in\mathcal{J}_k}I(J)$ into z_k disjoint intervals. For each job $J_{k,i}\in\mathcal{J}_k$ ($i\in\{1,\ldots,z_k-1\}$), define

$$X(J_{k,i})=\big[I(J_{k,i})^-,\min\{I(J_{k,i})^+,I(J_{k,i+1})^-\}\big).$$

For the job $J_{k,z_k}\in\mathcal{J}_k$, define $X(J_{k,z_k})=I(J_{k,z_k})$. Then, the total busy time of the layer-k machines can be written as

$$\mathrm{len}\Big(\bigcup_{J\in\mathcal{J}_k^{\mathrm{MFFs}}}I(J)\Big)=\mathrm{len}\Big(\bigcup_{J\in\mathcal{J}_k}I(J)\Big)=\sum_{J\in\mathcal{J}_k}\mathrm{len}(X(J)).$$

Now, we define

$$d_k=\sum_{J\in\mathcal{J}_k}s(J)\cdot\mathrm{len}(X(J)).$$

It is easy to see that

$$d_k\le\sum_{J\in\mathcal{J}_k}s(J)\cdot\mathrm{len}(I(J))\le\sum_{J\in\mathcal{J}_k^{\mathrm{MFFs}}}s(J)\cdot\mathrm{len}(I(J)).\quad(1)$$

Recall that each job J in $\mathcal{J}_k$ is placed on a machine in $\mathcal{M}_k^{\mathrm{MFFs}}$. According to the Modified First-Fit algorithm, when job J arrives at time $X(J)^-=I(J)^-$, there must exist a layer-$(k-1)$ machine $M\in\mathcal{M}_{k-1}^{\mathrm{MFFs}}$ which is busy and does not have enough capacity available to accommodate job J. Therefore, at time $X(J)^-$, the total size of the active jobs on machine M must be larger than $1-s(J)$. Let $\mathcal{P}(J)$ denote these jobs. Then, we have $s(J)+\sum_{\hat{J}\in\mathcal{P}(J)}s(\hat{J})>1$. Define

$$d_k^*=\sum_{J\in\mathcal{J}_k}\Big(\sum_{\hat{J}\in\mathcal{P}(J)}s(\hat{J})\cdot\mathrm{len}(X(J))\Big).$$

Then, the total busy time of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ can be bounded by

$$\sum_{J\in\mathcal{J}_k}\mathrm{len}(X(J))<\sum_{J\in\mathcal{J}_k}\Big(\big(s(J)+\sum_{\hat{J}\in\mathcal{P}(J)}s(\hat{J})\big)\cdot\mathrm{len}(X(J))\Big)$$
$$=d_k+d_k^*.\quad(2)$$

We next show that $d_k^*\le(\mu+1)\cdot\sum_{J\in\mathcal{J}_{k-1}^{\mathrm{MFFs}}}s(J)\cdot\mathrm{len}(I(J))$. Define $\widehat{\mathcal{J}_k}=\bigcup_{J\in\mathcal{J}_k}\mathcal{P}(J)$. Obviously, $\widehat{\mathcal{J}_k}\subseteq\mathcal{J}_{k-1}^{\mathrm{MFFs}}$. For each job $\hat{J}\in\widehat{\mathcal{J}_k}$, let $\mathcal{P}^{-1}(\hat{J})$ denote the set of all the jobs J in $\mathcal{J}_k$ such that $\hat{J}\in\mathcal{P}(J)$. Note that the jobs in $\mathcal{P}^{-1}(\hat{J})$ may come from different machines in $\mathcal{M}_k^{\mathrm{MFFs}}$. For each job $J\in\mathcal{P}^{-1}(\hat{J})$, since job $\hat{J}$ has already been placed on a machine in $\mathcal{M}_{k-1}^{\mathrm{MFFs}}$ when J arrives, $\hat{J}$ cannot arrive later than J. Thus,

$$X(J)^-=I(J)^-\ge I(\hat{J})^-.\quad(3)$$

By definition, $I(\hat{J})$ covers time $X(J)^-$, i.e., $X(J)^-<I(\hat{J})^+$. Recall that the max/min job processing length ratio is μ, which suggests that $\mathrm{len}(X(J))\le\mathrm{len}(I(J))\le\mu\cdot\mathrm{len}(I(\hat{J}))$. Therefore,

$$X(J)^+=X(J)^-+\mathrm{len}(X(J))<I(\hat{J})^+\mu\cdot\mathrm{len}(I(\hat{J})).\quad(4)$$

Recall that $X(J)$ of all the jobs J in $\mathcal{P}^{-1}(\hat{J})\subseteq\mathcal{J}_k$ are disjoint. It follows from (3) and (4) that

$$\begin{aligned}\sum_{J\in\mathcal{P}^{-1}(\hat{J})}\mathrm{len}(X(J))&\le\max_{J\in\mathcal{P}^{-1}(\hat{J})}X(J)^+-\min_{J\in\mathcal{P}^{-1}(\hat{J})}X(J)^-\\&\le I(\hat{J})^+\mu\cdot\mathrm{len}(I(\hat{J}))-I(\hat{J})^-\\&=(\mu+1)\cdot\mathrm{len}(I(\hat{J})).\end{aligned}$$

As a result, we can rewrite d_k^* as

$$\begin{aligned}d_k^*&=\sum_{\hat{J}\in\widehat{\mathcal{J}_k}}s(\hat{J})\cdot\Big(\sum_{J\in\mathcal{P}^{-1}(\hat{J})}\mathrm{len}(X(J))\Big)\\&\le\sum_{\hat{J}\in\widehat{\mathcal{J}_k}}\Big(s(\hat{J})\cdot(\mu+1)\cdot\mathrm{len}(I(\hat{J}))\Big)\\&\le(\mu+1)\cdot\sum_{J\in\mathcal{J}_{k-1}^{\mathrm{MFFs}}}s(J)\cdot\mathrm{len}(I(J)).\end{aligned}$$

By (1) and (2), we can bound the total busy time of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ by

$$d_k+d_k^*\le\sum_{J\in\mathcal{J}_k^{\mathrm{MFFs}}}s(J)\cdot\mathrm{len}(I(J))+(\mu+1)\cdot\sum_{J\in\mathcal{J}_{k-1}^{\mathrm{MFFs}}}s(J)\cdot\mathrm{len}(I(J))$$

and thus bound the total busy time of the machines $\bigcup_{k=2}^m\mathcal{M}_k^{\mathrm{MFFs}}$ by

$$\begin{aligned}\sum_{k=2}^m\Big(d_k+d_k^*\Big)&\le\sum_{J\in\mathcal{J}^s}s(J)\cdot\mathrm{len}(I(J))\\&\quad+(\mu+1)\cdot\sum_{J\in\mathcal{J}^s}s(J)\cdot\mathrm{len}(I(J))\\&=(\mu+2)\cdot\sum_{J\in\mathcal{J}^s}s(J)\cdot\mathrm{len}(I(J))\\&\le(\mu+2)\cdot\int_{\bigcup_{J\in\mathcal{J}^s}I(J)}\lceil s(\mathcal{J}^s,t)\rceil\,\mathrm{d}t.\end{aligned}$$

Combining with the busy time of the machines $\mathcal{M}_1^{\mathrm{MFFs}}$, we can bound the total busy cost for processing all the small jobs by $(\mu+3)\cdot\int_{\bigcup_{J\in\mathcal{J}^s}I(J)}\lceil s(\mathcal{J}^s,t)\rceil\mathrm{d}t$.

Bounding the auxiliary cost of machines used for $\mathcal{J}^s$: We first analyze the auxiliary cost incurred by the machines of a given layer k. Recall that at most one machine of each layer can be "on" at any time. Thus, the busy intervals of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ must not overlap with each other.

Let $I_{k,1}, I_{k,2}, \ldots, I_{k,n_k}$ denote all the busy intervals of the machines $\mathcal{M}_k^{\text{MFFs}}$ in the chronological order. Then, a cost C must be incurred at time $I_{k,1}^-$ when the first machine of layer k is launched. According to our Modified First-Fit algorithm, for each $i \in \{1, \ldots, n_k - 1\}$, if $I_{k,i+1}^- - I_{k,i}^+ \leq C$, then the two consecutive busy intervals $I_{k,i}$ and $I_{k,i+1}$ must be associated with the same machine of layer k and this machine must be idle during the interval $[I_{k,i}^+, I_{k,i+1}^-)$. Consequently, a cost of $I_{k,i+1}^- - I_{k,i}^+$ is incurred over the idle interval. Otherwise, if $I_{k,i+1}^- - I_{k,i}^+ > C$, a machine of layer k must be terminated at time $I_{k,i}^+ + C$ and a new machine of layer k must be launched at time $I_{k,i+1}^-$. This suggests that a total cost of $2 \cdot C$ is incurred by these two machines during the interval $[I_{k,i}^+, I_{k,i+1}^-)$. Finally, note that the last machine of layer k must be idle during the interval $[I_{k,n_k}^+, I_{k,n_k}^+ + C)$ which incurs a cost of C. Therefore, the total auxiliary cost of the machines $\mathcal{M}_k^{\text{MFFs}}$ is at most $2 \cdot \big(C + \sum_{i=1}^{n_k - 1} \min\{I_{k,i+1}^- - I_{k,i}^+, C\}\big)$. In the following, we show that $C + \sum_{i=1}^{n_k - 1} \min\{I_{k,i+1}^- - I_{k,i}^+, C\}$ is no more than the cost incurred by the machines of level $\lceil k/2 \rceil$ in the optimal RISL schedule for $\mathcal{J}^s$. In this way, the auxiliary cost of all the machines for $\mathcal{J}^s$ by applying our Modified First-Fit algorithm can be bounded by $4 \cdot \text{OPT}_{\text{RISL}}(\mathcal{J}^s)$.

First, we show that if a machine of layer k is used by our Modified First-Fit algorithm, then at least $\lceil \frac{k}{2} \rceil$ levels of machines are needed in the optimal RISL schedule for $\mathcal{J}^s$, i.e., level $\lceil \frac{k}{2} \rceil$ exists. In fact, if a small job J is placed on a machine in $\mathcal{M}_k^{\text{MFFs}}$, then for each layer lower than k, there must exist a machine which is in the "on" state at J's arrival time $I(J)^-$ but does not have enough capacity available to accommodate J. This implies that the "on" machine of each layer lower than k must have more than $1 - s(J)$ capacity occupied at time $I(J)^-$. Consequently, the total size of the small jobs active at time $I(J)^-$ must be larger than $(k-1)(1-s(J)) + s(J) = k - 1 - (k-2) \cdot s(J) \geq \frac{k}{2}$, where the inequality follows from the fact that $s(J) \leq \frac{1}{2}$. Therefore, in the optimal RISL schedule for $\mathcal{J}^s$, at least $\lceil \frac{k}{2} \rceil$ machines need to be "on" to satisfy the job demands at time $I(J)^-$. As a result, there are at least $\lceil \frac{k}{2} \rceil$ levels of machines in the optimal RISL schedule for $\mathcal{J}^s$.

Let $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$ denote the machines of level $\lceil \frac{k}{2} \rceil$ in the optimal RISL schedule for $\mathcal{J}^s$. The above analysis suggests that a machine in $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$ must be in the "on" state at the arrival of every small job placed on a machine of layer k by our Modified First-Fit algorithm. It is obvious that there is a small job arriving at the beginning of each busy interval $I_{k,i}$ of the machines in $\mathcal{M}_k^{\text{MFFs}}$. Thus, a machine in $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$ must be "on" at times $I_{k,1}^-, I_{k,2}^-, \ldots, I_{k,n_k}^-$. In order for a machine to be launched at time $I_{k,1}^-$, a cost C must be incurred. For each interval $(I_{k,i}^-, I_{k,i+1}^-]$ (where $i \in \{1, \ldots, n_k - 1\}$), if different machines in $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$ are "on" at times $I_{k,i}^-$ and $I_{k,i+1}^-$ respectively, at least a launch cost C is incurred in $(I_{k,i}^-, I_{k,i+1}^-]$. Otherwise, suppose the same machine in $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$ is "on" at times $I_{k,i}^-$ and $I_{k,i+1}^-$. If $I_{k,i+1}^- - I_{k,i}^- \leq C$, the minimum possible cost incurred by the machine during $(I_{k,i}^-, I_{k,i+1}^-]$ is at least $I_{k,i+1}^- - I_{k,i}^-$, i.e., to keep the machine "on" throughout the interval, since terminating and then launching the machine again would incur more cost. If $I_{k,i+1}^- - I_{k,i}^- > C$, the minimum possible cost incurred by the machine during $(I_{k,i}^-, I_{k,i+1}^-]$ is at least a launch cost C, since it incurs more cost to keep the machine "on" throughout the interval. Therefore, the total cost incurred by the machines in $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$ is at least $C + \sum_{i=1}^{n_k - 1} \min\{I_{k,i+1}^- - I_{k,i}^-, C\} \geq C + \sum_{i=1}^{n_k - 1} \min\{I_{k,i+1}^- - I_{k,i}^+, C\}$. As a result, the total auxiliary cost of the machines $\mathcal{M}_k^{\text{MFFs}}$ is at most twice the cost of the machines $\mathcal{M}_{\lceil k/2 \rceil}^{\text{RISL}}$. Thus, the auxiliary cost of all the machines for $\mathcal{J}^s$ by our Modified First-Fit algorithm is bounded by $4 \cdot \sum_{k=1}^{\max_t \lceil s(\mathcal{J}^s, t) \rceil} E(\mathcal{M}_k^{\text{RISL}}) = 4 \cdot \text{OPT}_{\text{RISL}}(\mathcal{J}^s)$.

Theorem 3. The Modified First-Fit algorithm is $(\mu + 9)$-competitive for non-clairvoyant ISL, where μ is the max/min job processing length ratio.

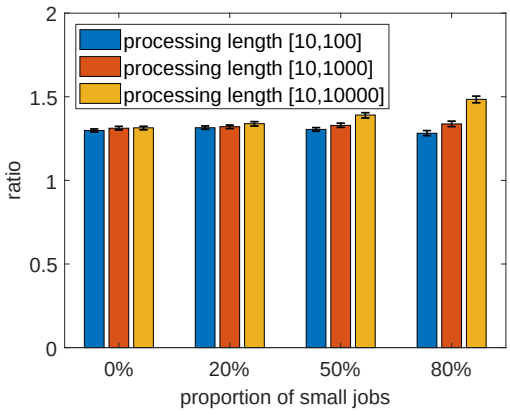


Fig. 6. Simulation results for the Modified First-Fit algorithm over 100 job sequences: the bars represent the average ratio between the total cost of Modified First-Fit and the lower bound of the optimal cost; the lines over the bars represent the 90th and 10th percentile ratios.

We also conduct simulations to study the practical performance of the Modified First-Fit algorithm. We use the same settings as those presented in Section 6. Figure 6 shows the average ratio between the cost of Modified First-Fit and the lower bound on the optimal cost as well as the 90th and 10th percentile ratios for different proportions of small jobs. It can be seen that the ratios are rather close to 1 and are well below the derived competitive ratio of $(\mu+9)$. This indicates that Modified First-Fit produces near-optimal schedules in the settings tested. We further vary the range of job processing length from $[10, 100]$ to $[10, 10000]$ time units. As shown by the results in Figure 6, the performance ratio of Modified First-Fit to the lower bound generally increases with the max/min job processing length ratio μ. This is consistent with the theoretical finding that the competitive ratio of Modified First-Fit increases with μ. The reason is that in the non-clairvoyant online setting, Modified First-Fit is likely to place jobs with different processing lengths on the same machine. As a result, the machines cannot be terminated when short jobs end and have to be kept "on" until long jobs end, increasing the cost of the schedule. In the simulation results, the increase in the performance ratio is more noticeable when a larger proportion of the jobs are small jobs because more jobs can run concurrently on a machine.

Next, we prove that our Modified First-Fit algorithm is $O(1)$-competitive for a special case where each job to be scheduled has a processing length no more than C. Recall

that the total cost for processing the large jobs is bounded by $2 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^l)$ and the auxiliary cost for processing the small jobs is bounded by $4 \cdot \mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s)$. We thus only need to bound the total busy time (busy cost) of the machines used for $\mathcal{J}^s$ in this special case. In the following, we prove that the busy cost for processing $\mathcal{J}^s$ is bounded by $2 \cdot \mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s)$.

Consider the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ of a layer k by our Modified First-Fit algorithm. Let $\mathcal{J}_k^{\mathrm{MFFs}}$ be the set of small jobs processed by the machines $\mathcal{M}_k^{\mathrm{MFFs}}$. Then, all the busy intervals of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ are given by the union of the active intervals of the jobs $\mathcal{J}_k^{\mathrm{MFFs}}$, i.e., $\bigcup_{i=1}^{n_k} I_{k,i} = \bigcup_{J \in \mathcal{J}_k^{\mathrm{MFFs}}} I(J)$. If each job has a processing length at most C, we have $\bigcup_{J \in \mathcal{J}_k^{\mathrm{MFFs}}} I(J) \subseteq \bigcup_{J \in \mathcal{J}_k^{\mathrm{MFFs}}} [I(J)^-, I(J)^- + C)$. As a result,

$$\bigcup_{i=1}^{n_k} I_{k,i} \subseteq \bigcup_{J \in \mathcal{J}_k^{\mathrm{MFFs}}} [I(J)^-, I(J)^- + C). \tag{5}$$

Now, consider the machines $\mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}$ of level $\lceil \frac{k}{2} \rceil$ in the optimal RISL schedule for $\mathcal{J}^s$. It has been shown by the former analysis that a machine in $\mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}$ must be in the "on" state at the arrival of every small job placed on a machine of $\mathcal{M}_k^{\mathrm{MFFs}}$ by our Modified First-Fit algorithm. Thus, for each job $J \in \mathcal{J}_k^{\mathrm{MFFs}}$, $I(J)^- \in \bigcup_{M \in \mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}} Q(M)$, where $Q(M)$ denotes the "on" interval of a machine M. As a result, if we extend the "on" interval of every machine in $\mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}$ by length C, all the intervals $[I(J)^-, I(J)^- + C)$ would be covered, i.e., $\forall\, J \in \mathcal{J}_k^{\mathrm{MFFs}}$,

$$[I(J)^-, I(J)^- + C) \subseteq \bigcup_{M \in \mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}} [Q(M)^-, Q(M)^+ + C). \tag{6}$$

Combining (5) and (6), we have

$$\bigcup_{i=1}^{n_k} I_{k,i} \subseteq \bigcup_{M \in \mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}} [Q(M)^-, Q(M)^+ + C).$$

Note that all the busy intervals of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ are disjoint. Therefore, the total busy time of the machines $\mathcal{M}_k^{\mathrm{MFFs}}$ is bounded by

$$\begin{aligned}
\sum_{i=1}^{n_k} \mathrm{len}(I_{k,i}) &= \mathrm{len}\Big(\bigcup_{i=1}^{n_k} I_{k,i}\Big) \\
&\le \mathrm{len}\Big(\bigcup_{M \in \mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}} [Q(M)^-, Q(M)^+ + C)\Big) \\
&\le \sum_{M \in \mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}} \big(\mathrm{len}(Q(M)) + C\big) \\
&= C \cdot |\mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}| + \sum_{M \in \mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}} \mathrm{len}(Q(M)).
\end{aligned}$$

Note that the right end of the above inequality is exactly the cost of the machines $\mathcal{M}_{\lceil k/2 \rceil}^{\mathrm{RISL}}$. By adding up the above inequality for all the layers of machines, the total busy time of the machines used for $\mathcal{J}^s$ by our Modified First-Fit algorithm is bounded by $2 \cdot \mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J}^s)$. Hence, we have the following result.

Theorem 4. The Modified First-Fit algorithm is 8-competitive for non-clairvoyant ISL when each job has a processing length at most C.

This important result of Theorem 4 motivates the design of our algorithms in the clairvoyant online setting next.

8 Clairvoyant Setting

Finally, we consider ISL in the clairvoyant online setting, where the departure time of a job is known at its arrival and can be used for scheduling purposes. Recall that Azar and Vainstein [6] proposed a $O(\sqrt{\log \mu})$-competitive Hybrid algorithm for MinUsageTime DBP in the clairvoyant setting. In this section, we propose a $O(\sqrt{\log \mu})$-competitive algorithm for clairvoyant ISL based on our Modified First-Fit algorithm and the Hybrid algorithm introduced in [6].

The algorithm works by scheduling jobs shorter and longer than C separately. If an incoming job J has a processing length less than C, i.e., $\mathrm{len}(I(J)) \le C$, we apply our Modified First-Fit algorithm to schedule the job. Otherwise, the Hybrid algorithm is applied to schedule the job. The Hybrid algorithm defines two types of bins (general bins and classify-by-duration bins) for placing items in MinUsageTime DBP. For both bin types, an open bin is closed once it becomes empty. In our context of job scheduling, this implies that whenever a machine used for processing the jobs by the Hybrid algorithm becomes idle, we terminate the machine immediately. Thus, the machines used for processing jobs longer than C are busy throughout their "on" intervals.

Now, we show that the above algorithm is $O(\sqrt{\log \mu})$-competitive, which is asymptotically tight. Let $\mathcal{J}^M$ and $\mathcal{J}^H$ denote the sets of jobs scheduled by the Modified First-Fit algorithm and the Hybrid algorithm respectively. According to Theorem 4, the total cost of the machines used for processing $\mathcal{J}^M$ is bounded by $8 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^M)$. On the other hand, each job in $\mathcal{J}^H$ has a processing length longer than C. By applying the Hybrid algorithm, any machine M used for processing the jobs $\mathcal{J}^H$ must keep busy for at least C time units and thus have an "on" interval at least C long. Therefore, the cost of machine M is bounded by $C + \mathrm{len}(Q(M)) \le 2 \cdot \mathrm{len}(Q(M))$, where $Q(M)$ denotes the "on" interval of M. This suggests that the total cost of the machines used for processing $\mathcal{J}^H$ is bounded by twice their total busy time. Based on the competitiveness of the Hybrid algorithm [6], the total busy time of the machines used for $\mathcal{J}^H$ is bounded by $O(\sqrt{\log \mu}) \cdot \mathrm{OPT}_{\mathrm{BUSY}}(\mathcal{J}^H)$, where $\mathrm{OPT}_{\mathrm{BUSY}}(\mathcal{J}^H)$ is the minimum possible busy time of machines for processing the jobs $\mathcal{J}^H$. Obviously, $\mathrm{OPT}_{\mathrm{BUSY}}(\mathcal{J}^H) \le \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^H)$. It follows that the total busy time of the machines used for $\mathcal{J}^H$ is bounded by $O(\sqrt{\log \mu}) \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^H)$. In summary, we can bound the total cost for processing all the jobs $\mathcal{J}$ by $8 \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^M) + O(\sqrt{\log \mu}) \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J}^H) \le O(\sqrt{\log \mu}) \cdot \mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$.

Theorem 5. There exists a $O(\sqrt{\log \mu})$-competitive algorithm for clairvoyant ISL, where μ is the max/min job processing length ratio.

9 Concluding Remarks

We have studied the algorithmic aspects of ISL in the offline, non-clairvoyant online and clairvoyant online settings. The results show that the approximation and competitiveness bounds of interval job scheduling remain asymptotically the

same with the consideration of machine launch cost. We remark that several results in this paper can be improved for a special case where all the jobs have the same size so that a machine can run at most a constant number of g ($g \geq 2$) jobs concurrently at any time (referred to as ISL-Uniform). In this case, there is no need to schedule the large and small jobs separately. By leveraging 2-approximation algorithms for the ISBP problem [3], [19], we can develop a 2-approximation algorithm for ISL-Uniform in the offline setting. Moreover, our Modified First-Fit algorithm would become $\min\{\mu + 4, g + 2\}$-competitive for ISL-Uniform in the non-clairvoyant setting (with the busy cost bounded by $\min\{\mu+2, g\}\cdot\mathrm{OPT}_{\mathrm{ISL}}(\mathcal{J})$ and the auxiliary cost bounded by $2\cdot\mathrm{OPT}_{\mathrm{RISL}}(\mathcal{J})$ because the "on" machines of all the layers lower than k must be fully occupied when a job is placed on a layer-k machine).

One direction for future work is to conduct more experimental evaluation of the proposed algorithms for other job characteristics and workloads [10]. Another future direction is to study interval job scheduling on heterogeneous machines. Our current work is limited to homogeneous machines. In cloud computing, a cloud service provider often provides multiple types of pre-defined virtual machines for customers to rent. A virtual machine with larger capacity generally has a higher cost rate for renting. Deriving a good combination of machines of various types to optimize the cost of processing jobs will be an interesting topic to study.

ACKNOWLEDGMENTS

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 1 under Grant 2019-T1-002-042, by the National Natural Science Foundation of China under Grant 61902063, and by the Provincial Natural Science Foundation of Jiangsu, China under Grant BK20190342.

REFERENCES

[1] Amazon EC2 pricing. http://aws.amazon.com/ec2/pricing/.

[2] S. Albers. On energy conservation in data centers. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 35–44. ACM, 2017.

[3] M. Alicherry and R. Bhatia. Line system design and a generalized coloring problem. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, pages 19–30, 2003.

[4] J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 530–539. IEEE, 2004.

[5] Y. Azar, N. Ben-Aroya, N. R. Devanur, and N. Jain. Cloud scheduling with setup cost. In *Proceedings of the 25th annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 298–304. ACM, 2013.

[6] Y. Azar and D. Vainstein. Tight bounds for clairvoyant dynamic bin packing. *ACM Transactions on Parallel Computing*, 6(3):article no. 15, 2019.

[7] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

[8] J. Chang, H. N. Gabow, and S. Khuller. A model for minimizing active processor time. *Algorithmica*, 70(3):368–405, 2014.

[9] 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.

[10] Y. Cheng, A. Anwar, and X. Duan. Analyzing alibaba's co-located datacenter workloads. In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, pages 292–297, 2018.

[11] 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. *Theoretical Computer Science*, 411(40-42):3553–3562, 2010.

[12] J. Gergov. Algorithms for compile-time memory optimization. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 907–908. Society for Industrial and Applied Mathematics, 1999.

[13] S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Transactions on Embedded Computing Systems*, 2(3):325–346, 2003.

[14] S. Kamali and A. López-Ortiz. Efficient online strategies for renting servers in the cloud. In *Proceedings of the 41st International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 277–288, 2015.

[15] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[16] 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.

[17] S. Khuller, J. Li, and B. Saha. Energy efficient scheduling via partial shutdown. In *Proceedings of the 21st annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1360–1372. Society for Industrial and Applied Mathematics, 2010.

[18] F. Koehler and S. Khuller. Busy time scheduling on a bounded number of machines. In *Proceedings of the 15th Algorithms and Data Structures Symposium (WADS)*, pages 521–532. Springer, 2017.

[19] V. Kumar and A. Rudra. Approximation algorithms for wavelength assignment. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 152–163. Springer-Verlag, 2005.

[20] J. Li and S. Khuller. Generalized machine activation problems. In *Proceedings of the 22nd annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 80–94. SIAM, 2011.

[21] Y. Li, X. Tang, and W. Cai. On dynamic bin packing for resource allocation in the cloud. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 2–11, 2014.

[22] 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.

[23] M. Mao and M. Humphrey. A performance study on the VM startup time in the cloud. In *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD)*, pages 423–430, 2012.

[24] 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.

[25] 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 (SPAA)*, pages 227–237, 2016.

[26] 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.

[27] 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.

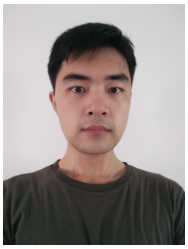
[28] X. Tang, Y. Li, R. Ren, and W. Cai. On first fit bin packing for online cloud server allocation. In *Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 323–332, 2016.

[29] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester. Trends in worldwide ict electricity consumption from 2007 to 2012. *Computer Communications*, 50:64–76, 2014.

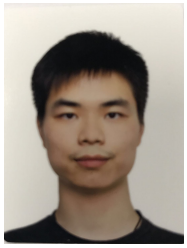
[30] N. Vasić, M. Barisits, V. Salzgeber, and D. Kostic. Making cluster applications energy-aware. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds (ACDC)*, pages 37–42. ACM, 2009.

[31] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

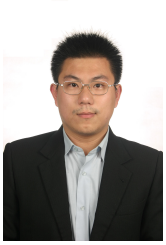
[32] P. Winkler and L. Zhang. Wavelength assignment and generalized interval graph coloring. In *Proceedings of the 14th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 830–831. Society for Industrial and Applied Mathematics, 2003.



Runtian Ren received the BSc degree in mathematics and applied mathematics from University of Science and Technology of China in 2014, and the PhD degree in computer science from Nanyang Technological University in 2019. He is currently a research fellow in School of Computer Science and Engineering at Nanyang Technological University, Singapore. His research interests include scheduling, online algorithm and approximation algorithm.



Yuqing Zhu received the BSc degree in computer science from Nanjing University of Science and Technology in 2017. He is currently a PhD student in Interdisciplinary Graduate School at Nanyang Technological University, Singapore.



Chuanyou Li received the PhD degree in computer science from Southeast University in 2015. He is currently an assistant professor in School of Computer Science and Engineering, Southeast University, Nanjing, China. He worked as a post-doctor at INRIA, France from 2015 to 2016 and worked as a research fellow at Nanyang Technological University, Singapore from 2016 to 2018. His research interests include principles of parallel and distributed computing, distributed systems and fault tolerance.

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. He has served as an associate editor of IEEE Transactions on Parallel and Distributed Systems, and a program co-chair of IEEE ICPADS 2012, CloudCom 2014 and ICDCS 2020. He is a senior member of the IEEE.