Busy-Time Scheduling on Heterogeneous Machines: Algorithms and Analysis

Mozhengfu Liu and Xueyan Tang

Abstract—We study a generalized busy-time scheduling model on heterogeneous machines. The input to the model includes a set of jobs and a set of machine types. Each job has a size and a time interval during which it should be processed. Each job is to be placed on a machine for execution. Different types of machines have distinct capacities and cost rates. The total size of the jobs running on a machine must always be kept within the machine's capacity, giving rise to placement restrictions for jobs of various sizes among the machine types. Each machine used is charged according to the time duration in which it is busy, i.e., it is processing jobs. The objective is to schedule the jobs into machines to minimize the total cost of all the machines used. We develop an $O(1)$-approximation algorithm in the offline setting and an $O(\mu)$-competitive algorithm in the online setting (where μ is the max/min job length ratio), both of which are asymptotically optimal. This paper significantly improves the analysis of the algorithms over our preliminary work.

Index Terms—busy time; scheduling; analysis of algorithms; approximation ratio; competitive ratio.

1 Introduction

In this paper, we study generalized busy-time scheduling on heterogeneous machines. In this model, each job is specified by a size and a time interval of execution. The job size refers to the instantaneous resource demand for job execution, while the duration of the execution interval is called the job length. The jobs are to be scheduled into machines nonpreemptively. At any time, the total size of the jobs running on a machine cannot exceed the machine's capacity. Each machine used is charged proportional to its busy time which is the total length of the time periods in which it is processing jobs. Multiple types of machines with different capacities and cost rates are available. The goal is to schedule the jobs into machines to minimize the accumulated cost of all the machines used. We focus on the algorithmic aspects of the above model and aim to develop effective solutions in both the offline and online settings.

Our busy-time scheduling model has useful applications. Major cloud providers such as Amazon EC2 [3], Google Cloud [9] and Microsoft Azure [16] provide different types of predefined server instances (virtual machines) for customers to rent at different rates. Jobs with various resource demands have placement constraints among the server types accordingly. The servers rented from the clouds are charged according to their working hours. It is critical for cloud users to decide the types and numbers of servers to rent in order to minimize the total rental cost for processing their jobs. Our model elegantly captures the "pay-as-you-go" billing feature of the clouds and the goal of optimizing the monetary expenses for cloud users.

There have been quite a few studies on busy-time scheduling, but almost all of them assumed homogeneous machines only. Earlier work has investigated scheduling interval jobs of uniform sizes, so that each machine can run at most a fixed number of jobs simultaneously [25], [2], [11], [8], [21], [15], [7]. This problem was termed interval scheduling with bounded parallelism and is NP-hard. More recent work has addressed scheduling interval jobs of non-uniform sizes, where the number of jobs that can share a machine is not fixed [12], [13], [23], [19], [17], [4]. This problem was termed MinUsageTime dynamic bin packing. For both problems, the objective is to minimize the total machine busy time for processing a given set of jobs. In the offline setting where all the jobs are known, there exist $O(1)$-approximation algorithms for both problems [2], [11], [8], [10], [17], [7]. For jobs of uniform sizes, Flammini *et al.* [8] presented a 4-approximation First Fit algorithm that schedules jobs in descending order of length. Chang *et al.* [7] showed that the work of Alicherry and Bhatia [2] and that of Kumar and Rudra [11] imply 2-approximation algorithms. They also proposed a 3-approximation GreedyTracking algorithm for a more general case of scheduling flexible jobs that have laxity in starting. For jobs of non-uniform sizes, Khandekar *et al.* [10] gave a 5-approximation algorithm by extending the work of [8]. Ren and Tang [17] presented a 4-approximation dual coloring algorithm by extending the work of [11]. Very recently, Buchbinder *et al.* [6] presented algorithms with improved asymptotic approximation ratios. In the online setting where jobs are released when they are to start execution and the job lengths are not known until they complete execution, the competitiveness of scheduling is bounded from below by the variation of job lengths for both problems [12], [13]. That is, the competitive ratio of any online algorithm is $\Omega(\mu)$, where μ is the max/min job length ratio among all the jobs to schedule. It has been shown that the First Fit algorithm achieves a competitive ratio of $\mu+3$ for scheduling jobs of non-uniform sizes, closely matching the lower bound [19]. In the case that the length of a job is revealed when it is released, the competitiveness of scheduling has a tight bound of $\Theta(\sqrt{\log \mu})$ [4]. In addition, recent work has also considered discrete charging unit [22], machine launch cost [20], and load predictions [6] in busy-time scheduling. However, none of the above work has studied multiple types of machines.

With heterogeneous machines, jobs have different restrictions on which types of machines can process them. In addition, various machine types can differ in the normalized cost rate per capacity

• *Mozhengfu Liu and Xueyan Tang are with School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: {mozhengfu.liu, asxytang}@ntu.edu.sg.*

unit. As a result, the cost of scheduling each job depends on not only other overlapping jobs scheduled on the same machine but also the machine type. To the best of our knowledge, the only work that considered heterogeneous machines was [18], which investigated just two simple cases in which the normalized cost rate per capacity unit increases or decreases monotonically with the machine capacity. In both cases, it was shown that there exist $O(1)$-approximation and $O(\mu)$-competitive algorithms. The authors of [18] conjectured that in the general case, the asymptotic approximability and competitiveness of scheduling would be dependent on the cost and capacity profiles of the machine types. In this paper, we close this open problem by developing $O(1)$-approximation and $O(\mu)$-competitive algorithms in the offline and online settings respectively for *any* set of machine types and *any* set of jobs, when there are plenty of machines available for each type.

We note that there has also been considerable research on power-down mechanisms to transition a machine into low-power modes for conserving energy and on dynamic speed scaling to apply low speeds when possible for saving energy [1]. However, the focuses of these problems are different from busy-time scheduling. The power management problem aims to strike a balance between the energy consumption in the high-power states and the energy overheads of power-up operations to transition from a low-power state to a high-power state when needed. Dynamic speed scaling assumes that jobs can run at variable speeds and the power consumption is a function of the machine speed. The goal is to dynamically set the machine speed to minimize energy consumption while providing the desired quality of service.

This paper significantly extends a preliminary conference version [14]. We streamline the algorithm analysis and considerably improve the approximation/competitiveness results. The rest of this paper is organized as follows. Section 2 formally defines the busy-time scheduling problem on heterogeneous machines. Section 3 introduces some preliminaries for the algorithm design and analysis. Sections 4 and 5 present the design and analysis of the offline and online algorithms respectively.

2 Problem Definition

Formally, the input to the Busy-time Scheduling on Heterogeneous Machines (BSHM) model includes a set of jobs $\mathcal{J}$ and a set of machine types $\mathcal{M}$.

Each job $J \in \mathcal{J}$ has its size $s(J)$ which represents the resource demand for processing J, and its time interval of execution $I(J) := [I(J)^-, I(J)^+)$. We refer to $I(J)$ as J's *active interval* and say that J is *active* during $I(J)$. We also refer to the two endpoints $I(J)^-$ and $I(J)^+$ of $I(J)$ as J's start and end times respectively. We denote the length of $I(J)$ by $\mathrm{len}(J) := I(J)^+ - I(J)^-$ and refer to it as the *job length*.

Each job needs to be scheduled to run on a single machine. Let $\mathcal{M} = \{1, 2, \ldots, |\mathcal{M}|\}$ be the set of the indices of all machine types available, where every machine type indexed by $z \in \{1, 2, \ldots, |\mathcal{M}|\}$ has a cost rate r_z (per unit time) and a resource capacity g_z. At any time, the total size of the jobs running on a machine cannot exceed the machine capacity. Each machine used is charged at its cost rate for the time duration in which it is processing at least one job (known as its busy time). There are sufficient machines of each type available. The objective of BSHM is to minimize the total cost of machine usage for processing all the jobs $\mathcal{J}$.

We study both the offline and online settings of BSHM. The difference between the two settings lies in how much information about $\mathcal{J}$ can be used for scheduling each job. In the offline setting, all the information about $\mathcal{J}$ can be used, while in the online setting, only the information available before each job J starts can be used for scheduling J, i.e., this includes the start times and sizes of the jobs started before $I(J)^-$ and the end times of the jobs ended before $I(J)^-$.

The performance of an offline algorithm or an online algorithm is often characterized by its *approximation ratio* or *competitive ratio*, i.e., the worst-case ratio between a solution constructed by the algorithm and an optimal solution over all instances of the problem [5], [24]. To facilitate algorithm analysis, we assume that the cost rate of each machine type z is a power of b, where $b > 1$ is a constant, i.e., $r_z \in \{b^n : n \in \mathbf{Z}\}$, where $\mathbf{Z}$ denotes the set of all integers. This assumption will cause us to lose at most a factor of $b = O(1)$ in deriving the approximation or competitive ratio of any algorithm. Specifically, for each $z \in \mathcal{M}$, suppose c_z is the real cost rate of machine type z, which can be any positive value, and $r_z \in \{b^n : n \in \mathbf{Z}\}$ is the power of b such that $\frac{1}{b} \cdot r_z < c_z \leq r_z$, which is the assumed cost rate for machine type z to be used throughout the rest of this paper. Given a set of jobs $\mathcal{J}$, the optimal scheduling for machine cost rates r_z's must have a cost within b times the optimal scheduling for machine cost rates c_z's. Then, the approximation or competitive ratio of an algorithm increases by at most a factor of b from cost rates r_z's to c_z's.

For two different machine types i and j, if their capacities satisfy $g_i \leq g_j$ and their cost rates satisfy $r_i \geq r_j$, then type-i machines will not be needed for processing jobs because any type-i machine used can be replaced by a type-j machine that has at least the same capacity but lower or equal cost. Thus, without loss of generality, we assume that the machine types have distinct capacities and it holds that $g_1 < g_2 < \cdots < g_{|\mathcal{M}|}$ and $r_1 < r_2 < \cdots < r_{|\mathcal{M}|}$.

To facilitate presentation, we further define some notations. We denote by $\mathrm{OPT}_{\mathrm{BSHM}}(\mathcal{X})$ the optimal cost of scheduling any given set of jobs $\mathcal{X}$ for the BSHM problem. For any job J, $m(J)$ denotes the machine type in $\{1, 2, \ldots, |\mathcal{M}|\}$ such that $s(J) \in (g_{m(J)-1}, g_{m(J)}]$, i.e., $m(J)$ is the lowest-indexed machine type that can accommodate job J. We refer to $m(J)$ as the *exact machine type* of J. For any set of jobs $\mathcal{X}$, $S(\mathcal{X}) := \sum_{J \in \mathcal{X}} s(J)$ denotes the total size of these jobs, and $\mathrm{span}(\mathcal{X}) := \cup_{J \in \mathcal{X}} I(J)$ denotes the time interval(s) in which at least one job in $\mathcal{X}$ is active. For any set of jobs $\mathcal{X}$ and any time instant t, $\mathcal{X}(t) := \{J \in \mathcal{X} : t \in I(J)\}$ denotes the active jobs in $\mathcal{X}$ at time t, and $S(\mathcal{X}, t) := S(\mathcal{X}(t))$ denotes the total size of the active jobs at time t.

For ease of reference, Table 1 summarizes the key notations in this paper (including those to be introduced later).

3 Preliminaries

Our algorithm design and analysis are built upon two basic concepts: cost-per-capacity graph and one-shot job scheduling. The cost-per-capacity graph characterizes the cost-effectiveness of different machine types, which will guide which machine type to use in scheduling jobs. One-shot job scheduling is a relaxed problem that focuses on assigning jobs to machines at a single time instant. It provides a lower bound on the total cost needed to run the active jobs and facilitates the analysis of approximation and competitiveness.

TABLE 1
Summary of key notations

Notation	Definition
$s(J)$	size of job J
$I(J)$	active interval of job J
$\text{len}(J)$	length of job J
$m(J)$	exact machine type of job J
$\text{span}(\mathcal{J})$	time span of job set $\mathcal{J}$
$\mathcal{J}(t)$	active jobs in job set $\mathcal{J}$ at time t
$S(\mathcal{J})$	total size of job set $\mathcal{J}$
$S(\mathcal{J}, t)$	total size of active jobs in job set $\mathcal{J}$ at time t
μ	max/min job length ratio
b	base value for cost rates of machine types
$\mathcal{M}$	set of machine types
r_z	cost rate of a type-z machine
g_z	capacity of a type-z machine
$p(z)$	parent of machine type z
$P(z)$	machine type z and all its ancestors
$f(z)$	children of machine type z
$A(z)$	tree rooted at machine type z
$y(z)$	younger siblings of machine type z
$e(z)$	elder siblings of machine type z
$T(z)$	machine type z and all the younger siblings of $P(z)$
H_z	jobs whose exact machine types are in the tree rooted at type z

3.1 Cost-per-capacity graph

A main challenge for the general BSHM problem comes from the arbitrary order of the normalized cost rates per capacity unit among different machine types. We construct a directed graph called the *cost-per-capacity graph* to describe the relations among the machine types in terms of their normalized cost rates.

Definition 3.1. In the cost-per-capacity graph, each node i represents a machine type $i \in \mathcal{M}$. Each node i has a directed edge pointing to node $p(i) := \min\{j : j > i \wedge r_j/g_j < r_i/g_i\}$ if such $p(i)$ exists (i.e., $p(i)$ is the lowest-indexed machine type above i that has a lower normalized cost rate than i). By convention, we define $p(i) = \perp$ if the set $\{j : j > i \wedge r_j/g_j < r_i/g_i\}$ is empty.

Since edges always point from lower-indexed nodes to higher-indexed nodes, there is no cycle in the cost-per-capacity graph. Moreover, since each node has at most one outgoing edge, for any two nodes $i < j$, there is at most one path from i to j. Thus, the graph must be a forest.

Proposition 3.1.1. The cost-per-capacity graph is a forest.

For simplicity, we shall use the terms "node" and "machine type" (or "type") interchangeably. We say that $p(i)$ is the parent of i, and i is a child of $p(i)$. Let $f(i)$ denote the set of children of node i: $f(i) := \{z : p(z) = i\}$. Let $P(i)$ denote the set of nodes including node i and all its ancestors $p(i)$, $p(p(i))$, $p(p(p(i)))$, ... Let $A(i)$ denote the set of nodes in the tree rooted at node i: $A(i) := \{z : i \in P(z)\}$.

Let $y(i)$ denote the set of younger siblings of node i: $y(i) := \{z : z < i \wedge p(z) = p(i)\}$, and let $e(i)$ denote the set of elder siblings of node i: $e(i) := \{z : z > i \wedge p(z) = p(i)\}$. Furthermore, let $T(i)$ denote the set of nodes including i and all the younger siblings of $P(i)$: $T(i) := \{i\} \cup \big(\cup_{z \in P(i)} y(z)\big)$.

Figure 1 shows an example cost-per-capacity graph. By the above definitions, $p(10) = 11$, $P(10) = \{10, 11, 13\}$, $f(11) = \{9, 10\}$, $A(11) = \{8, 9, 10, 11\}$, $y(11) = \{7\}$, and $e(11) = \{12\}$. In Figure 1, the nodes in grey constitute $T(10) = \{3, 5, 7, 9, 10\}$.

The cost-per-capacity graph has several properties useful to our analysis. The first property below follows directly from the definition of the graph.

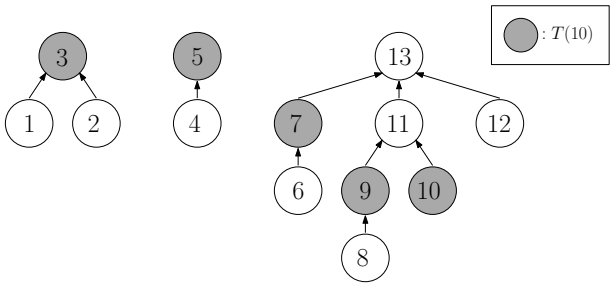

Fig. 1. An example of the cost-per-capacity graph

Proposition 3.1.2. For any $k \in \mathcal{M}$, $r_k/g_k \leq r_i/g_i$ for each $i \in A(k)$, and $r_k/g_k \leq r_j/g_j$ for each $j \in e(k)$.

By the above proposition, the root of the leftmost tree (e.g., machine type 3 in the example of Figure 1) has the lowest normalized cost rate among all machine types.

Proposition 3.1.3. For any $k \in \mathcal{M}$, the node set $A(k)$ of the tree rooted at k has consecutive indexes, and k is the highest-indexed node in $A(k)$.

Proof. Since all the edges point from lower-indexed nodes to higher-indexed nodes, each node $k \in \mathcal{M}$ must have the highest index among all the nodes in the tree rooted at k.

Given any k, let j_1 denote the lowest-indexed node in the tree rooted at k. Let $j_1 < j_2 < \cdots < j_n = k$ denote all the nodes along the path from j_1 to k. By definition, we have $r_k/g_k = r_{j_n}/g_{j_n} < r_{j_{n-1}}/g_{j_{n-1}} < \cdots < r_{j_1}/g_{j_1}$. In addition, for each $q = 1, 2, \ldots, n-1$, since j_{q+1} is the lowest-indexed node having a lower normalized cost rate than j_q, for any node i between j_q and j_{q+1}, we have $r_k/g_k < r_{j_q}/g_{j_q} \leq r_i/g_i$. This implies that the parent of i must have an index no higher than k. As a result, all the nodes between j_1 and k have parents indexed no higher than k. Thus, they must all be in the tree rooted at k. □

Proposition 3.1.4. For any $k \in \mathcal{M}$ and any $i_1, i_2 \in T(k)$, if $i_1 < i_2$, then $r_{i_1}/g_{i_1} \leq r_{i_2}/g_{i_2}$. That is, for the nodes in $T(k)$, their normalized cost rates are non-decreasing with indexes.

Proof. By definition, $T(k) = \{k\} \cup \big(\cup_{z \in P(k)} y(z)\big)$. Thus, for any $i \in T(k) \setminus \{k\}$, we have $i \in y(z)$ for some $z \in P(k)$. Proposition 3.1.3 implies that i is smaller than all the nodes in $A(z)$. Since $k \in A(z)$, it follows that $i < k$. Thus, k is the highest-indexed node in $T(k)$.

Case 1: $i_1 < i_2 < k$.

Since $i_1, i_2 \in T(k)$, we have $i_1 \in y(z_1), i_2 \in y(z_2)$ for some $z_1, z_2 \in P(k)$. We show that $z_1 \geq z_2$ must hold. Otherwise, if $z_1 < z_2$, we have $i_1 \in y(z_1) \subset A(z_2)$ and $i_2 \in y(z_2)$. By Proposition 3.1.3, it holds that $i_2 < i_1$ which contradicts to the condition $i_1 < i_2$. Therefore, $z_1 \geq z_2$. It follows that $i_1 < i_2 < p(i_2) = p(z_2) \leq p(z_1) = p(i_1)$. By definition, $p(i_1)$ is the lowest-indexed node having a lower normalized cost rate than i_1. Thus, we have $r_{i_1}/g_{i_1} \leq r_{i_2}/g_{i_2}$.

Case 2: $i_1 < i_2 = k$.

In this case, we have $i_1 \in y(z)$ for some $z \in P(k)$. Hence, $i_1 < k \leq z < p(z) = p(i_1)$. Similar to Case 1, by the definition of parent, $r_{i_1}/g_{i_1} \leq r_k/g_k$ must hold. □

Proposition 3.1.5. For any $k \in \mathcal{M}$, the trees rooted at the nodes $T(k)$, i.e., $\{A(i) : i \in T(k)\}$, are a partitioning of $\{1, 2, \ldots, k\}$. Furthermore, the trees rooted at the nodes $\cup_{z \in P(k)} e(z)$, i.e., $\{A(j) : j \in e(z) \wedge z \in P(k)\}$, are a partitioning of $\mathcal{M} \setminus (\{1, 2, \ldots, k-1\} \cup P(k))$.

Proof. For each $i \in T(k)$, i is the highest-indexed node in $A(i)$. In addition, k is the highest-indexed node in $T(k)$. Therefore, $\cup_{i \in T(k)} A(i) \subset \{1, 2, \ldots, k\}$.

Since the cost-per-capacity graph is a forest, all the nodes $\mathcal{M}$ can be partitioned into the tree rooted at k ($A(k)$), the ancestors of k ($P(k) \setminus \{k\}$), and the trees rooted at the younger and elder siblings of k and its ancestors ($A(j)$ for each $j \in \cup_{z \in P(k)} (y(z) \cup e(z))$). For any $q \in \mathcal{M} \setminus \cup_{i \in T(k)} A(i)$, it can be shown that $q > k$. Note that by definition $T(k) = \{k\} \cup \left(\cup_{z \in P(k)} y(z)\right)$. Thus, either $q \in P(k) \setminus \{k\}$ or $q \in A(j)$ for some $j \in e(z)$ where $z \in P(k)$. In the former case, clearly $q > k$. In the latter case, by Proposition 3.1.3, since $k \in A(z)$, it holds that $k \leq z < q$. Therefore, $\cup_{i \in T(k)} A(i) = \{1, 2, \ldots, k\}$ is proven. It also follows that $\cup_{j \in \cup_{z \in P(k)} e(z)} A(j) = \mathcal{M} \setminus \cup_{i \in T(k)} A(i) \setminus (P(k) \setminus \{k\}) = \mathcal{M} \setminus (\{1, 2, \ldots, k-1\} \cup P(k))$. □

3.2 One-shot job scheduling

To understand the optimal cost of BSHM, we start by considering scheduling jobs on heterogeneous machines at a single time instant and refer to this problem as *one-shot scheduling*. In the one-shot scheduling problem, we relax the constraint that each job must be scheduled into a single machine and allow a job to be divided into multiple pieces along its size dimension and each piece to be scheduled into a distinct machine. However, we retain the restriction that all the machines into which a job is scheduled must have capacities no less than the original size of the job. Furthermore, we relax the constraint that the number of machines used for each machine type must be an integer. We allow the number of machines for each machine type to be any non-negative real number. For the highest-indexed machine type used, we require that its number of machines must be at least 1. The goal of one-shot scheduling is to minimize the total cost rate of all the machines used for accommodating the jobs.

Note that BSHM does not allow jobs to be divided and it enforces each job to be scheduled into the same machine throughout its active interval. BSHM also requires the number of machines used to be non-negative integers. Therefore, the optimal cost $\mathrm{OPT}_{\mathrm{BSHM}}(\mathcal{J})$ of BSHM for a set of jobs $\mathcal{J}$ is bounded from below by the accumulated cost of optimal one-shot scheduling for the active jobs $\mathcal{J}(t)$ at each time instant t, i.e.,

$$\mathrm{OPT}_{\mathrm{BSHM}}(\mathcal{J}) \geq \int_t \mathrm{OPT}_1(\mathcal{J}(t))\,\mathrm{d}t, \tag{1}$$

where $\mathrm{OPT}_1(\mathcal{J}(t))$ denotes the optimal cost rate of one-shot scheduling for the jobs $\mathcal{J}(t)$. Note that in one-shot scheduling, only the sizes of the jobs $\mathcal{J}(t)$ matter while the active intervals of the jobs are irrelevant. We shall use the above lower bound in the analysis of algorithm performance with respect to $\mathrm{OPT}_{\mathrm{BSHM}}(\mathcal{J})$.

We define a *machine configuration* w as a set of numbers $\{w(z) : z \in \mathcal{M}\}$, each representing the number of machines for a machine type. Given a set of jobs $\mathcal{J}^{1d}$, the one-shot scheduling problem essentially seeks a minimum-cost machine configuration described by the following optimization problem:[1]

$$\min \sum_{z \in \mathcal{M}} w(z) \cdot r_z \tag{2}$$

$$\text{s.t. } S(\{J \in \mathcal{J}^{1d} : m(J) \geq i\}) \leq \sum_{z \geq i} w(z) \cdot g_z, \forall i \in \mathcal{M}; \tag{3}$$

$$w(z) \geq 0, \forall z \in \mathcal{M}; \tag{4}$$

$$w(k) \geq 1, \text{ where } k = \max\{z : w(z) > 0\}. \tag{5}$$

Constraint (3) says that the total capacity of the machines of types at least i must be no less than the total size of the jobs whose exact machine types are at least i. Constraint (4) says that the number of machines used for each machine type is non-negative. Constraint (5) says that the number of machines for the highest-indexed machine type used must be at least 1. We remark that the last constraint (5) is essential for one-shot scheduling to be a reasonable approximation of real scheduling that requires the machine numbers to be integers. Without constraint (5), the optimal one-shot scheduling would schedule every job J into the machine type of the lowest normalized cost rate among those with sufficient capacity for J, so that the number of machines used for the highest-indexed type can be a very small fractional number. As a result, the cost rate of real scheduling can be arbitrarily higher than the optimal one-shot scheduling, making the latter useless in the algorithm analysis.

Remark 3.2.1. The one-shot scheduling problem can be solved in polynomial time. An intuitive approach is to iterate through k (i.e., the highest-indexed machine type used) and compare the best machine configuration for each k value. Given a k value, the best machine configuration is to fill up the capacity of one type-k machine with jobs in decreasing order of size. Each remaining job (including the fraction of the last job that cannot completely fit into the type-k machine) can be scheduled into the machine type of the lowest normalized cost rate and sufficient capacity.

The following are some observations for an optimal machine configurations, where $k_{opt} := \max\{z : w^*(z) > 0\}$ denotes the highest-indexed machine type used by an optimal machine configuration w^*.

Proposition 3.2.1. All the machine types used by w^* must be from $T(k_{opt})$, i.e., $w^*(i) = 0$ for each type $i \in \mathcal{M} \setminus T(k_{opt})$.

Proof. By Proposition 3.1.5, it suffices to show that the machine numbers are 0 for the non-root types in the trees rooted at types $T(k_{opt})$. Assume on the contrary that $w^*(i) > 0$ for some type $i \in A(z) \setminus \{z\}$ for some $z \in T(k_{opt})$. We can replace $w^*(i)$ type-i machines with $\frac{g_i}{g_z} \cdot w^*(i)$ type-z machines that have the same capacity. The new machine configuration is also a feasible solution to one-shot scheduling. Since $i \in A(z) \setminus \{z\}$, by definition, type z has a lower normalized cost rate than type i. As a result, the new machine configuration has a lower total cost rate than w^*, which contradicts to the optimality of w^*. □

Proposition 3.2.2. For each type $i \in P(k_{opt}) \setminus \{k_{opt}\}$, the total cost rate of the machine types used in the tree rooted at i must not exceed the cost rate of one type-i machine, i.e., $\sum_{z \in A(i) \setminus \{i\}} w^*(z) \cdot r_z \leq r_i$.

1. Since only the sizes of the jobs matter in one-shot scheduling, we use the notation $\mathcal{J}^{1d}$ for the input to one-shot scheduling, differentiating it from the input $\mathcal{J}$ to BSHM.

Proof. The argument for this observation is similar to the previous observation. If not, we would be able to construct a feasible machine configuration by replacing type-z machines (where $z \in A(i) \setminus \{i\}$) with type-$i$ machines which leads to a strictly lower total cost rate than w^*. □

Let $H_z = \{J \in \mathcal{J}^{1d} : m(J) \in A(z)\}$ denote the set of jobs whose exact machine types are in the tree rooted at type z.

Proposition 3.2.3. For each type $i \in P(k_{opt}) \setminus \{k_{opt}\}$, the total size of the jobs whose exact machine types are in the tree rooted at i must not exceed the capacity of one type-i machine, i.e., $S(H_i) < g_i$.

Proof. Note that for each type $z \in A(i) \setminus \{i\}$, by the definition of the cost-per-capacity graph, $\frac{g_z}{r_z} < \frac{g_i}{r_i}$. It follows that

$$\begin{aligned}\sum_{z \in A(i)\setminus\{i\}} w^*(z) \cdot g_z &= \sum_{z \in A(i)\setminus\{i\}} w^*(z) \cdot r_z \cdot \frac{g_z}{r_z} \\ &< \sum_{z \in A(i)\setminus\{i\}} w^*(z) \cdot r_z \cdot \frac{g_i}{r_i} \\ &\le r_i \cdot \frac{g_i}{r_i} = g_i \quad \text{(by Proposition 3.2.2).}\end{aligned}$$

On the other hand, observe that since $i \in P(k_{opt}) \setminus \{k_{opt}\}$, $(A(i) \setminus \{i\}) \cap \{1, 2, \ldots, k_{opt}\} = A(i) \cap \{1, 2, \ldots, k_{opt}\} = \{z_0, z_0 + 1, \ldots, k_{opt}\}$, where $z_0 = \min A(i) \cap \{1, 2, \ldots, k_{opt}\}$ (refer to Proposition 3.1.3). By constraint (3) of one-shot scheduling, the feasibility of the machine configuration w^* implies that $\sum_{z \in A(i)\setminus\{i\}} w^*(z) \cdot g_z = \sum_{z=z_0, z_0+1, \ldots, k_{opt}} w^*(z) \cdot g_z \ge S(\{J \in \mathcal{J}^{1d} : m(J) \ge z_0\}) = S(\{J \in \mathcal{J}^{1d} : m(J) \in A(i)\}) = S(H_i)$. □

Proposition 3.2.4. There are two lower bounds for the total cost rate of the optimal machine configuration: (i) $\mathrm{OPT}_1(\mathcal{J}^{1d}) \ge r_{k_{opt}}$; and (ii) $\mathrm{OPT}_1(\mathcal{J}^{1d}) \ge \sum_{z \in T(k_{opt})} \frac{S(H_z)}{g_z} \cdot r_z$.

Proof. The first bound is due to constraint (5) of one-shot scheduling. The second bound is due to constraint (3) of one-shot scheduling and that the normalized cost rates of the machine types in $T(k_{opt})$ are non-decreasing with indexes (Proposition 3.1.4). □

Proposition 3.2.5. There is an upper bound for the total cost rate of the optimal machine configuration: $\mathrm{OPT}_1(\mathcal{J}^{1d}) \le \max\left\{1, \frac{S(H_{k_{opt}})}{g_{k_{opt}}}\right\} \cdot r_{k_{opt}} + \sum_{z \in T(k_{opt})\setminus\{k_{opt}\}} \frac{S(H_z)}{g_z} \cdot r_z$.

Proof. The reason is simple: the machine configuration on the right-hand side is a feasible solution to one-shot scheduling. □

Remark 3.2.2. The optimal machine configuration for one-shot scheduling may not be unique. Consider two machine types which have the same normalized cost rate, i.e., $r_1/g_1 = r_2/g_2$. They are siblings in the cost-per-capacity graph. Suppose all the jobs have sizes less than or equal to g_1 and the total size of them is at least g_2. Clearly, one optimal machine configuration is to use machines of type 2 only which can exactly accommodate all the jobs, and another optimal machine configuration is to use machines of type 1 only which can exactly accommodate all the jobs.

Although the optimal machine configuration may not be unique, for the purpose of analysis, we shall always refer to one particular class of optimal machine configurations described below.

Theorem 3.1. There exists an optimal machine configuration w^* such that $k_{opt} := \max\{z : w^*(z) > 0\} \in P(k_0)$, i.e., the highest-indexed machine type k_{opt} used is either k_0 or an ancestor of k_0 in the cost-per-capacity graph, where $k_0 := \max\{m(J) : J \in \mathcal{J}^{1d}\}$ is the highest-indexed exact machine type among all the jobs.

Proof. Take any optimal machine configuration w_1. We construct another machine configuration w_2 based on w_1 as follows:

$$w_2(z) := \begin{cases} w_1(z) & \text{if } 1 \le z < k_0, \\ w_1(z) + \sum_{j \in e(z)} \sum_{i \in A(j)} \frac{r_i}{r_z} \cdot w_1(i) & \text{if } z \in P(k_0), \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In essence, w_2 modifies w_1 by shifting the costs from the machine types $\{k_0, k_0+1, \ldots, |\mathcal{M}|\} \setminus P(k_0)$ to the machine types $P(k_0)$. By Proposition 3.1.5, the trees rooted at the nodes $\cup_{z \in P(k_0)} e(z)$, i.e., $\{A(j) : j \in e(z) \wedge z \in P(k_0)\}$, form a partitioning of $\mathcal{M} \setminus (\{1, 2, \ldots, k_0 - 1\} \cup P(k_0))$. By the above construction, for each $z \in P(k_0)$, we have $w_1(z) \cdot r_z + \sum_{j \in e(z)} \sum_{i \in A(j)} w_1(i) \cdot r_i = w_2(z) \cdot r_z = w_2(z) \cdot r_z + \sum_{j \in e(z)} \sum_{i \in A(j)} w_2(i) \cdot r_i$. Hence, w_1 and w_2 have the same cost rate.

It remains to show that w_2 is a feasible machine configuration for one-shot scheduling. First, we check w_2 against constraint (3). By the definition of k_0, for each $l > k_0$, $S(\{J \in \mathcal{J}^{1d} : m(J) \ge l\}) = 0 \le \sum_{z=l}^{|\mathcal{M}|} w_2(z) \cdot g_z$.

By Proposition 3.1.2, for each $i \in A(j)$ where $j \in e(z)$ and $z \in P(k_0)$, we have $r_z/g_z \le r_j/g_j \le r_i/g_i$. Therefore, for each $z \in P(k_0)$, $w_1(z) \cdot g_z + \sum_{j \in e(z)} \sum_{i \in A(j)} w_1(i) \cdot g_i = w_1(z) \cdot r_z \cdot \frac{g_z}{r_z} + \sum_{j \in e(z)} \sum_{i \in A(j)} w_1(i) \cdot r_i \cdot \frac{g_i}{r_i} \le w_1(z) \cdot r_z \cdot \frac{g_z}{r_z} + \sum_{j \in e(z)} \sum_{i \in A(j)} w_1(i) \cdot r_i \cdot \frac{g_z}{r_z} = w_2(z) \cdot r_z \cdot \frac{g_z}{r_z} = w_2(z) \cdot g_z$. As a result,

$$\begin{aligned} S(\{J \in \mathcal{J}^{1d} : m(J) \ge k_0\}) &\le \sum_{z=k_0}^{|\mathcal{M}|} w_1(z) \cdot g_z \\ &= \sum_{z \in P(k_0)} \left(w_1(z) \cdot g_z + \sum_{j \in e(z)} \sum_{i \in A(j)} w_1(i) \cdot g_i \right) \\ &\le \sum_{z \in P(k_0)} w_2(z) \cdot g_z \le \sum_{z=k_0}^{|\mathcal{M}|} w_2(z) \cdot g_z, \end{aligned} \tag{7}$$

where the first inequality is due to the feasibility of w_1.

For each $l = 1, 2, \ldots, k_0 - 1$, we have $S(\{J \in \mathcal{J}^{1d} : m(J) \ge l\}) \le \sum_{z=l}^{|\mathcal{M}|} w_1(z) \cdot g_z \le \sum_{z=l}^{|\mathcal{M}|} w_2(z) \cdot g_z$, where the first inequality is due to the feasibility of w_1, and the second inequality is because of (7) and $w_2(z) = w_1(z)$ for each $z = 1, 2, \ldots, k_0 - 1$.

Next, we check w_2 against constraint (5). Let $k_1 := \max\{z : w_1(z) > 0\}$ and $k_2 := \max\{z : w_2(z) > 0\}$ denote the highest-indexed machine types used by w_1 and w_2 respectively. Clearly, $k_2 \in P(k_0)$. If $k_1 \in P(k_0)$, then $k_2 = k_1$ and $w_2(k_2) = w_1(k_1) \ge 1$. If $k_1 \notin P(k_0)$, then $k_1 \in A(j)$ for some $j \in e(k_2)$. Note that in this case $k_2 < k_1$. By definition, $w_2(k_2) \ge \frac{r_{k_1}}{r_{k_2}} \cdot w_1(k_1) \ge \frac{r_{k_1}}{r_{k_2}} \ge b > 1$.

Therefore, w_2 is feasible and hence optimal. □

Remark 3.2.3. The optimal machine configuration described by Theorem 3.1 is still not necessarily unique. Consider two machine types where $r_2 = b \cdot r_1$ and $g_2 = (b + 1) \cdot g_1$. The normalized cost rates satisfy $r_2/g_2 < r_1/g_1$, so type 2 is the parent of type 1. Suppose all the jobs have sizes less than or equal to g_1

and the total size of them is $b \cdot g_1$. Then, one optimal machine configuration is to use one machine of type 2, and another optimal machine configuration is to use b machines of type 1. Both machine configurations conform to the description of Theorem 3.1. For the analysis of our offline algorithm in Section 4, any optimal machine configuration described by Theorem 3.1 suffices. For the analysis of our online algorithm in Section 5, among all the optimal machine configurations described by Theorem 3.1, we shall choose the one with the largest k_{opt}.

4 The Offline Setting

4.1 The offline algorithm: $ALG_{offline}$

We now discuss the offline BSHM problem, which is NP-hard since it is a generalization of interval scheduling with bounded parallelism.

Note that jobs of small sizes may be put into machines of large capacities, but jobs of large sizes cannot be put into machines of smaller capacities. Thus, to make full use of available machine capacities, we schedule jobs into machines in decreasing order of machine capacity. To optimize the cost, large machines are used only when (1) they are required by the exact machine types of outstanding jobs; or (2) they are more cost-effective than small machines to meet the total resource demand of outstanding jobs.

Consider a set of jobs $\mathcal{J}$ for BSHM. For each machine type $z \in \mathcal{M}$, let $H_z = \{J \in \mathcal{J} : m(J) \in A(z)\}$ denote the set of jobs whose exact machine types are in the tree $A(z)$ rooted at z. Algorithm 1 shows our offline algorithm $ALG_{offline}$ for BSHM. The algorithm iteratively determines the set of jobs K_z assigned to each machine type z in descending order of type indexes (line 1). For each machine type z, we consider all the unassigned jobs in H_z, denoted by $\mathcal{R}_z$ (line 2). Note that $\mathcal{R}_z$ includes jobs whose exact machine types are z (denoted by $\mathcal{R}_z^h$ in line 3) and jobs whose exact machine types are z's descendants in the cost-per-capacity graph (given by $\mathcal{R}_z \setminus \mathcal{R}_z^h$). The jobs in $\mathcal{R}_z^h$ must be assigned to type z (line 4) since all the types indexed higher than z have been considered before. For each job J in $\mathcal{R}_z \setminus \mathcal{R}_z^h$, we check whether it is *cost-effective* to open a type-z machine throughout J's active interval. If so, J is assigned to type z (lines 8-9). If not, J is left to subsequent iterations and will be assigned to a descendant type of z. To decide whether it is cost-effective to open a type-z machine at a time instant t, we examine $\mathcal{R}_z(t)$, i.e., all the active jobs in $\mathcal{R}_z$ at time t. If there exists at least one job in $\mathcal{R}_z^h$ active at t (i.e., $t \in \text{span}(\mathcal{R}_z^h)$), time instant t is considered cost-effective (line 6). Otherwise, all the jobs active at t are from $\mathcal{R}_z \setminus \mathcal{R}_z^h$. Thus, each job active at t must have its exact machine type in one of the trees rooted at z's children. For each child type x of z, we compute the number of type-x machines needed to host all the active jobs $F_{x,t}$ whose exact machine types are in the tree $A(x)$ rooted at x (lines 5-6). The total cost rate of the machines calculated in this way gives a lower bound on the cost rate to host all the jobs active at t, since each type x has the lowest normalized cost rate in the tree rooted at x (Proposition 3.1.2). If the total cost rate exceeds a fraction $p > 0$ of the cost rate of a type-z machine, time instant t is considered cost-effective (line 6). Note that the set of active jobs does not change between two successive job starts/ends. Thus, the cost-effectiveness only needs to be evaluated once for each interval between two successive job starts/ends. As a result, the cost-effectiveness evaluation can be conducted in polynomial time.

Algorithm 1: $ALG_{offline}$

Input: A set of jobs $\mathcal{J}$ and a parameter p.
Output: A schedule for $\mathcal{J}$.

1 **for** $z = |\mathcal{M}|, |\mathcal{M}|-1, \ldots, 1$ **do**
2 $\quad \mathcal{R}_z \leftarrow H_z \setminus (\cup_{i=|\mathcal{M}|,|\mathcal{M}|-1,\ldots,z+1} K_i)$;
3 $\quad \mathcal{R}_z^h \leftarrow \{J \in \mathcal{R}_z : m(J) = z\}$;
4 $\quad K_z \leftarrow \mathcal{R}_z^h$;
5 $\quad F_{x,t} \leftarrow \{J \in \mathcal{R}_z(t) : m(J) \in A(x)\}$, for each $x \in f(z)$ and t;
6 $\quad \mathcal{T}_z \leftarrow \text{span}(\mathcal{R}_z^h) \cup \{t : \sum_{x \in f(z)} \frac{S(F_{x,t})}{g_x} r_x \geq p \cdot r_z\}$;
7 $\quad$ **for** *each* $J \in \mathcal{R}_z \setminus \mathcal{R}_z^h$ **do**
8 $\quad\quad$ **if** $I(J) \subset \mathcal{T}_z$ **then**
9 $\quad\quad\quad$ add J into K_z;
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 $\quad$ schedule jobs in K_z into type-z machines by using the dual coloring algorithm (see [17]);
13 **end**

It is easy to infer that $ALG_{offline}$ eventually assigns each job J to either its exact machine type $m(J)$ or an ancestor type of $m(J)$. After assigning the set of jobs K_z to each machine type z, we use an existing dual coloring algorithm [17] to schedule the jobs in K_z into type-z machines (line 12). Dual coloring is a 4-approximation algorithm for scheduling a set of jobs into homogeneous machines (i.e., all machines have the same cost rate and capacity) to minimize the total cost of machine usage.

The output of the $ALG_{offline}$ algorithm has the following properties for any time instant $t \in \text{span}(\mathcal{J})$.

Property 4.1. Let $k_{off} := \max\{z : K_z(t) \neq \emptyset\}$ be the highest-indexed machine type used by $ALG_{offline}$ at time t, and $k_0 := \max\{m(J) : J \in \mathcal{J}(t)\}$ be the highest-indexed exact machine type among the active jobs at time t. We have $k_{off} \in P(k_0)$, i.e., k_{off} is either k_0 or an ancestor of k_0.

Proof. Obviously, $k_{off} \geq k_0$. It suffices to show that k_0 is in the tree rooted at k_{off}. Assume on the contrary that k_0 is not in the tree rooted at k_{off}. By Proposition 3.1.3, k_0 is smaller than all the nodes in the tree rooted at k_{off}. Take any job J assigned to type k_{off}. By the algorithm definition, J's exact machine type $m(J)$ is in the tree rooted at k_{off}. As a result, $k_0 < m(J)$, which contradicts to the definition of k_0. □

Property 4.2 follows from the algorithm definition directly. It says that if k_{off} is higher than the exact machine types of all the active jobs, assigning the active jobs to k_{off}'s child types would incur a total cost rate at least a fraction p of a type-k_{off} machine.

Property 4.2. If $k_{off} > k_0$, we have $\sum_{x \in f(k_{off})} S(H_x, t) \cdot \frac{r_x}{g_x} = \sum_{x \in f(k_{off})} S(F_{x,t}) \cdot \frac{r_x}{g_x} \geq p \cdot r_{k_{off}}$, where $f(k_{off})$ is the set of k_{off}'s child types.

Recall that the cost rate of each machine type is a power of b. For the offline setting, we set the base value $b = 3$.[2] Property 4.3 says that for each type z, the total cost rate of the machines needed for hosting all the jobs assigned to z's descendants is bounded by the cost rate of $\frac{16}{5}p$ type-z machines.

2. Our analysis of the $ALG_{offline}$ algorithm can be easily extended to any base value b. For ease of exposition, we set $b = 3$ in the analysis which can achieve a close to best approximation ratio for $ALG_{offline}$.

$\mathbf{T} = \cup_{i=1}^{4}[a_i, b_i)$
$\mathcal{R}'_z(t) = \{J_1, J_2\}$

$S_1 = [I(J_1)^-, b_1) \cup [a_2, b_2)$
$S_2 = [a_3, I(J_2)^+)$

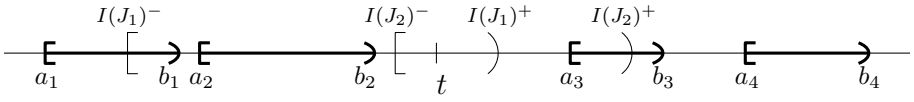

Fig. 2. An illustration of S_1 and S_2 for Case 2 in Property 4.3

Property 4.3. For any machine type $z \in \mathcal{M}$, we have $\sum_{i\in A(z)\setminus\{z\}} \frac{S(K_i,t)}{g_i} \cdot r_i \le \frac{16}{5}p \cdot r_z$, where K_i is the set of jobs assigned to type i, and $S(K_i, t)$ is the total size of the active jobs in K_i at time t.

Proof. We are to show that the above inequality holds for any $z \in \mathcal{M}$ and $t \in \text{span}(\mathcal{J})$. We use induction on the height $l \ge 1$ of the tree rooted at z, where the height of the tree is defined as the number of nodes along the longest path from a leaf node to the root. The base case when $l = 1$ is trivially true because $A(z) \setminus \{z\} = \emptyset$, so it suffices to consider the inductive case. Let $\mathbf{T} := \{t \in \text{span}(\mathcal{J}) : \mathcal{R}_z^h(t) = \emptyset \wedge \sum_{x\in f(z)} \frac{S(F_{x,t})}{g_x} r_x < p \cdot r_z\}$, i.e., $\mathbf{T}$ consists of all time instants that are not cost-effective to open a type-z machine. Observe that the set $\mathbf{T}$ is a union of left closed and right open intervals because the active interval of each job is a left closed and right open interval. For each job $J \in \mathcal{R}_z$, by the definition of $ALG_{offline}$ (lines 8-9), J is assigned to type z if and only if $I(J) \cap \mathbf{T} = \emptyset$, i.e., J's active interval does not contain any time instant of $\mathbf{T}$. For a fixed time instant t, we have either $t \in \mathbf{T}$ or $t \notin \mathbf{T}$.

Case 1: $t \in \mathbf{T}$.

In this case, $K_z(t) = \emptyset$, i.e., no job active at t is assigned to type z. All the jobs $\mathcal{R}_z(t)$ active at t are passed to z's child types $f(z)$. Thus, $\mathcal{R}_z(t) = \cup_{x\in f(z)}\mathcal{R}_x(t)$. By definition, for each child type $x \in f(z)$, $F_{x,t} = \mathcal{R}_x(t) \supset K_x(t)$, where $F_{x,t}$ is computed by line 5 in the iteration of type z, $\mathcal{R}_x(t)$ is the set of active jobs at t passed to type x, and $K_x(t)$ is the set of active jobs at t assigned to type x. Therefore,

$$\begin{aligned}
&\sum_{i\in A(z)\setminus\{z\}} \frac{S(K_i,t)}{g_i} \cdot r_i \\
=&\sum_{x\in f(z)} \left(\frac{S(K_x,t)}{g_x} \cdot r_x + \sum_{i\in A(x)\setminus\{x\}} \frac{S(K_i,t)}{g_i} \cdot r_i \right) \\
\le&\sum_{x\in f(z)} \left(\frac{S(F_{x,t})}{g_x} \cdot r_x + \sum_{i\in A(x)\setminus\{x\}} \frac{S(K_i,t)}{g_i} \cdot r_i \right) \\
<&p \cdot r_z + \sum_{x\in f(z)} \sum_{i\in A(x)\setminus\{x\}} \frac{S(K_i,t)}{g_i} \cdot r_i \\
&\text{(by the definition of } \mathbf{T}) \\
=&p \cdot r_z + \sum_{x\in f(z)\wedge A(x)\setminus\{x\}\neq\emptyset} \sum_{i\in A(x)\setminus\{x\}} \frac{S(K_i,t)}{g_i} \cdot r_i \\
\le&p \cdot r_z + \sum_{x\in f(z)\wedge A(x)\setminus\{x\}\neq\emptyset} \frac{16}{5}p \cdot r_x \\
&\text{(by induction hypothesis).}
\end{aligned}$$

Suppose $f_1 < f_2 < \ldots < f_n$ are all the child types x of z satisfying $A(x) \setminus \{x\} \neq \emptyset$, i.e., x is not a leaf. For each $q = 2, 3, \ldots, n$, since $A(f_q) \setminus \{f_q\} \neq \emptyset$, we have $f_{q-1} \le f_q - 2$ by Proposition 3.1.3. It is easy to see that $\sum_{x\in f(z)\wedge A(x)\setminus\{x\}\neq\emptyset} \frac{16}{5}p \cdot r_x$ is bounded by $\frac{16}{5}p \cdot (\frac{1}{b} + \frac{1}{b^3} + \frac{1}{b^5} + \ldots) \cdot r_z = \frac{16}{5}p \cdot \frac{b}{b^2-1} \cdot r_z = \frac{6}{5}p \cdot r_z$. Hence, eventually, we have

$$\sum_{i\in A(z)\setminus\{z\}} \frac{S(K_i,t)}{g_i} \cdot r_i < p \cdot r_z + \frac{6}{5}p \cdot r_z = \frac{11}{5}p \cdot r_z.$$

Case 2: $t \notin \mathbf{T}$.

Let $\mathcal{R}'_z(t) = \mathcal{R}_z(t) \setminus K_z(t)$ denote the set of jobs active at t which are passed to z's child types $f(z)$. Clearly, $\mathcal{R}'_z(t) = \cup_{x\in f(z)}\mathcal{R}_x(t)$, where $\mathcal{R}_x(t)$ is the set of active jobs at t passed to type x. By definition, $\mathcal{R}'_z(t)$ can also be written as $\mathcal{R}'_z(t) = \{J \in \mathcal{R}_z(t) : m(J) \in A(z) \setminus \{z\} \wedge I(J) \cap \mathbf{T} \neq \emptyset\}$, i.e., each job in $\mathcal{R}'_z(t)$ must contain in its active internal some time instant of $\mathbf{T}$. Since $t \notin \mathbf{T}$, we find the maximum time instant in $\mathbf{T}$ before t and the minimum time instant in $\mathbf{T}$ after t. Then, each job in $\mathcal{R}'_z(t)$ must contain one or both of these two time instants. Thus, we can bound the total cost needed for hosting the jobs $\mathcal{R}'_z(t)$ based on the fact that the time instants in $\mathbf{T}$ are not cost-effective.

Formally, consider two sets of time instants: $S_1 := \text{span}(\mathcal{R}'_z(t)) \cap \mathbf{T} \cap (-\infty, t)$ and $S_2 := \text{span}(\mathcal{R}'_z(t)) \cap \mathbf{T} \cap (t, \infty)$. Figure 2 shows an example of $\mathcal{R}'_z(t)$, $\mathbf{T}$, S_1 and S_2, where $\mathcal{R}'_z(t)$ consists of two jobs J_1 and J_2. Note that S_1 and S_2 are both unions of left closed and right open intervals because $\text{span}(\mathcal{R}'_z(t))$ and $\mathbf{T}$ are.

Let $t_1 := \sup S_1$ if S_1 is nonempty (e.g., $t_1 = b_2$ in Figure 2), and $t_2 := \min S_2$ if S_2 is nonempty (e.g., $t_2 = a_3$ in Figure 2). Observe that $t_1 \le t < t_2$, if both exists. Suppose that $\epsilon > 0$ is an infinitesimal. Note that $t_1 - \epsilon \in \mathbf{T}$ and $t_2 \in \mathbf{T}$. Then, for each job $J \in \mathcal{R}'_z(t)$, either $t_1 - \epsilon$ or t_2 is in $I(J)$, since otherwise $I(J) \cap \mathbf{T}$ would be empty so that $J \in K_z(t)$. Thus, we have $\mathcal{R}'_z(t) \subset \mathcal{R}_z(t_1 - \epsilon) \cup \mathcal{R}_z(t_2)$. Consequently, after the partitioning of jobs among z's child types $f(z)$, we have $\mathcal{R}_x(t) \subset F_{x,t_1-\epsilon} \cup F_{x,t_2}$ for each child type $x \in f(z)$, where $F_{x,t_1-\epsilon}$ or F_{x,t_2} is defined to be empty if S_1 or S_2 is empty. Therefore,

$$\begin{aligned}
&\sum_{x\in f(z)} \frac{S(\mathcal{R}_x,t)}{g_x} \cdot r_x \\
\le&\sum_{x\in f(z)} \frac{S(F_{x,t_1-\epsilon} \cup F_{x,t_2})}{g_x} \cdot r_x \\
\le&\sum_{x\in f(z)} \frac{S(F_{x,t_1-\epsilon})}{g_x} \cdot r_x + \sum_{x\in f(z)} \frac{S(F_{x,t_2})}{g_x} \cdot r_x \\
<&p \cdot r_z + p \cdot r_z = 2p \cdot r_z \quad \text{(since } t_1 - \epsilon, t_2 \in \mathbf{T}\text{).} \qquad (8)
\end{aligned}$$

By definition, $K_x(t) \subset \mathcal{R}_x(t)$, i.e., the set of jobs assigned to type x must be among those passed to type x. Similar to Case 1, we have

$$\sum_{i\in A(z)\setminus\{z\}} \frac{S(K_i,t)}{g_i} \cdot r_i$$

$$= \sum_{x\in f(z)} \left(\frac{S(K_x,t)}{g_x}\cdot r_x + \sum_{i\in A(x)\setminus\{x\}} \frac{S(K_i,t)}{g_i}\cdot r_i \right)$$
$$\le \sum_{x\in f(z)} \frac{S(\mathcal{R}_x,t)}{g_x}\cdot r_x + \sum_{x\in f(z)}\sum_{i\in A(x)\setminus\{x\}} \frac{S(K_i,t)}{g_i}\cdot r_i$$
$$\le \sum_{x\in f(z)} \frac{S(\mathcal{R}_x,t)}{g_x}\cdot r_x + \sum_{x\in f(z)\wedge A(x)\setminus\{x\}\neq\emptyset} \frac{16}{5}p\cdot r_x$$

(by induction hypothesis)

$$\le \sum_{x\in f(z)} \frac{S(\mathcal{R}_x,t)}{g_x}\cdot r_x + \frac{6}{5}p\cdot r_z \quad \text{(same as Case 1)}$$
$$< \frac{16}{5}p\cdot r_z \quad \text{(by equation (8)).}$$

Hence, the inductive case is proven. □

4.2 $ALG_{offline}$ achieves $O(1)$ approximation

We exploit the properties of the cost-per-capacity graph to analyze the $ALG_{offline}$ algorithm. In particular, Proposition 3.1.5 indicates that all the machine types indexed from 1 to any k form a disjoint union of the trees rooted at nodes from the set $T(k)$, where $T(k)$ includes type k and all the younger siblings of k's ancestors. Proposition 3.1.2 says that the machine type at the root of each tree has the lowest normalized cost rate among all the types in the tree. This implies that given any k and any job J within the capacity of a type-k machine, the machine type with the lowest normalized cost rate that can accommodate J must be from $T(k)$. In our analysis, $T(k)$ plays a critical role to bridge the cost of $ALG_{offline}$ and the optimal cost of BSHM. The general idea is as follows. For each time instant, we charge the cost of the machines used by the $ALG_{offline}$ algorithm to only machine types $T(k_{off})$ (where k_{off} is the highest-indexed machine type used by $ALG_{offline}$). We also charge the cost of the optimal one-shot scheduling to only machine types $T(k_{opt})$ (i.e., Proposition 3.2.4, where k_{opt} is the highest-indexed machine type used by the optimal one-shot scheduling). Finally, we establish the connections between the costs of $T(k_{off})$ and $T(k_{opt})$ by carefully analyzing different possible relations between k_{off} and k_{opt} according to the definition of the $ALG_{offline}$ algorithm.

$ALG_{offline}$ schedules the jobs in each K_z into type-z machines by the dual coloring algorithm. The dual coloring algorithm [17] guarantees that the total cost rate of type-z machines used at any time instant t is bounded by $4\cdot\lceil S(K_z,t)/g_z\rceil\cdot r_z$, where $S(K_z,t)$ is the total size of the active jobs in K_z at time t, g_z is the capacity of a type-z machine, r_z is the cost rate of a type-z machine, and 4 comes from the approximation ratio of the dual coloring algorithm. The following theorem shows that the sum of $\lceil S(K_z,t)/g_z\rceil\cdot r_z$ over all machine types z is bounded by $O(1)$ times the cost rate of the optimal one-shot scheduling for all the active jobs $\mathcal{J}(t)$ at time t.

Theorem 4.1. At each time instant t, we have $\sum_{z\in\mathcal{M}}\left\lceil\frac{S(K_z,t)}{g_z}\right\rceil\cdot r_z \le \max\left\{\frac{5}{2}+\frac{24}{5}p, \frac{24}{5}+\frac{3}{2p}\right\}\cdot \mathrm{OPT}_1(\mathcal{J}(t))$.

Proof. Let $k_0 := \max\{m(J) : J\in\mathcal{J}(t)\}$ be the highest-indexed exact machine type among the active jobs at time t. Let $k_{off} := \max\{z : K_z(t)\neq\emptyset\}$ be the highest-indexed machine type used by $ALG_{offline}$ at time t. Let $k_{opt} := \max\{z : w^*(z) > 0\}$ be the highest-indexed machine type used by the optimal machine configuration w^* for one-shot scheduling of $\mathcal{J}(t)$. By Property 4.1 and Theorem 3.1, both k_{off} and k_{opt} are in $P(k_0)$.

Case 1: $k_{opt}\ge k_{off}$.

By Proposition 3.1.5, $\{1,2,\ldots,k_{off}\}\subset\{1,2,\ldots,k_{opt}\} = \cup_{z\in T(k_{opt})}A(z)$. Let $H_z = \{J\in\mathcal{J} : m(J)\in A(z)\}$ denote the set of jobs whose exact machine types are in $A(z)$.

For each type $z\in T(k_{opt})$, we have

$$\sum_{i\in A(z)}\left\lceil\frac{S(K_i,t)}{g_i}\right\rceil\cdot r_i$$
$$\le \frac{S(K_z,t)}{g_z}\cdot r_z + \sum_{i\in A(z)\setminus\{z\}}\frac{S(K_i,t)}{g_i}\cdot r_i + \sum_{i\in A(z)} r_i$$
$$\le \frac{S(K_z,t)}{g_z}\cdot r_z + \frac{16}{5}p\cdot r_z + \sum_{i\in A(z)} r_i \quad \text{(by Property 4.3)}$$
$$\le \frac{S(H_z,t)}{g_z}\cdot r_z + \frac{16}{5}p\cdot r_z + \sum_{i\in A(z)} r_i \quad \text{(since } K_z(t)\subset H_z(t)\text{)}.$$

Since the cost rates are powers of $b=3$, we have $\sum_{z=1}^{k_{opt}} r_z < \sum_{i=0}^{\infty}\frac{1}{b^i}\cdot r_{k_{opt}} = 1/(1-\frac{1}{b})\cdot r_{k_{opt}} = \frac{3}{2}\cdot r_{k_{opt}}$. Therefore,

$$\sum_{z=1}^{k_{off}}\left\lceil\frac{S(K_z,t)}{g_z}\right\rceil\cdot r_z$$
$$= \sum_{z\in T(k_{opt})}\sum_{i\in A(z)}\left\lceil\frac{S(K_i,t)}{g_i}\right\rceil\cdot r_i$$
$$\le \sum_{z\in T(k_{opt})}\left(\frac{S(H_z,t)}{g_z}\cdot r_z + \frac{16}{5}p\cdot r_z + \sum_{i\in A(z)} r_i\right)$$
$$\le \sum_{z\in T(k_{opt})}\frac{S(H_z,t)}{g_z}\cdot r_z + \left(1+\frac{16}{5}p\right)\cdot\sum_{z=1}^{k_{opt}} r_z$$
$$\le \sum_{z\in T(k_{opt})}\frac{S(H_z,t)}{g_z}\cdot r_z + \left(1+\frac{16}{5}p\right)\cdot\frac{3}{2}\cdot r_{k_{opt}}$$
$$\le \left(\frac{5}{2}+\frac{24}{5}p\right)\cdot\mathrm{OPT}_1(\mathcal{J}(t)) \quad \text{(by Proposition 3.2.4).} \quad (9)$$

Case 2: $k_{off} > k_{opt}$, which implies $k_{off}\in P(k_{opt})\setminus\{k_{opt}\}$ (see Figure 3 for an illustration).

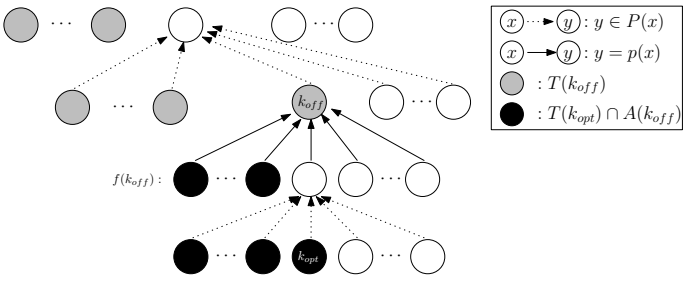

Fig. 3. A diagram illustration for Case 2 in Theorem 4.1

Since $k_{off}\in P(k_{opt})\setminus\{k_{opt}\}$, by Proposition 3.2.3, we have $S(H_{k_{off}},t) < g_{k_{off}}$. By similar arguments to equation (9), we have

$$\sum_{z=1}^{k_{off}}\left\lceil\frac{S(K_z,t)}{g_z}\right\rceil\cdot r_z$$

$$\leq \sum_{z=1}^{k_{off}-1} \left(S(K_z,t) \cdot \frac{r_z}{g_z} + r_z \right) + \left\lceil \frac{S(H_{k_{off}},t)}{g_{k_{off}}} \right\rceil \cdot r_{k_{off}}$$
(since $K_{k_{off}}(t) \subset H_{k_{off}}(t)$)
$$\leq \sum_{z \in T(k_{off}) \setminus \{k_{off}\}} S(K_z,t) \cdot \frac{r_z}{g_z} + \frac{16}{5}p \cdot \sum_{z \in T(k_{off})} r_z + \sum_{z=1}^{k_{off}-1} r_z + r_{k_{off}} \quad \text{(by Property 4.3)}$$
$$\leq \sum_{z \in T(k_{off}) \setminus \{k_{off}\}} S(H_z,t) \cdot \frac{r_z}{g_z} + \left(1 + \frac{16}{5}p\right) \cdot \sum_{z=1}^{k_{off}} r_z$$
(since $K_z(t) \subset H_z(t)$)
$$\leq \sum_{z \in T(k_{off}) \setminus \{k_{off}\}} S(H_z,t) \cdot \frac{r_z}{g_z} + \left(1 + \frac{16}{5}p\right) \cdot \frac{3}{2} \cdot r_{k_{off}}, \tag{10}$$

where $T(k_{off})$ are the grey nodes in Figure 3.

Now we give an upper bound to $r_{k_{off}}$. Since $k_{off} > k_{opt}$ and $k_{opt} \in P(k_0)$, it follows that $k_{off} > k_0$. By Property 4.2, we have

$$p \cdot r_{k_{off}} \leq \sum_{x \in f(k_{off})} S(H_x,t) \cdot \frac{r_x}{g_x},$$

where $f(k_{off})$ is the set of k_{off}'s child types (see Figure 3). Observe that the set of jobs whose exact machine types are in $A(k_{off})$, i.e., $H_{k_{off}}$, can be partitioned into $\{H_x : x \in f(k_{off})\}$ because the exact machine type of each job in $H_{k_{off}}$ must be lower than k_{off}. Furthermore, the partitioning $\{H_x : x \in f(k_{off})\}$ can be further partitioned into $\{H_z : z \in T(k_{opt}) \cap A(k_{off})\}$, where $T(k_{opt}) \cap A(k_{off})$ are the black nodes in Figure 3. This is because the exact machine type of each job in $H_{k_{off}}$ must be lower than or equal to k_{opt}, and each machine type $z \in T(k_{opt}) \cap A(k_{off})$ must be in the tree rooted at some type $x \in f(k_{off})$. For each z and the corresponding x, type z has a normalized cost rate no less than type x. Therefore,

$$\sum_{x \in f(k_{off})} S(H_x,t) \cdot \frac{r_x}{g_x} \leq \sum_{z \in T(k_{opt}) \cap A(k_{off})} S(H_z,t) \cdot \frac{r_z}{g_z}.$$

As a result,

$$r_{k_{off}} \leq \frac{1}{p} \cdot \sum_{z \in T(k_{opt}) \cap A(k_{off})} S(H_z,t) \cdot \frac{r_z}{g_z}. \tag{11}$$

Combining equations (10) and (11), we get

$$\sum_{z=1}^{k_{off}} \left\lceil \frac{S(K_z,t)}{g_z} \right\rceil \cdot r_z$$
$$\leq \sum_{z \in T(k_{off}) \setminus \{k_{off}\}} S(H_z,t) \cdot \frac{r_z}{g_z} + \left(\frac{24}{5} + \frac{3}{2p}\right) \cdot \sum_{z \in T(k_{opt}) \cap A(k_{off})} S(H_z,t) \cdot \frac{r_z}{g_z}$$
$$\leq \max\left\{1, \frac{24}{5} + \frac{3}{2p}\right\} \cdot \sum_{z \in T(k_{opt})} S(H_z,t) \cdot \frac{r_z}{g_z}$$
$$\leq \left(\frac{24}{5} + \frac{3}{2p}\right) \cdot \mathrm{OPT}_1(\mathcal{J}(t)) \quad \text{(by Proposition 3.2.4),}$$

where the second inequality is because $T(k_{off}) \setminus \{k_{off}\}$ and $T(k_{opt}) \cap A(k_{off})$ are disjoint subsets of $T(k_{opt})$. □

The cost of scheduling $\mathcal{J}$ by $ALG_{offline}$ is bounded by $\int_{t \in \mathrm{span}(\mathcal{J})} \left(\sum_{z \in \mathcal{M}} 4 \cdot \lceil S(K_z,t)/g_z \rceil \cdot r_z\right) \mathrm{d}t$, where $\mathrm{span}(\mathcal{J})$ is the time interval(s) in which at least one job in $\mathcal{J}$ is active. By Theorem 4.1, we have

$$\int_{t \in \mathrm{span}(\mathcal{J})} \left(\sum_{z \in \mathcal{M}} 4 \cdot \left\lceil \frac{S(K_z,t)}{g_z} \right\rceil \cdot r_z \right) \mathrm{d}t$$
$$\leq 4 \cdot \max\left\{\frac{5}{2} + \frac{24}{5}p, \frac{24}{5} + \frac{3}{2p}\right\} \cdot \int_{t \in \mathrm{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t))\, \mathrm{d}t$$
$$\leq 4 \cdot \max\left\{\frac{5}{2} + \frac{24}{5}p, \frac{24}{5} + \frac{3}{2p}\right\} \cdot \mathrm{OPT}_{\mathrm{BSHM}}(\mathcal{J})$$
(by equation (1)).

Together with an additional factor of $b = 3$ due to the power of b assumption for cost rates (Section 2), the approximation ratio of $ALG_{offline}$ is $12 \cdot \max\left\{\frac{5}{2} + \frac{24}{5}p, \frac{24}{5} + \frac{3}{2p}\right\}$.

Corollary 4.2. $ALG_{offline}$ achieves an approximation ratio of $12 \cdot \max\left\{\frac{5}{2} + \frac{24}{5}p, \frac{24}{5} + \frac{3}{2p}\right\}$.

Therefore, $ALG_{offline}$ is an $O(1)$-approximation algorithm. Since $p > 0$, the approximation ratio is minimized when $\frac{5}{2} + \frac{24}{5}p = \frac{24}{5} + \frac{3}{2p}$. The best achievable approximation ratio is approximately 78.83 when $p \approx 0.8478$.

5 The Online Setting

5.1 The online algorithm ALG_{online}

We now discuss the online BSHM problem. We say that a machine is *opened* when it receives the first job to process. When all the active jobs end in an open machine, the machine is *closed*. In the online setting, jobs are released when they are to start execution. For simplicity, we assume that jobs are released one at a time. Algorithm 2 shows our online algorithm ALG_{online} for each new job J released. The algorithm iteratively considers the exact machine type $m(J)$ and its ancestor types for processing J (lines 1 and 9). When a machine type z is considered, if there are one or more type-z machines that are open and have available capacity to host job J, J is scheduled into the machine which was opened earliest among these machines (this is known as the First Fit rule) (lines 3-5). If not, we check whether a new type-z machine should be opened. If type z has no parent in the cost-per-capacity graph or opening a new type-z machine does not cause the total cost rate of the open machines for all the descendant types of each z's ancestor $z_a \in P(z) \setminus \{z\}$ to exceed that of one type-z_a machine, a new type-z machine is opened to host job J (lines 6-8). Otherwise, we proceed to consider the parent type $p(z)$ (line 9).

By the definition of ALG_{online}, each job is scheduled into its exact machine type or an ancestor of its exact machine type. Thus, all the jobs scheduled into a machine type z must have exact machine types in the tree rooted at z. For each machine type $z \in \mathcal{M}$ and each time instant $t \in \mathrm{span}(\mathcal{J})$, let $N(z,t)$ denote the number of type-z machines being open at time t.

To analyze the ALG_{online} algorithm, we create a set of artificial jobs to fill up the unused capacities of open machines, in order to establish the relation between the cost of ALG_{online} and the cost of the optimal one-shot scheduling (Section 5.2). For each time instant, we invent a mechanism to charge the cost of the optimal one-shot scheduling to individual jobs within an $O(1)$ factor (Section 5.4). This charging mechanism provides a nice "monotonic" property (adding new jobs never decreases the

Algorithm 2: ALG_{online}

Input: A new job J released at time $I(J)^-$
Output: A machine for processing J

$z \leftarrow m(J)$;
while *true* **do**
  if *there exist type-z machines open at time $I(J)^-$ with available capacity at least $s(J)$* **then**
    among these machines, **return** the machine which was opened earliest;
  end
  if *$p(z)$ does not exist or $\forall z_a \in P(z) \setminus \{z\}$, $\sum_{x \in A(z_a) \setminus \{z_a\}} n_x r_x < r_{z_a} - r_z$ where n_x is the number of type-x machines open at time $I(J)^-$* **then**
    open and **return** a new type-z machine;
  end
  $z \leftarrow p(z)$;
end

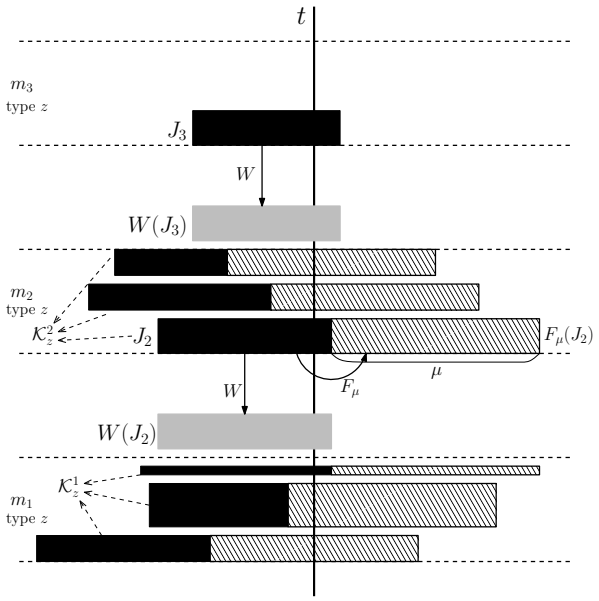

Fig. 4. Artificial jobs fill up capacities of open machines at time t

costs charged to existing jobs, see Theorem 5.8). Based on this property, we show that the cost due to the artificial jobs is bounded by a factor $O(\mu)$ of the cost due to the original jobs (Theorem 5.4), where $\mu := \max_{J \in \mathcal{J}} \text{len}(J) / \min_{J \in \mathcal{J}} \text{len}(J)$ denotes the max/min job length ratio. This leads to the $O(\mu)$ competitive ratio of the ALG_{online} algorithm. Without loss of generality, in the following analysis, we assume that the maximum and minimum job lengths are μ and 1 respectively.

5.2 A set of artificial jobs $\mathcal{R}$

We start by creating some artificial jobs to fill up the capacities of the open machines by ALG_{online}. By "fill up", we mean that the original jobs and the artificial jobs have a total size no less than the machine capacities. We do not physically place the artificial jobs into machines while observing the machine capacities.

For each job $J \in \mathcal{J}$, we create three artificial jobs: J's twin job $W(J)$, μ-extension job $F_\mu(J)$ and 2μ-extension job $F_{2\mu}(J)$. They have the same sizes as J, i.e., $s(W(J)) = s(F_\mu(J)) = s(F_{2\mu}(J)) = s(J)$. Their active intervals are defined as follows:

- $I(W(J)) = I(J)$, i.e., $W(J)$ has the same active interval as J;
- $I(F_\mu(J)) = [I(J)^+, I(J)^+ + \mu)$, i.e., $F_\mu(J)$ extends J's active interval by a period μ;
- $I(F_{2\mu}(J)) = [I(J)^+, I(J)^+ + 2\mu)$, i.e., $F_{2\mu}(J)$ extends J's active interval by a period 2μ.

Let $\mathcal{R} = \{W(J), F_\mu(J), F_{2\mu}(J) : J \in \mathcal{J}\}$ denote all the artificial jobs created. In the following, we show that at each time instant t, the active jobs $\mathcal{J}(t)$ together with the active jobs $\mathcal{R}(t)$ satisfy some properties. Given a time instant t, for each machine type z, let $H_z = \{J \in \mathcal{J}(t) \cup \mathcal{R}(t) : m(J) \in A(z)\}$ denote all the active jobs in $\mathcal{J}(t) \cup \mathcal{R}(t)$ whose exact machine types are in the tree rooted at z. $S(H_z)$ is the total size of the jobs in H_z. Lemma 5.1 says that for each machine type z, if there are multiple type-z machines open at time t, the jobs in H_z fill up the capacities of these machines except one.

Lemma 5.1. For each machine type z such that $N(z,t) > 1$, we have $S(H_z) > (N(z,t) - 1) \cdot g_z$.

Proof. As illustrated in Figure 4, suppose $n = N(z,t) > 1$ type-z machines being open at time t were opened in the order of $m_1, m_2, \ldots, m_n$. Recall that all the jobs scheduled into these machines must have exact machine types in the tree rooted at z. We pick an active job J_i (black rectangle) in each machine m_i at time t. For each $i > 1$, when J_i was scheduled into m_i at its start time $I(J_i)^-$, machine m_{i-1} was also open at that time. Let $\mathcal{K}_z^{i-1}$ denote the set of active jobs in machine m_{i-1} at time $I(J_i)^-$ (black rectangles). By the First Fit scheduling rule, we must have $s(J_i) + S(\mathcal{K}_z^{i-1}) > g_z$ for each $i = 2, \ldots, n$. As a result,

$$\sum_{i=2}^{n} \left(s(J_i) + S(\mathcal{K}_z^{i-1})\right) > (n-1) \cdot g_z.$$

On the other hand, each job J_i has a twin job $W(J_i)$ active at t (grey rectangles), where $W(J_i) \in H_z$. In addition, each job $J \in \mathcal{K}_z^{i-1}$ has a μ-extension job $F_\mu(J)$ which extends J by a period μ (rectangles in back slash pattern). Either J or $F_\mu(J)$ is active at t, since $I(J)^- \le I(J_i)^- \le t$ and $t - I(J)^+ < \text{len}(J_i) \le \mu$. Thus, either J or $F_\mu(J)$ is in H_z. Therefore, the total size of the jobs in H_z must be greater than $(n-1) \cdot g_z$. □

Lemma 5.2 says that for any active job $\hat{J}$ at time t, for each machine type z, if there are multiple type-z machines open at time $I(\hat{J})^-$, the jobs in H_z fill up the capacities of these machines except one.

Lemma 5.2. Take any job $\hat{J} \in \mathcal{J}(t)$. For each machine type z such that $N(z, I(\hat{J})^-) > 1$, we have $S(H_z) > (N(z, I(\hat{J})^-) - 1) \cdot g_z$.

Proof. The proof is similar to Lemma 5.1. As illustrated in Figure 5, take any active job $\hat{J}$ (black rectangle) at time t, and we consider all the type-z machines being open at its start time $I(\hat{J})^-$. Suppose $n = N(z, I(\hat{J})^-) > 1$ type-z machines being open at time $I(\hat{J})^-$ were opened in the order of $m_1, m_2, \ldots, m_n$. All the jobs scheduled into these machines must have exact machine types in the tree rooted at z. We pick an active job J_i (black rectangle) in each machine m_i at time $I(\hat{J})^-$. For each $i > 1$, when J_i was scheduled into m_i at its start time $I(J_i)^-$, machine m_{i-1} was also open at that time. Let $\mathcal{K}_z^{i-1}$ denote the set of active jobs in machine m_{i-1} at time $I(J_i)^-$ (black rectangles). By the

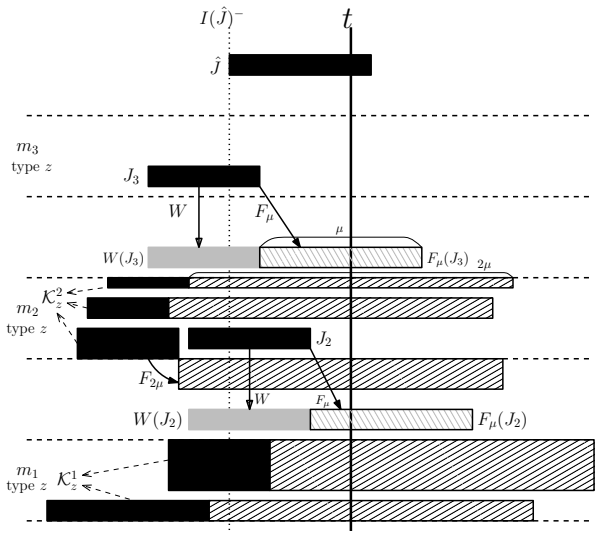

Fig. 5. Artificial jobs fill up capacities of open machines at time $I(\hat{J})^-$

First Fit scheduling rule, we must have $s(J_i) + S(\mathcal{K}_z^{i-1}) > g_z$ for each $i = 2, \ldots, n$. As a result,

$$\sum_{i=2}^{n} \big(s(J_i) + S(\mathcal{K}_z^{i-1})\big) > (n-1) \cdot g_z.$$

Each job J_i has a twin job $W(J_i)$ (grey rectangles) and a μ-extension job $F_\mu(J_i)$ which extends J_i by a period μ (rectangles in back slash pattern). One of these two artificial jobs must be active at time t, since $I(J_i)^- \le I(\hat{J})^- \le t$ and $t - I(J_i)^+ < \mathrm{len}(\hat{J}) \le \mu$. Thus, either $W(J_i)$ or $F_\mu(J_i)$ is in H_z. In addition, each job $J \in \mathcal{K}_z^{i-1}$ has a 2μ-extension job $F_{2\mu}(J)$ which extends J by a period 2μ (rectangles in slash pattern). Either J or $F_{2\mu}(J)$ is active at time t, since $I(J)^- \le I(J_i)^- \le t$ and $t - I(J)^+ < \mathrm{len}(J_i) + \mathrm{len}(\hat{J}) \le 2\mu$. Thus, either J or $F_{2\mu}(J)$ is in H_z. Therefore, the total size of the jobs in H_z must be greater than $(n-1) \cdot g_z$. □

Based on Lemmas 5.1 and 5.2, we can prove that the total cost of the machines used by ALG_{online} at any time t is bounded by $O(1)$ times the cost of the optimal one-shot scheduling for the active jobs $\mathcal{J}(t) \cup \mathcal{R}(t)$.

Theorem 5.3. If $b^3 - 3b^2 - b + 1 > 0$, at each time instant t, we have $\sum_{z\in\mathcal{M}} N(z,t) \cdot r_z \le \max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\} \cdot \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t))$.

Proof. Please refer to Appendix A. □

5.3 A sufficient condition

Note that all the artificial jobs $\mathcal{R}$ can be broken down into $\mathcal{W} = \{W(J) : J \in \mathcal{J}\}$, $\mathcal{F}_\mu = \{F_\mu(J) : J \in \mathcal{J}\}$ and $\mathcal{F}_{2\mu} = \{F_{2\mu}(J) : J \in \mathcal{J}\}$. For any time instant t, the combination of any optimal one-shot scheduling for the active jobs in $\mathcal{J} \cup \mathcal{F}_{2\mu}$ and any optimal one-shot scheduling for the active jobs in $\mathcal{W} \cup \mathcal{F}_\mu$ is a feasible one-shot scheduling for the active jobs in $\mathcal{J} \cup \mathcal{R}$. Recall that $\mathcal{J}(t)$ denotes the active jobs in $\mathcal{J}$ at time t. By optimality,

$$\begin{aligned}&\mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t))\\ &\le \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{F}_{2\mu}(t)) + \mathrm{OPT}_1(\mathcal{W}(t) \cup \mathcal{F}_\mu(t)). \quad (12)\end{aligned}$$

In order to prove that ALG_{online} is an $O(\mu)$-competitive algorithm, it suffices to show the following theorem. Define a function F_d with $d \ge \mu$ which maps each job J in $\mathcal{J}$ to its d-extension job $F_d(J)$ such that $s(F_d(J)) = s(J)$ and $I(F_d(J)) = [I(J)^+, I(J)^+ + d)$, i.e., the job $F_d(J)$ has the same size as J and extends J's active interval by a period d.

Theorem 5.4. Let $\mathcal{H} = \{F_d(J) : J \in \mathcal{J}\}$ with $d \ge \mu$. We have $\int_{t\in\,\mathrm{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{H}(t))\,\mathrm{d}t \le \frac{2b}{b-1}(d+1) \cdot \int_{t\in\,\mathrm{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t))\,\mathrm{d}t$.

Applying Theorem 5.4 by letting $d = \mu$ and 2μ, we have

$$\begin{aligned}&\int_{t\in\,\mathrm{span}(\mathcal{J})} \sum_{z\in\mathcal{M}} N(z,t) \cdot r_z\,\mathrm{d}t\\ &\le \max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\}\\ &\quad \cdot \int_{t\in\mathrm{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t))\,\mathrm{d}t \quad \text{(by Theorem 5.3)}\\ &\le \max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\}\\ &\quad \cdot \int_{t\in\mathrm{span}(\mathcal{J})} \Big(\mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{F}_{2\mu}(t))\\ &\qquad + \mathrm{OPT}_1(\mathcal{W}(t) \cup \mathcal{F}_\mu(t))\Big)\,\mathrm{d}t \quad \text{(by equation (12))}\\ &\le \max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\} \cdot \frac{2b}{b-1}(3\mu+2)\\ &\quad \cdot \int_{t\in\mathrm{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t))\,\mathrm{d}t \quad \text{(by Theorem 5.4)}\\ &\le \max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\} \cdot \frac{2b}{b-1}(3\mu+2)\\ &\quad \cdot \mathrm{OPT}_{\mathrm{BSHM}}(\mathcal{J}) \quad \text{(by equation (1)).}\end{aligned}$$

Together with an additional factor of b due to the power of b assumption for cost rates (Section 2), the competitive ratio of ALG_{online} is $\max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\} \cdot \frac{2b^2}{b-1}(3\mu+2)$.

Theorem 5.5. ALG_{online} achieves a competitive ratio of $\max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\} \cdot \frac{2b^2}{b-1}(3\mu+2)$.

Therefore, ALG_{online} is an $O(\mu)$-competitive algorithm. The best achievable competitive ratio is approximately $59 \cdot (3\mu+2)$ when $b = 7$.

5.4 A modified $O(1)$ approximation of optimal one-shot scheduling

Recall that in Section 3.2, we defined the one-shot scheduling problem for a set of jobs $\mathcal{J}^{1d}$. In order to prove Theorem 5.4, we shall charge the machine cost of the optimal one-shot scheduling to individual jobs in $\mathcal{J}^{1d}$ and have a desired "monotonic" property that the cost charged to each job is non-increasing as the job set $\mathcal{J}^{1d}$ expands. A major challenge to guarantee the "monotonic" property is that the highest-indexed machine type used by the optimal one-shot scheduling is derived from $\mathcal{J}^{1d}$ and it may change as $\mathcal{J}^{1d}$ expands. In fact, the optimal one-shot scheduling is not adequate to address this challenge. In the following, we present a modified machine configuration which is an $O(1)$ approximation of the optimal one-shot scheduling and has the desired "monotonic" property.

Given a set of jobs $\mathcal{J}^{1d}$, recall that the optimal machine configuration for one-shot scheduling may not be unique. In other

TABLE 2
Definition of $\tilde{r}(\mathcal{J}^{1d}, J)$

condition	$\tilde{r}(\mathcal{J}^{1d}, J)$ for each job $J \in H_i$ $(i \in T(k_{opt}) \setminus \{k_{opt}\})$	$\tilde{r}(\mathcal{J}^{1d}, J)$ for each job $J \in H_i$ $(i \in f(k_{opt}))$	$\tilde{r}(\mathcal{J}^{1d}, J)$ for each job $J \in H^h_{k_{opt}}$
$S(H_{k_{opt}}) \geq g_{k_{opt}}$	$s(J) \cdot \frac{r_i}{g_i}$	$s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$	$s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$
$S(H_{k_{opt}}) < g_{k_{opt}}$ and $c \geq r_{k_{opt}}$	$s(J) \cdot \frac{r_i}{g_i}$	$s(J) \cdot \left(\frac{r_i}{g_i} - \left(\frac{r_i}{g_i} - \frac{r_{k_{opt}}}{g_{k_{opt}}}\right) \cdot \alpha^*\right)$	$s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$
$S(H_{k_{opt}}) < g_{k_{opt}}$ and $c < r_{k_{opt}}$	$s(J) \cdot \frac{r_i}{g_i}$	$s(J) \cdot \frac{r_i}{g_i}$	$s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}} \cdot (1 + \beta^*)$

words, the choice of the highest-indexed machine type k_{opt} used by the optimal one-shot scheduling may not be unique. Among all the optimal machine configurations for $\mathcal{J}^{1d}$ described by Theorem 3.1, we choose the one with the *highest* highest-indexed machine type used, and denote its highest-indexed machine type used by k_{opt}. That is, if there exists another optimal machine configuration with the highest-indexed machine type k used, then $k_{opt} \geq k$ must hold. Next, we define the modified machine configuration based on the chosen optimal machine configuration. For ease of reference, Table 2 summarizes how the machine cost of the modified machine configuration is charged to individual jobs in $\mathcal{J}^{1d}$.

For each machine type $i \in \mathcal{M}$, let $H_i = \{J \in \mathcal{J}^{1d} : m(J) \in A(i)\}$ denote the set of jobs whose exact machine types are in the tree $A(i)$ rooted at type i. Then, by Proposition 3.1.5, $\{H_i : i \in T(k_{opt})\}$ is a partitioning of $\mathcal{J}^{1d}$, where $T(k_{opt})$ includes type k_{opt} and all the younger siblings of k_{opt}'s ancestors.

For each $i \in T(k_{opt}) \setminus \{k_{opt}\}$, the jobs in H_i are always accommodated by type-i machines in the modified machine configuration. Hence, we need $\frac{S(H_i)}{g_i}$ type-i machines with a total cost of $S(H_i) \cdot \frac{r_i}{g_i}$. Each job is charged a cost proportional to its size, i.e., each job $J \in H_i$ is charged a cost of $\tilde{r}(\mathcal{J}^{1d}, J) := s(J) \cdot \frac{r_i}{g_i}$. Note that we include the job set $\mathcal{J}^{1d}$ in the notation $\tilde{r}(\mathcal{J}^{1d}, J)$ to indicate that the machine configuration and hence the cost charged to each job is dependent on $\mathcal{J}^{1d}$.

For the jobs in $H_{k_{opt}}$, if their total size is at least the capacity of one type-k_{opt} machine, i.e., $S(H_{k_{opt}}) \geq g_{k_{opt}}$, all of them are accommodated by type-k_{opt} machines in the modified machine configuration. Hence, we need $\frac{S(H_{k_{opt}})}{g_{k_{opt}}}$ type-k_{opt} machines with a total cost of $S(H_{k_{opt}}) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$. Again, each job is charged a cost proportional to its size, i.e., each job $J \in H_{k_{opt}}$ is charged a cost of $\tilde{r}(\mathcal{J}^{1d}, J) := s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$.

If $S(H_{k_{opt}}) < g_{k_{opt}}$, we aim to use one type-k_{opt} machine to accommodate all the jobs in $H_{k_{opt}}$ with a cost of $r_{k_{opt}}$. The cost is charged to the jobs in $H_{k_{opt}}$ as follows. Note that the jobs $H_{k_{opt}}$ can be further partitioned into $H^h_{k_{opt}} := \{J \in \mathcal{J}^{1d} : m(J) = k_{opt}\}$ and $\{H_i : i \in f(k_{opt})\}$ where $f(k_{opt})$ is the set of k_{opt}'s child types. Let $c := S(H^h_{k_{opt}}) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}} + \sum_{i \in f(k_{opt})} S(H_i) \cdot \frac{r_i}{g_i}$ be the cost of using type-k_{opt} machines to accommodate $H^h_{k_{opt}}$ and type-i machines to accommodate each H_i where $i \in f(k_{opt})$.

(i) If $c \geq r_{k_{opt}}$, each job $J \in H^h_{k_{opt}}$ is charged a cost of $\tilde{r}(\mathcal{J}^{1d}, J) := s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$. Note that in this case, we must have $\cup_{i \in f(k_{opt})} H_i \neq \emptyset$ since otherwise $c = S(H_{k_{opt}}) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}} < r_{k_{opt}}$. Each job $J \in H_i$ where $i \in f(k_{opt})$ is charged a cost of $\tilde{r}(\mathcal{J}^{1d}, J) := s(J) \cdot \left(\frac{r_i}{g_i} - \left(\frac{r_i}{g_i} - \frac{r_{k_{opt}}}{g_{k_{opt}}}\right) \cdot \alpha^*\right)$, where $\alpha^* \in [0, 1)$ is given by $\alpha^* = (c - r_{k_{opt}}) / \sum_{i \in f(k_{opt})} S(H_i) \cdot \left(\frac{r_i}{g_i} - \frac{r_{k_{opt}}}{g_{k_{opt}}}\right)$ to ensure that the total cost charged is $r_{k_{opt}}$. Since $\alpha^* \in [0, 1)$, we have $\tilde{r}(\mathcal{J}^{1d}, J) \in \left(s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}, s(J) \cdot \frac{r_i}{g_i}\right]$ for each job $J \in H_i$.

(ii) If $c < r_{k_{opt}}$, each job $J \in H_i$ where $i \in f(k_{opt})$ is charged a cost of $\tilde{r}(\mathcal{J}^{1d}, J) := s(J) \cdot \frac{r_i}{g_i}$. If $H^h_{k_{opt}} \neq \emptyset$, each job $J \in H^h_{k_{opt}}$ is charged a cost of $\tilde{r}(\mathcal{J}^{1d}, J) := s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}} \cdot (1 + \beta^*)$, where $\beta^* \geq 0$ is given by $\beta^* = (r_{k_{opt}} - c) / \left(S(H^h_{k_{opt}}) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}\right)$ to make the total cost charged equal to $r_{k_{opt}}$.

It is easy to see that the charging mechanism described above has the following properties.

Property 5.1. (1) $\tilde{r}(\mathcal{J}^{1d}, J) = s(J) \cdot \frac{r_i}{g_i}$ for each job $J \in H_i$ where $i \in T(k_{opt}) \setminus \{k_{opt}\}$;
(2) $\tilde{r}(\mathcal{J}^{1d}, J) \geq s(J) \cdot \frac{r_{k_{opt}}}{g_{k_{opt}}}$ for each job $J \in H_{k_{opt}}$;
(3) for each type $i \in f(k_{opt})$, $\tilde{r}(\mathcal{J}^{1d}, J) \leq s(J) \cdot \frac{r_i}{g_i}$ for each job $J \in H_i$.

Property 5.2. When $S(H_{k_{opt}}) < g_{k_{opt}}$,
(1) if $c \geq r_{k_{opt}}$ or $H^h_{k_{opt}} \neq \emptyset$, the total cost charged to all the jobs in $H_{k_{opt}}$ is $r_{k_{opt}}$;
(2) if $c < r_{k_{opt}}$ and $H^h_{k_{opt}} = \emptyset$, the total cost charged to all the jobs in $H_{k_{opt}}$ is $c = \sum_{i \in f(k_{opt})} S(H_i) \cdot \frac{r_i}{g_i}$.

With the costs charged to individual jobs, the total cost of the modified machine configuration is given by $\sum_{J \in \mathcal{J}^{1d}} \tilde{r}(\mathcal{J}^{1d}, J)$. Next, we prove that the modified machine configuration is an $O(1)$ approximation of the optimal one-shot scheduling in terms of the total cost.

Theorem 5.6 ($O(1)$ Approximation)**.** $\frac{1}{2} \cdot \sum_{J \in \mathcal{J}^{1d}} \tilde{r}(\mathcal{J}^{1d}, J) \leq \mathrm{OPT}_1(\mathcal{J}^{1d}) \leq \frac{b}{b-1} \cdot \sum_{J \in \mathcal{J}^{1d}} \tilde{r}(\mathcal{J}^{1d}, J)$.

Proof. If $S(H_{k_{opt}}) \geq g_{k_{opt}}$, the total cost of the modified machine configuration is exactly the same as that of the optimal one-shot scheduling. Thus, it suffices to consider the case when $S(H_{k_{opt}}) < g_{k_{opt}}$.

For the left inequality, we have

$$\begin{aligned}\sum_{J \in \mathcal{J}^{1d}} \tilde{r}(\mathcal{J}^{1d}, J) &= \sum_{z \in T(k_{opt})} \sum_{J \in H_z} \tilde{r}(\mathcal{J}^{1d}, J) \\ &\leq r_{k_{opt}} + \sum_{z \in T(k_{opt}) \setminus \{k_{opt}\}} S(H_z) \cdot \frac{r_z}{g_z} \\ &\quad \text{(by Property 5.2 and Property 5.1 (1))} \\ &\leq 2 \cdot \mathrm{OPT}_1(\mathcal{J}^{1d}) \quad \text{(by Proposition 3.2.4).}\end{aligned}$$

For the right inequality, if $c \geq r_{k_{opt}}$ or $H^h_{k_{opt}} \neq \emptyset$, by Property 5.2 (1), the cost of the modified machine configuration due to the

jobs in $H_{k_{opt}}$ is $r_{k_{opt}}$, i.e., $\sum_{J\in H_{k_{opt}}} \tilde{r}(\mathcal{J}^{1d}, J) = r_{k_{opt}}$. Thus, we have

$$\begin{aligned}
&\sum_{J\in\mathcal{J}^{1d}} \tilde{r}(\mathcal{J}^{1d}, J)\\
&= r_{k_{opt}} + \sum_{z\in T(k_{opt})\setminus\{k_{opt}\}} S(H_z)\cdot\frac{r_z}{g_z}\\
&= \max\left\{1, \frac{S(H_{k_{opt}})}{g_{k_{opt}}}\right\}\cdot r_{k_{opt}} + \sum_{z\in T(k_{opt})\setminus\{k_{opt}\}} S(H_z)\cdot\frac{r_z}{g_z}\\
&\geq \mathrm{OPT}_1(\mathcal{J}^{1d}) \quad \text{(by Proposition 3.2.5).}
\end{aligned}$$

It remains to consider the case when $c < r_{k_{opt}}$ and $H^h_{k_{opt}} = \emptyset$. Note that $H^h_{k_{opt}} = \emptyset$ implies all the jobs in $H_{k_{opt}}$ have exact machine types lower than k_{opt}. Thus, machine type k_{opt} must have at least one child type. By Proposition 3.1.3, $k_{opt}-1$ is the highest-indexed child type of k_{opt}. Consider the following machine configuration in which the highest-indexed machine type used is $k_{opt}-1$:

$$w'(z) := \begin{cases} \max\{1, \frac{S(H_z)}{g_z}\} & \text{if } z = k_{opt}-1,\\ \frac{S(H_z)}{g_z} & \text{if } z\in T(k_{opt}-1)\setminus\{k_{opt}-1\},\\ 0 & \text{otherwise.}\end{cases}$$

It is easy to see that w' is a feasible machine configuration for one-shot scheduling. By optimality, we have

$$\sum_{z\in T(k_{opt}-1)} w'(z)\cdot r_z \geq \mathrm{OPT}_1(\mathcal{J}^{1d}). \tag{13}$$

On the other hand,

$$\begin{aligned}
&\sum_{z\in T(k_{opt}-1)} w'(z)\cdot r_z\\
&= \max\left\{1, \frac{S(H_{k_{opt}-1})}{g_{k_{opt}-1}}\right\}\cdot r_{k_{opt}-1}\\
&\quad + \sum_{z\in T(k_{opt}-1)\setminus\{k_{opt}-1\}} S(H_z)\cdot\frac{r_z}{g_z}\\
&\leq r_{k_{opt}-1} + \sum_{z\in T(k_{opt}-1)} S(H_z)\cdot\frac{r_z}{g_z}.
\end{aligned}\tag{14}$$

Since $k_{opt}-1$ is the highest-indexed child type of k_{opt}, by definition, $T(k_{opt}-1) = (T(k_{opt})\setminus\{k_{opt}\})\cup f(k_{opt})$. Thus, by Property 5.1 (1) and Property 5.2 (2), $\sum_{z\in T(k_{opt}-1)} S(H_z)\cdot\frac{r_z}{g_z}$ is exactly the total cost of the modified machine configuration. Therefore, we have

$$\begin{aligned}
&\sum_{J\in\mathcal{J}^{1d}} \tilde{r}(\mathcal{J}^{1d}, J)\\
&= \sum_{z\in T(k_{opt}-1)} S(H_z)\cdot\frac{r_z}{g_z}\\
&\geq \left(\sum_{z\in T(k_{opt}-1)} w'(z)\cdot r_z\right) - r_{k_{opt}-1} \quad \text{(by equation (14))}\\
&\geq \mathrm{OPT}_1(\mathcal{J}^{1d}) - r_{k_{opt}-1} \quad \text{(by equation (13))}\\
&\geq \mathrm{OPT}_1(\mathcal{J}^{1d}) - \frac{1}{b}\cdot r_{k_{opt}}\\
&\geq \mathrm{OPT}_1(\mathcal{J}^{1d}) - \frac{1}{b}\cdot\mathrm{OPT}_1(\mathcal{J}^{1d}) \quad \text{(by Proposition 3.2.4)}\\
&= \frac{b-1}{b}\cdot\mathrm{OPT}_1(\mathcal{J}^{1d}).
\end{aligned}$$

□

An essential step to prove the "monotonic" property of the modified machine configuration is to show that the highest-indexed machine type used is non-decreasing as the job set $\mathcal{J}^{1d}$ expands.

Lemma 5.7. For any two sets of jobs $\mathcal{X}\subset\mathcal{Y}$, we have $k_2\geq k_1$, where k_1 is the highest-indexed machine type used by the modified machine configuration for $\mathcal{X}$, and k_2 is the highest-indexed machine type used by the modified machine configuration for $\mathcal{Y}$.

Proof. Please refer to Appendix B. □

Theorem 5.8 (Monotonic Property)**.** For any two sets of jobs $\mathcal{X}\subset\mathcal{Y}$, we have $\tilde{r}(\mathcal{X}, J)\geq\tilde{r}(\mathcal{Y}, J)$ for each job $J\in\mathcal{X}$.

Proof. It suffices to prove the special case of the theorem in which $\mathcal{Y}\setminus\mathcal{X}$ is a singleton, say J_0. Denote by k_1 the highest-indexed machine type used by the modified machine configuration for $\mathcal{X}$. Denote by k_2 the highest-indexed machine type used by the modified machine configuration for $\mathcal{Y}=\mathcal{X}\cup\{J_0\}$. By Lemma 5.7, $k_2\geq k_1$. Thus, either $k_2\in P(k_1)$ or $k_2\in A(i)$ for some $i\in e(a)$ for some $a\in P(k_1)$, where $P(k_1)$ includes k_1 and all its ancestor types, and $A(i)$ is the tree rooted at type i.

Let $H_z = \{J\in\mathcal{X} : m(J)\in A(z)\}$ denote the set of jobs in $\mathcal{X}$ whose exact machine types are in the tree $A(z)$ rooted at type z.

Recall that $T(k_1)$ includes k_1 and all the younger siblings of k_1's ancestors, and $T(k_2)$ includes k_2 and all the younger siblings of k_2's ancestors. If $k_2\in P(k_1)\setminus\{k_1\}$ (see Figure 6 for an illustration), for each type $z\in T(k_1)\cap A(k_2)$ (the black nodes), for each job $J\in H_z$, it follows from Property 5.1 that $\tilde{r}(\mathcal{X}, J)\geq s(J)\cdot\frac{r_z}{g_z}\geq s(J)\cdot\frac{r_x}{g_x}\geq\tilde{r}(\mathcal{X}\cup\{J_0\}, J)$, where $x\in f(k_2)$ such that $z\in A(x)$. For each type $z\in T(k_1)\setminus A(k_2)\subset T(k_2)\setminus\{k_2\}$ (the grey nodes), for each job $J\in H_z$, we have $\tilde{r}(\mathcal{X}, J) = s(J)\cdot\frac{r_z}{g_z} = \tilde{r}(\mathcal{X}\cup\{J_0\}, J)$ by Property 5.1 (1).

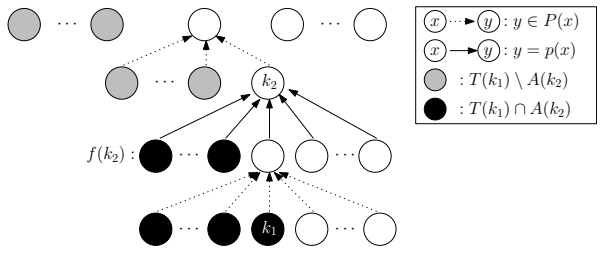

Fig. 6. A diagram illustration for the case of $k_2\in P(k_1)\setminus\{k_1\}$ in Theorem 5.8

If $k_2\in A(i)$ for some $i\in e(a)$ for some $a\in P(k_1)$ (see Figure 7 for an illustration), for each type $z\in T(k_1)\cap A(a)$ (the black nodes[3]), for each job $J\in H_z$, it follows from Property 5.1 (1) and (2) that $\tilde{r}(\mathcal{X}, J)\geq s(J)\cdot\frac{r_z}{g_z}\geq s(J)\cdot\frac{r_a}{g_a} = \tilde{r}(\mathcal{X}\cup\{J_0\}, J)$, because $a\in T(k_2)\setminus\{k_2\}$ by definition. For each type $z\in T(k_1)\setminus A(a)$ (the grey nodes), for each job $J\in H_z$, we have $\tilde{r}(\mathcal{X}, J) = s(J)\cdot\frac{r_z}{g_z} = \tilde{r}(\mathcal{X}\cup\{J_0\}, J)$ by Property 5.1 (1).

If $k_2 = k_1$, we have $m(J_0)\leq k_1$. By Proposition 3.1.5, $\{A(z) : z\in T(k_1)\}$ form a partitioning of $\{1, 2, \ldots, k_1\}$. Note that $A(k_1)$ can be further divided into k_1 and the subtrees rooted at k_1's child types. Thus, there are three possible scenarios: (i) $m(J_0) = k_1$; (ii) $m(J_0)\in A(i^*)$ for some $i^*\in f(k_1)$; and (iii)

3. k_1 can be equal to a. In this case, $T(k_1)\cap A(a) = \{k_1\}$.

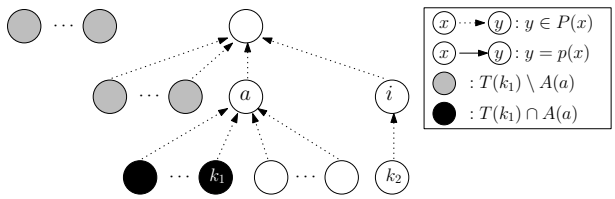

Fig. 7. A diagram illustration for the case of $k_2 \in A(i)$ in Theorem 5.8

$m(J_0) \in A(z)$ for some $z \in T(k_1) \setminus \{k_1\}$. It suffices to consider scenarios (i) and (ii) because $\tilde{r}(\mathcal{X}, J) = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in \mathcal{X}$ in scenario (iii) by definition. In scenarios (i) and (ii), the monotonicity can be verified by mechanically checking the definitions for different cases in Table 2. Note that when a new job is added to $\mathcal{J}^{1d}$, the total size of the jobs in $H_{k_{opt}}$ does not decrease and c does not decrease. Thus, the applicable definition of $\tilde{r}(\mathcal{J}^{1d}, J)$ can either stay in the same row or move from a lower row to an upper row in Table 2. Recall that α^* and β^* are parameters for adjusting the costs charged to individual jobs to tally with the total cost specified. It is easy to see that the cost charged to each job never increases by observing that α^* does not decrease and β^* does not increase. We present the detailed verifications below by enumerating all the possible cases.

Specifically, let $c := S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}} + \sum_{i \in f(k_1)} S(H_i) \cdot \frac{r_i}{g_i}$, where $H^h_{k_1} = \{J \in \mathcal{X} : m(J) = k_1\}$. If $m(J_0) = k_1$ (scenario (i)), there are three cases: (i.1) $S(H_{k_1}) \geq g_{k_1}$; (i.2) $S(H_{k_1}) < g_{k_1}$ and $c \geq r_{k_1}$; and (i.3) $S(H_{k_1}) < g_{k_1}$ and $c < r_{k_1}$ (see the three rows in Table 2). Observe that it suffices to compare only the costs charged to the jobs in H_{k_1} before and after J_0 is added. In case (i.1), clearly $\tilde{r}(\mathcal{X}, J) = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in H_{k_1}$. In case (i.2), if $S(H_{k_1}) + s(J_0) \geq g_{k_1}$, by Property 5.1 (2) and definition, for each job $J \in H_{k_1}$, $\tilde{r}(\mathcal{X}, J) \geq s(J) \cdot \frac{r_{k_1}}{g_{k_1}} = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$. If $S(H_{k_1}) + s(J_0) < g_{k_1}$, on the one hand, by definition, $\tilde{r}(\mathcal{X}, J) = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in H^h_{k_1}$. On the other hand, for each type $i \in f(k_1)$, for each job $J \in H_i$,

$$\tilde{r}(\mathcal{X}, J) = s(J) \cdot \left(\frac{r_i}{g_i} - \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right) \cdot \alpha_1^* \right)$$

and

$$\tilde{r}(\mathcal{X} \cup \{J_0\}, J) = s(J) \cdot \left(\frac{r_i}{g_i} - \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right) \cdot \alpha_2^* \right).$$

It is easy to see that $\alpha_1^* \leq \alpha_2^*$ because

$$\begin{aligned} \alpha_1^* &= \frac{c - r_{k_1}}{\sum_{i \in f(k_1)} S(H_i) \cdot \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right)} \\ &\leq \frac{c + s(J_0) \cdot \frac{r_{k_1}}{g_{k_1}} - r_{k_1}}{\sum_{i \in f(k_1)} S(H_i) \cdot \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right)} = \alpha_2^*. \end{aligned}$$

In case (i.3), for each type $i \in f(k_1)$, for each job $J \in H_i$, by definition and Property 5.1 (3), $\tilde{r}(\mathcal{X}, J) = s(J) \cdot \frac{r_i}{g_i} \geq \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$. For each job $J \in H^h_{k_1}$, by definition,

$$\tilde{r}(\mathcal{X}, J) = s(J) \cdot \frac{r_{k_1}}{g_{k_1}} \cdot (1 + \beta_1^*) \geq s(J) \cdot \frac{r_{k_1}}{g_{k_1}},$$

where

$$\beta_1^* = \frac{r_{k_1} - c}{S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}}}.$$

Therefore, if $S(H_{k_1}) + s(J_0) \geq g_{k_1}$ or $c + s(J_0) \cdot \frac{r_{k_1}}{g_{k_1}} \geq r_{k_1}$, then $\tilde{r}(\mathcal{X}, J) \geq s(J) \cdot \frac{r_{k_1}}{g_{k_1}} = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in H^h_{k_1}$. If $S(H_{k_1}) + s(J_0) < g_{k_1}$ and $c + s(J_0) \cdot \frac{r_{k_1}}{g_{k_1}} < r_{k_1}$, note that for each job $J \in H^h_{k_1}$,

$$\tilde{r}(\mathcal{X}, J) = s(J) \cdot \frac{r_{k_1}}{g_{k_1}} \cdot \frac{r_{k_1} - \sum_{i \in f(k_1)} S(H_i) \cdot \frac{r_i}{g_i}}{S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}}}.$$

Similarly, for each job $J \in H^h_{k_1}$,

$$\tilde{r}(\mathcal{X} \cup \{J_0\}, J) = s(J) \cdot \frac{r_{k_1}}{g_{k_1}} \cdot \frac{r_{k_1} - \sum_{i \in f(k_1)} S(H_i) \cdot \frac{r_i}{g_i}}{s(J_0) \cdot \frac{r_{k_1}}{g_{k_1}} + S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}}}.$$

Clearly, $\tilde{r}(\mathcal{X}, J) \geq \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$.

If $m(J_0)$ is in $A(i^*)$ for some $i^* \in f(k_1)$ (scenario (ii)), similarly there are three cases: (ii.1) $S(H_{k_1}) \geq g_{k_1}$; (ii.2) $S(H_{k_1}) < g_{k_1}$ and $c \geq r_{k_1}$; and (ii.3) $S(H_{k_1}) < g_{k_1}$ and $c < r_{k_1}$ (see the three rows in Table 2). Again, it suffices to compare only the costs charged to the jobs in H_{k_1} before and after J_0 is added. In case (ii.1), clearly $\tilde{r}(\mathcal{X}, J) = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in H_{k_1}$. In case (ii.2), if $S(H_{k_1}) + s(J_0) \geq g_{k_1}$, by Property 5.1 (2) and definition, for each job $J \in H_{k_1}$, $\tilde{r}(\mathcal{X}, J) \geq s(J) \cdot \frac{r_{k_1}}{g_{k_1}} = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$. If $S(H_{k_1}) + s(J_0) < g_{k_1}$, on the one hand, by definition, $\tilde{r}(\mathcal{X}, J) = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in H^h_{k_1}$. On the other hand, for each type $i \in f(k_1)$, for each job $J \in H_i$,

$$\tilde{r}(\mathcal{X}, J) = s(J) \cdot \left(\frac{r_i}{g_i} - \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right) \cdot \alpha_1^* \right)$$

and

$$\tilde{r}(\mathcal{X} \cup \{J_0\}, J) = s(J) \cdot \left(\frac{r_i}{g_i} - \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right) \cdot \alpha_2^* \right).$$

It can be shown that $\alpha_1^* \leq \alpha_2^*$. In fact, by definition,

$$\alpha_1^* = \frac{c - r_{k_1}}{\sum_{i \in f(k_1)} S(H_i) \cdot \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right)}$$

and

$$\alpha_2^* = \frac{c + s(J_0) \cdot \frac{r_{i^*}}{g_{i^*}} - r_{k_1}}{\sum_{i \in f(k_1)} S(H_i) \cdot \left(\frac{r_i}{g_i} - \frac{r_{k_1}}{g_{k_1}} \right) + s(J_0) \cdot \left(\frac{r_{i^*}}{g_{i^*}} - \frac{r_{k_1}}{g_{k_1}} \right)},$$

where $i^* \in f(k_1)$ such that $m(J_0) \in A(i^*)$. Note that for α_1^*, its nominator is less than or equal to its denominator, meanwhile for α_2^*, the term $s(J_0) \cdot \frac{r_{i^*}}{g_{i^*}}$ added to its nominator is actually greater than the term $s(J_0) \cdot \left(\frac{r_{i^*}}{g_{i^*}} - \frac{r_{k_1}}{g_{k_1}} \right)$ added to its denominator. Thus, we must have $\alpha_1^* \leq \alpha_2^*$. In case (ii.3), for each type $i \in f(k_1)$, for each job $J \in H_i$, by definition and Property 5.1 (3), $\tilde{r}(\mathcal{X}, J) = s(J) \cdot \frac{r_i}{g_i} \geq \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$. For each job $J \in H^h_{k_1}$, by definition,

$$\tilde{r}(\mathcal{X}, J) = s(J) \cdot \frac{r_{k_1}}{g_{k_1}} \cdot (1 + \beta_1^*) \geq s(J) \cdot \frac{r_{k_1}}{g_{k_1}},$$

where

$$\beta_1^* = \frac{r_{k_1} - c}{S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}}}.$$

Therefore, if $S(H_{k_1}) + s(J_0) \geq g_{k_1}$ or $c + s(J_0) \cdot \frac{r_{i^*}}{g_{i^*}} \geq r_{k_1}$, then $\tilde{r}(\mathcal{X}, J) \geq s(J) \cdot \frac{r_{k_1}}{g_{k_1}} = \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$ for each job $J \in H^h_{k_1}$.

If $S(H_{k_1}) + s(J_0) < g_{k_1}$ and $c + s(J_0) \cdot \frac{r_{i^*}}{g_{i^*}} < r_{k_1}$, note that for each job $J \in H^h_{k_1}$,

$$\tilde{r}(\mathcal{X}, J) = s(J) \cdot \frac{r_{k_1}}{g_{k_1}} \cdot \frac{r_{k_1} - \sum_{i \in f(k_1)} S(H_i) \cdot \frac{r_i}{g_i}}{S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}}}.$$

Similarly, for each job $J \in H^h_{k_1}$,

$$\begin{aligned}&\tilde{r}(\mathcal{X} \cup \{J_0\}, J)\\ =&s(J) \cdot \frac{r_{k_1}}{g_{k_1}} \cdot \frac{r_{k_1} - \sum_{i \in f(k_1)} S(H_i) \cdot \frac{r_i}{g_i} - s(J_0) \cdot \frac{r_{i^*}}{g_{i^*}}}{S(H^h_{k_1}) \cdot \frac{r_{k_1}}{g_{k_1}}}.\end{aligned}$$

Clearly, $\tilde{r}(\mathcal{X}, J) \geq \tilde{r}(\mathcal{X} \cup \{J_0\}, J)$. □

5.5 Proof of Theorem 5.4

Now, we are ready to finish the proof of Theorem 5.4 which is the last piece in the analysis of ALG_{online}.

Proof of Theorem 5.4. For each job $J \in \mathcal{J}$ and each time instant $t \in I(J)$, define $\tilde{r}_1(J,t) := \tilde{r}(\mathcal{J}(t), J)$. For each job $J \in \mathcal{J}$ and each time instant $t \in [I(J)^+, I(J)^+ + d)$, define $\tilde{r}_2(J,t) := \tilde{r}_1(J,\tau)$, where $\tau := \frac{t - I(J)^+}{d} \operatorname{len}(J) + I(J)^-$ (note that $I(J)^- \leq \tau < I(J)^+$ and hence $\tilde{r}_1(J,\tau)$ is valid, as illustrated in Figure 8).

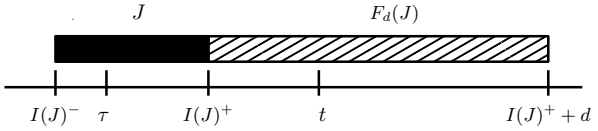

Fig. 8. An illustration of $\tilde{r}_2(J,t) := \tilde{r}_1(J,\tau)$ in Theorem 5.4

We have

$$\begin{aligned}&\sum_{J \in \mathcal{J}} \left(\int_{t \in [I(J)^+, I(J)^+ + d) \cap \operatorname{span}(\mathcal{J})} \tilde{r}_2(J,t)\, \mathrm{d}t \right)\\ \leq &\sum_{J \in \mathcal{J}} \left(\int_{t \in [I(J)^+, I(J)^+ + d)} \tilde{r}_2(J,t)\, \mathrm{d}t \right)\\ = &\sum_{J \in \mathcal{J}} \left(\int_{t \in [I(J)^+, I(J)^+ + d)} \tilde{r}_1(J,\tau)\, \mathrm{d}t \right)\\ &\text{(where } \tau = \frac{t - I(J)^+}{d} \operatorname{len}(J) + I(J)^-\text{)}\\ = &\sum_{J \in \mathcal{J}} \left(\int_{\tau \in I(J)} \tilde{r}_1(J,\tau) \frac{\mathrm{d}t}{\mathrm{d}\tau}\, \mathrm{d}\tau \right)\\ = &\sum_{J \in \mathcal{J}} \left(\int_{\tau \in I(J)} \tilde{r}_1(J,\tau) \frac{d}{\operatorname{len}(J)}\, \mathrm{d}\tau \right)\\ \leq &\, d \cdot \sum_{J \in \mathcal{J}} \left(\int_{\tau \in I(J)} \tilde{r}_1(J,\tau)\, \mathrm{d}\tau \right) \quad \text{(since } 1 \leq \operatorname{len}(J)\text{)}\\ = &\, d \cdot \int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t)} \tilde{r}(\mathcal{J}(t), J)\, \mathrm{d}t\\ &\text{(by swapping } \textstyle\sum \text{ and } \int\text{)}. \end{aligned} \tag{15}$$

Claim: For each time instant $t \in \operatorname{span}(\mathcal{J})$, let $\mathcal{G}_t := \{J \in \mathcal{J} : t - d < I(J)^+ \leq t\}$ be the set of jobs ending in the period $(t-d, t]$. We have $\sum_{J \in \mathcal{J}(t)} \tilde{r}(\mathcal{J}(t), J) + \sum_{J \in \mathcal{G}_t} \tilde{r}_2(J,t) \geq \sum_{J \in \mathcal{J}(t) \cup \mathcal{H}(t)} \tilde{r}(\mathcal{J}(t) \cup \mathcal{H}(t), J)$.

Proof of Claim. By definition, $\mathcal{G}_t$ consists of all the jobs whose d extensions to right cover time t. Since $\mathcal{H} = \{F_d(J) : J \in \mathcal{J}\}$ where $F_d(J)$ is the d extension of job J, the active jobs in $\mathcal{H}$ at time t are exactly the d extensions of the jobs in $\mathcal{G}_t$, i.e., $\mathcal{H}(t) = \{F_d(J) : J \in \mathcal{G}_t\}$, Hence, F_d is actually a 1-1 correspondence between $\mathcal{G}_t$ and $\mathcal{H}(t)$ such that $s(J) = s(F_d(J))$ for each job $J \in \mathcal{G}_t$. Note that for the definition $\tilde{r}$, only the job size matters while the job's active interval does not. Therefore, we have

$$\tilde{r}(\mathcal{J}(t) \cup \mathcal{H}(t), J) = \tilde{r}(\mathcal{J}(t) \cup \mathcal{G}_t, J) \text{ for each job } J \in \mathcal{J}(t),$$

and

$$\tilde{r}(\mathcal{J}(t) \cup \mathcal{H}(t), F_d(J)) = \tilde{r}(\mathcal{J}(t) \cup \mathcal{G}_t, J) \text{ for each job } J \in \mathcal{G}_t.$$

By Theorem 5.8 (monotonic property), $\mathcal{J}(t) \subset \mathcal{J}(t) \cup \mathcal{G}_t$ implies that for each job $J \in \mathcal{J}(t)$, $\tilde{r}(\mathcal{J}(t), J) \geq \tilde{r}(\mathcal{J}(t) \cup \mathcal{G}_t, J)$. It remains to show that for each job $J \in \mathcal{G}_t$, $\tilde{r}_2(J,t) \geq \tilde{r}(\mathcal{J}(t) \cup \mathcal{G}_t, J)$.

Take any job $J \in \mathcal{G}_t$. Let $\tau := \frac{t - I(J)^+}{d} \operatorname{len}(J) + I(J)^-$. We first show that $t - d \leq \tau < t$. Clearly, since $t - d < I(J)^+ \leq t$, we have $\frac{t - I(J)^+}{d} < 1$ and hence $\tau < I(J)^+ \leq t$. It remains to show that $t - \tau \leq d$. Consider the function $f(x) := t - \left(\frac{t - I(J)^+}{d} x + I(J)^+ - x \right)$. Since $1 - \frac{t - I(J)^+}{d} = \frac{d + I(J)^+ - t}{d} > 0$, the function $f(x)$ is increasing with x. Thus, $t - \tau = f(\operatorname{len}(J)) \leq f(d) = d$, since $\operatorname{len}(J) \leq \mu \leq d$.

We then show that $\mathcal{J}(\tau) \subset \mathcal{J}(t) \cup \mathcal{G}_t$. Each job $\hat{J} \in \mathcal{J}(\tau)$ is active at τ. If $\hat{J}$ is also active at t, then $\hat{J} \in \mathcal{J}(t)$. If $\hat{J}$ is not active at t, it must end in the period $(\tau, t]$, i.e., $\tau < I(\hat{J})^+ \leq t$. It follows from $t - d \leq \tau$ that $t - d < I(\hat{J})^+ \leq t$. Hence, $\hat{J} \in \mathcal{G}_t$. Therefore, $\mathcal{J}(\tau) \subset \mathcal{J}(t) \cup \mathcal{G}_t$.

Note that $J \in \mathcal{G}_t$ indicates $t - d < I(J)^+ \leq t$, which implies that $I(J)^- \leq \tau < I(J)^+$. Thus, $J \in \mathcal{J}(\tau) \subset \mathcal{J}(t) \cup \mathcal{G}_t$. By Theorem 5.8 (monotonic property), we have $\tilde{r}_2(J,t) = \tilde{r}_1(J,\tau) = \tilde{r}(\mathcal{J}(\tau), J) \geq \tilde{r}(\mathcal{J}(t) \cup \mathcal{G}_t, J)$. **End of Claim**

It follows that

$$\begin{aligned}&\int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t) \cup \mathcal{H}(t)} \tilde{r}(\mathcal{J}(t) \cup \mathcal{H}(t), J)\, \mathrm{d}t\\ \leq &\int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t)} \tilde{r}(\mathcal{J}(t), J)\, \mathrm{d}t\\ &+ \int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{G}_t} \tilde{r}_2(J,t)\, \mathrm{d}t \quad \text{(by the above Claim)}\\ = &\int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t)} \tilde{r}(\mathcal{J}(t), J)\, \mathrm{d}t\\ &+ \sum_{J \in \mathcal{J}} \left(\int_{t \in [I(J)^+, I(J)^+ + d) \cap \operatorname{span}(\mathcal{J})} \tilde{r}_2(J,t)\, \mathrm{d}t \right)\\ &\text{(by swapping } \textstyle\int \text{ and } \sum\text{)}\\ \leq &\, (d+1) \cdot \int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t)} \tilde{r}(\mathcal{J}(t), J)\, \mathrm{d}t \quad \text{(by equation (15)).}\end{aligned}$$

Eventually, by Theorem 5.6, we have

$$\begin{aligned}&\int_{t \in \operatorname{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{H}(t))\, \mathrm{d}t\\ \leq &\frac{b}{b-1} \cdot \int_{t \in \operatorname{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t) \cup \mathcal{H}(t)} \tilde{r}(\mathcal{J}(t) \cup \mathcal{H}(t), J)\, \mathrm{d}t\end{aligned}$$

$$\leq \frac{b}{b-1}(d+1) \cdot \int_{t \in \mathrm{span}(\mathcal{J})} \sum_{J \in \mathcal{J}(t)} \tilde{r}(\mathcal{J}(t), J)\, \mathrm{d}t$$

$$\leq \frac{2b}{b-1}(d+1) \cdot \int_{t \in \mathrm{span}(\mathcal{J})} \mathrm{OPT}_1(\mathcal{J}(t))\, \mathrm{d}t.$$

□

6 Concluding Remarks

We have studied the general problem of busy-time scheduling on heterogeneous machines. An $O(1)$-approximation offline algorithm and an $O(\mu)$-competitive non-clairvoyant online algorithm have been developed for any sets of jobs and machine types. One future direction is to investigate or improve the tightness of the approximation ratio in the offline setting. Another direction is to study the problem in the clairvoyant online setting where the length of a job is revealed when it is released.

Acknowledgments

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award MOE-T2EP20121-0005) and Academic Research Fund Tier 1 (Award RG112/19).

References

[1] S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.

[2] 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.

[3] Amazon. Amazon EC2, http://aws.amazon.com/ec2/, 2021.

[4] Y. Azar and D. Vainstein. Tight bounds for clairvoyant dynamic bin packing. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 77–86, 2017.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*, volume 53. Cambridge University Press Cambridge, 1998.

[6] N. Buchbinder, Y. Fairstein, K. Mellou, I. Menache, and J. S. Naor. Online virtual machine allocation with lifetime and load predictions. In *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2021.

[7] 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.

[8] 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.

[9] Google. Google Cloud, https://cloud.google.com/, 2021.

[10] 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.

[11] 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, 2005.

[12] 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.

[13] 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.

[14] M. Liu and X. Tang. Analysis of busy-time scheduling on heterogeneous machines. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 340–350, 2021.

[15] 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.

[16] Microsoft. Microsoft Azure, https://azure.microsoft.com/, 2021.

[17] 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.

[18] R. Ren and X. Tang. Busy-time scheduling on heterogeneous machines. In *Proceedings of the 34th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 306–315, 2020.

[19] 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.

[20] R. Ren, Y. Zhu, C. Li, and X. Tang. Interval job scheduling with machine launch cost. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2776–2788, 2020.

[21] 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.

[22] M. M. Tan, R. Ren, and X. Tang. Cloud scheduling with discrete charging units. *IEEE Transactions on Parallel and Distributed Systems*, 30(7):1541–1551, 2019.

[23] 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.

[24] V. V. Vazirani. *Approximation Algorithms*. Springer Science & Business Media, 2013.

[25] 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, 2003.

Mozhengfu Liu received the BSc degree in applied mathematics from National University of Singapore in 2019. He is currently a research assistant in the School of Computer Science and Engineering at Nanyang Technological University, Singapore. His research interests include approximation algorithms and online algorithms.

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 now serving as an associate editor of IEEE Transactions on Cloud Computing. He is a senior member of the IEEE.

APPENDIX A
PROOF OF THEOREM 5.3

Lemma A.1. For each machine type $z \in \mathcal{M}$ and each time instant $t \in \text{span}(\mathcal{J})$, we have $\sum_{i\in A(z)\setminus\{z\}} N(i,t)\cdot r_i < r_z$.

Proof. By the definition of ALG_{online} (lines 6-7), the claim holds obviously for any time instant t when a machine of any type $i \in A(z)\setminus\{z\}$ is opened. For any other time instant t', the total cost rate of the open machines of types $A(z)\setminus\{z\}$ cannot exceed that at the immediate previous time instant t when a machine of any type $i \in A(z)\setminus\{z\}$ is opened. Hence, we have $\sum_{i\in A(z)\setminus\{z\}} N(i,t')\cdot r_i \le \sum_{i\in A(z)\setminus\{z\}} N(i,t)\cdot r_i < r_z$. □

Lemma A.2. Suppose $a_0, a_1, a_2, \ldots$ is a sequence of integers such that each $a_n \in \{0,1,2\}$ and $a_n = 2$ only if $a_{n+1} = 0$. We have $\sum_{n=0,1,2,\ldots} \frac{a_n}{b^n} \le \sum_{n=0,2,4,\ldots} \frac{2}{b^n} = \frac{2b^2}{b^2-1}$ where $b > 1$ is a constant.

Proof.

$$\begin{aligned}\sum_{n=0,1,\ldots}\frac{a_n}{b^n} \le & \sum_{n=0,1,\ldots \text{ s.t. } a_n\le 1}\frac{1}{b^n} \\ & + \sum_{n=0,1,\ldots \text{ s.t. } a_n=2}\left(\frac{1}{b^n}+\frac{1}{b^{n+1}}+\left(\frac{1}{b^n}-\frac{1}{b^{n+1}}\right)\right) \\ \le & \sum_{n=0,1,\ldots}\frac{1}{b^n} + \sum_{n=0,1,\ldots \text{ s.t. } a_n=2}\left(\frac{1}{b^n}-\frac{1}{b^{n+1}}\right).\end{aligned}$$

Note that for each two distinct indices n_1 and n_2 such that $a_{n_1} = a_{n_2} = 2$, we have $|n_1 - n_2| \ge 2$. Thus, the term $\sum_{n=0,1,\ldots \text{ s.t. } a_n=2}\left(\frac{1}{b^n}-\frac{1}{b^{n+1}}\right)$ must be bounded by $\sum_{n=0,2,4,\ldots}\left(\frac{1}{b^n}-\frac{1}{b^{n+1}}\right)$ because $\frac{1}{b^n}-\frac{1}{b^{n+1}}$ decreases with increasing n. Therefore,

$$\begin{aligned}\sum_{n=0,1,\ldots}\frac{a_n}{b^n} \le & \sum_{n=0,1,\ldots}\frac{1}{b^n} + \sum_{n=0,2,4,\ldots}\left(\frac{1}{b^n}-\frac{1}{b^{n+1}}\right) \\ = & \sum_{n=0,2,4,\ldots}\frac{2}{b^n} = \frac{2}{1-\frac{1}{b^2}} = \frac{2b^2}{b^2-1}.\end{aligned}$$

□

Proof of Theorem 5.3. Let k_{on} be the highest-indexed machine type used by ALG_{online} at time t. Let w^* be an optimal machine configuration for one-shot scheduling of $\mathcal{J}(t)\cup\mathcal{R}(t)$, which is chosen by Theorem 3.1. Let $k_{opt} := \max\{z : w^*(z) > 0\}$ be the highest-indexed machine type used by w^*. Recall that $H_z = \{J \in \mathcal{J}(t)\cup\mathcal{R}(t) : m(J) \in A(z)\}$ denotes the set of jobs in $\mathcal{J}(t)\cup\mathcal{R}(t)$ whose exact machine types are in the tree rooted at z.

Case 1: $k_{opt} \ge k_{on}$.

Observe that $\{1,2,\ldots,k_{on}\} \subset \{1,2,\ldots,k_{opt}\} = \cup_{z\in T(k_{opt})} A(z)$ by Proposition 3.1.5. So, we have

$$\begin{aligned}&\text{OPT}_1(\mathcal{J}(t)\cup\mathcal{R}(t)) \\ \ge & \sum_{z\in T(k_{opt})} S(H_z)\cdot\frac{r_z}{g_z} \quad \text{(by Proposition 3.2.4)} \\ \ge & \sum_{z\in T(k_{opt})\wedge N(z,t)>1}(N(z,t)-1)\cdot r_z \quad \text{(by Lemma 5.1).(16)}\end{aligned}$$

Consequently,

$$\begin{aligned}\sum_{z=1}^{k_{on}} N(z,t)\cdot r_z &= \sum_{z\in T(k_{opt})}\sum_{i\in A(z)} N(i,t)\cdot r_i \\ &= \sum_{z\in T(k_{opt})} N(z,t)\cdot r_z + \sum_{z\in T(k_{opt})}\sum_{i\in A(z)\setminus\{z\}} N(i,t)\cdot r_i \\ &\le \sum_{z\in T(k_{opt})\wedge N(z,t)>1}(N(z,t)-1)\cdot r_z + \sum_{z\in T(k_{opt})} r_z \\ &\quad + \sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}\neq\emptyset} r_z \quad \text{(by Lemma A.1)} \\ &\le \text{OPT}_1(\mathcal{J}(t)\cup\mathcal{R}(t)) + \sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}\neq\emptyset} 2\cdot r_z \\ &\quad + \sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}=\emptyset} r_z \quad \text{(by equation (16)).} \qquad (17)\end{aligned}$$

Since the cost rate of each machine type is a power of $b > 1$, for each $z \in T(k_{opt})$, $r_z = \frac{1}{b^n}\cdot r_{k_{opt}}$ for some non-negative integer n. Denote by a_n the coefficient of $\frac{1}{b^n}\cdot r_{k_{opt}}$ in the sum $\sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}\neq\emptyset} 2\cdot r_z + \sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}=\emptyset} r_z$. Observe that each $a_n \in \{0,1,2\}$ and $a_n = 2$ only if $a_{n+1} = 0$. Thus, by Lemma A.2, we have $\sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}\neq\emptyset} 2\cdot r_z + \sum_{z\in T(k_{opt})\wedge A(z)\setminus\{z\}=\emptyset} r_z \le \frac{2b^2}{b^2-1}\cdot r_{k_{opt}}$. Furthermore, by Proposition 3.2.4, $r_{k_{opt}}$ is a lower bound of $\text{OPT}_1(\mathcal{J}(t)\cup\mathcal{R}(t))$. Therefore, it follows from equation (17) that

$$\begin{aligned}\sum_{z=1}^{k_{on}} N(z,t)\cdot r_z &\le \left(1+\frac{2b^2}{b^2-1}\right)\cdot\text{OPT}_1(\mathcal{J}(t)\cup\mathcal{R}(t)) \\ &= \frac{3b^2-1}{b^2-1}\cdot\text{OPT}_1(\mathcal{J}(t)\cup\mathcal{R}(t)).\end{aligned}$$

Case 2: $k_{opt} < k_{on}$.

First, we show that $N(k_{on},t) = 1$. In fact, if $N(k_{on},t) \ge 2$, by Lemma 5.1, we have $S(H_{k_{on}}) > (N(k_{on},t)-1)\cdot g_{k_{on}} \ge g_{k_{on}}$, which contradicts to Proposition 3.2.3.

Furthermore, since $k_{opt} < k_{on}$, the exact machine types of all the jobs in $\mathcal{J}(t)$ must be lower than k_{on}. Take any active job $\hat{J}$ in type-k_{on} machines at time t. Since $\hat{J} \in \mathcal{J}(t)$, we have $m(\hat{J}) \in A(f_1)$ for some type $f_1 \in f(k_{on})$ (see Figure 9). When scheduling $\hat{J}$, ALG_{online} checks if $\hat{J}$ should be scheduled into machine type f_1 before considering type k_{on}. Since $\hat{J}$ is rejected by type f_1, by the definition of ALG_{online}, we know that at $\hat{J}$'s start time $I(\hat{J})^-$, no open type-f_1 machine can accommodate $\hat{J}$ (lines 3-5); and one of the following conditions must hold to prevent opening a new type-f_1 machine (lines 6-8):

(i) $\sum_{z\in A(k_{on})\setminus\{k_{on}\}} N(z, I(\hat{J})^-)\cdot r_z + r_{f_1} \ge r_{k_{on}}$.

(ii) There exists some type $k^\triangle \in P(k_{on})\setminus\{k_{on}\}$ such that $\sum_{z\in A(k^\triangle)\setminus\{k^\triangle,k_{on}\}} N(z, I(\hat{J})^-)\cdot r_z + n_{k_{on}}\cdot r_{k_{on}} + r_{f_1} \ge r_{k^\triangle}$, where $n_{k_{on}}$ represents the number of type-k_{on} machines being open immediately before $\hat{J}$ is scheduled. Note that $n_{k_{on}} = N(k_{on}, I(\hat{J})^-)$ if $\hat{J}$ is scheduled into an open type-k_{on} machine; and $n_{k_{on}} = N(k_{on}, I(\hat{J})^-) - 1$ if $\hat{J}$ is scheduled into a new type-k_{on} machine. However, the latter case cannot happen. If the latter case happens, we have $\sum_{z\in A(k^\triangle)\setminus\{k^\triangle\}} N(z, I(\hat{J})^-)\cdot r_z - r_{k_{on}} + r_{f_1} \ge r_{k^\triangle}$. Since $r_{k_{on}} > r_{f_1}$, it follows that $\sum_{z\in A(k^\triangle)\setminus\{k^\triangle\}} N(z, I(\hat{J})^-)\cdot r_z \ge r_{k^\triangle}$, which contradicts to the definition of ALG_{online} (line 6).

Therefore, to summarize (i) and (ii), there exists some type $k^\triangle \in P(k_{on})$ such that

$$\sum_{z\in A(k^\triangle)\setminus\{k^\triangle\}} N(z, I(\hat{J})^-)\cdot r_z + r_{f_1} \ge r_{k^\triangle}. \qquad (18)$$

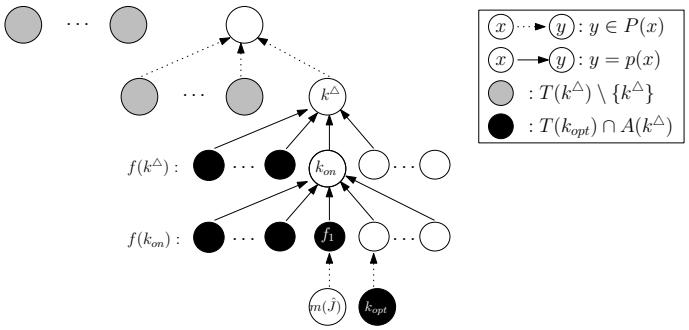

Fig. 9. A diagram illustration for Case 2 in Theorem 5.3

By the definition of ALG_{online}, $H_{k_{on}}$ is nonempty. As illustrated in Figure 9,[4] since $k_{opt} < k_{on} \le k^\triangle$, k_{opt} must be in $A(k_{on}) \setminus \{k_{on}\}$ and thus in $A(k^\triangle) \setminus \{k^\triangle\}$. As a result, $T(k_{opt})$ can be partitioned into $T(k_{opt}) \cap A(k^\triangle)$ and $T(k^\triangle) \setminus \{k^\triangle\}$ (see the black nodes and the grey nodes in Figure 9 respectively). Therefore, we have

$$\begin{aligned}
&\mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t)) \\
&\ge \sum_{z \in T(k_{opt})} S(H_z) \cdot \frac{r_z}{g_z} \quad \text{(by Proposition 3.2.4)} \\
&= \sum_{z \in T(k_{opt}) \cap A(k^\triangle)} S(H_z) \cdot \frac{r_z}{g_z} + \sum_{z \in T(k^\triangle) \setminus \{k^\triangle\}} S(H_z) \cdot \frac{r_z}{g_z} \quad (19)
\end{aligned}$$

Since $k_{opt} < k^\triangle$, all the jobs in $\mathcal{J}(t) \cup \mathcal{R}(t)$ have exact machine types lower than $k^\triangle$. Thus, $H_{k^\triangle}$ can be partitioned into $\{H_x : x \in f(k^\triangle)\}$. In addition, $\{H_x : x \in f(k^\triangle)\}$ can be further partitioned into $\{H_z : z \in T(k_{opt}) \cap A(k^\triangle)\}$. Each machine type $z \in T(k_{opt}) \cap A(k^\triangle)$ must be in the tree rooted at some type $x \in f(k^\triangle)$ and hence type z has a normalized cost rate no less than type x. Therefore, we have

$$\sum_{z \in T(k_{opt}) \cap A(k^\triangle)} S(H_z) \cdot \frac{r_z}{g_z} \ge \sum_{x \in f(k^\triangle)} S(H_x) \cdot \frac{r_x}{g_x}.$$

It follows from equation (19) that

$$\begin{aligned}
&\mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t)) \\
&\ge \sum_{x \in f(k^\triangle)} S(H_x) \cdot \frac{r_x}{g_x} + \sum_{z \in T(k^\triangle) \setminus \{k^\triangle\}} S(H_z) \cdot \frac{r_z}{g_z} \\
&\ge \sum_{x \in f(k^\triangle) \wedge N(x, I(\hat{J})^-) > 1} S(H_x) \cdot \frac{r_x}{g_x} \\
&\quad + \sum_{z \in T(k^\triangle) \wedge N(z,t) > 1} S(H_z) \cdot \frac{r_z}{g_z} \\
&\ge \sum_{x \in f(k^\triangle) \wedge N(x, I(\hat{J})^-) > 1} (N(x, I(\hat{J})^-) - 1) \cdot r_x \\
&\quad + \sum_{z \in T(k^\triangle) \wedge N(z,t) > 1} (N(z,t) - 1) \cdot r_z \\
&\quad \text{(by Lemmas 5.2 and 5.1).} \qquad (20)
\end{aligned}$$

4. In Figure 9, $k^\triangle$ is deliberately made to be the parent node of k_{on} for the illustration purpose, but $k^\triangle$ can be any node in $P(k_{on})$ in general. The same for f_1 and k_{opt}. In general, either of $f_1 \le k_{opt}$ or $f_1 > k_{opt}$ can happen. $f_1 > k_{opt}$ happens when $m(\hat{J}) \in A(k_{opt})$ and $k_{opt} \in A(f_1)$.

For the first item on the right-hand side of (20), equation (18) implies that

$$\begin{aligned}
&\sum_{x \in f(k^\triangle) \wedge N(x, I(\hat{J})^-) > 1} (N(x, I(\hat{J})^-) - 1) \cdot r_x \\
&\ge r_{k^\triangle} - r_{f_1} - \sum_{x \in f(k^\triangle)} \left(r_x + \sum_{z \in A(x) \setminus \{x\}} N(z, I(\hat{J})^-) \cdot r_z \right) \\
&\ge r_{k^\triangle} - r_{f_1} - \left(\sum_{x \in f(k^\triangle)} r_x + \sum_{x \in f(k^\triangle) \wedge A(x) \setminus \{x\} \ne \emptyset} r_x \right) \\
&\quad \text{(by Lemma A.1)} \\
&= r_{k^\triangle} - r_{f_1} - \left(\sum_{x \in f(k^\triangle) \wedge A(x) \setminus \{x\} \ne \emptyset} 2 \cdot r_x + \sum_{x \in f(k^\triangle) \wedge A(x) \setminus \{x\} = \emptyset} r_x \right).
\end{aligned}$$

Since $f_1 \in f(k_{on})$ and $k^\triangle \in P(k_{on})$, we have $r_{f_1} \le \frac{1}{b} \cdot r_{k^\triangle}$. Likewise, for each $x \in f(k^\triangle)$, we have $r_x \le \frac{1}{b} \cdot r_{k^\triangle}$. Then, similar to the discussion in Case 1, by Lemma A.2, we can bound the term $\sum_{x \in f(k^\triangle) \wedge A(x) \setminus \{x\} \ne \emptyset} 2 \cdot r_x + \sum_{x \in f(k^\triangle) \wedge A(x) \setminus \{x\} = \emptyset} r_x$ by $\frac{2b^2}{b^2-1} \cdot \frac{1}{b} \cdot r_{k^\triangle} = \frac{2b}{b^2-1} \cdot r_{k^\triangle}$. Therefore, following the previous equation,

$$\begin{aligned}
&\sum_{x \in f(k^\triangle) \wedge N(x, I(\hat{J})^-) > 1} (N(x, I(\hat{J})^-) - 1) \cdot r_x \\
&\ge \left(1 - \frac{1}{b} - \frac{2b}{b^2 - 1} \right) \cdot r_{k^\triangle} = \frac{b^3 - 3b^2 - b + 1}{b^3 - b} \cdot r_{k^\triangle}. \quad (21)
\end{aligned}$$

Thus, by equations (20) and (21),

$$\begin{aligned}
&\mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t)) \\
&\ge \frac{b^3 - 3b^2 - b + 1}{b^3 - b} \cdot r_{k^\triangle} + \sum_{z \in T(k^\triangle) \wedge N(z,t) > 1} (N(z,t) - 1) \cdot r_z. \quad (22)
\end{aligned}$$

Eventually, we have

$$\begin{aligned}
&\sum_{z=1}^{k_{on}} N(z,t) \cdot r_z = \sum_{z=1}^{k^\triangle} N(z,t) \cdot r_z \\
&= \sum_{z \in T(k^\triangle)} N(z,t) \cdot r_z + \sum_{z \in T(k^\triangle)} \sum_{i \in A(z) \setminus \{z\}} N(i,t) \cdot r_i \\
&\le \sum_{z \in T(k^\triangle) \wedge N(z,t) > 1} (N(z,t) - 1) \cdot r_z + \sum_{z \in T(k^\triangle)} r_z \\
&\quad + \sum_{z \in T(k^\triangle) \wedge A(z) \setminus \{z\} \ne \emptyset} r_z \quad \text{(by Lemma A.1)} \\
&= \sum_{z \in T(k^\triangle) \wedge N(z,t) > 1} (N(z,t) - 1) \cdot r_z + \sum_{z \in T(k^\triangle) \wedge A(z) \setminus \{z\} \ne \emptyset} 2 \cdot r_z \\
&\quad + \sum_{z \in T(k^\triangle) \wedge A(z) \setminus \{z\} = \emptyset} r_z.
\end{aligned}$$

By Lemma A.2, we have $\sum_{z \in T(k^\triangle) \wedge A(z) \setminus \{z\} \ne \emptyset} 2 \cdot r_z + \sum_{z \in T(k^\triangle) \wedge A(z) \setminus \{z\} = \emptyset} r_z \le \frac{2b^2}{b^2-1} \cdot r_{k^\triangle}$. Therefore, following the previous equation, when $b^3 - 3b^2 - b + 1 > 0$, we have

$$\begin{aligned}
&\sum_{z=1}^{k_{on}} N(z,t) \cdot r_z \\
&\le \sum_{z \in T(k^\triangle) \wedge N(z,t) > 1} (N(z,t) - 1) \cdot r_z + \frac{2b^2}{b^2 - 1} \cdot r_{k^\triangle}
\end{aligned}$$

$\leq \frac{\frac{2b^2}{b^2-1}}{\frac{b^3-3b^2-b+1}{b^3-b}} \cdot \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t))$ (by equation (22)).

In summary of Cases 1 and 2, $\sum_{z=1}^{k_{on}} N(z,t) \cdot r_z \leq \max\left\{\frac{3b^2-1}{b^2-1}, \frac{2b^3}{b^3-3b^2-b+1}\right\} \cdot \mathrm{OPT}_1(\mathcal{J}(t) \cup \mathcal{R}(t))$. □

APPENDIX B
PROOF OF LEMMA 5.7

Before proving Lemma 5.7, we consider the following problem first. Given the highest-indexed machine type used k, how to compute the *best machine configuration* for the optimization problem (2)-(5)? By Proposition 3.2.1, all the machine types used by the best machine configuration are from $T(k)$. Let H_z denote the set of jobs whose exact machine types are in the tree rooted at type z. By Proposition 3.1.5, $\{H_z : z \in T(k)\}$ is a partitioning of all the jobs $\mathcal{J}^{1d}$. If there was no constraint that at least one type-k machine must be used, for each type $z \in T(k)$, all the jobs in H_z should be accommodated by type-z machines since z is the type with the lowest cost-per-capacity rate among all the machine types that can accommodate these jobs and have indexes not exceeding k. Due to the constraint that at least one type-k machine must be used, if $S(H_k) < g_k$, some jobs from other H_z's where $z \in T(k) \setminus \{k\}$ will have to be accommodated by the only type-k machine used. To optimize the total cost, these jobs should be selected from the H_z's with the highest indexes z, since the cost-per-capacity rates of the machine types in $T(k) \setminus \{k\}$ is non-decreasing with indexes (Proposition 3.1.4). We refer to the jobs accommodated by the type-k machine (including those from H_z's where $z \in T(k) \setminus \{k\}$) as the *upgraded* part U, and the jobs accommodated by the type-z machines where $z \in T(k) \setminus \{k\}$ as the *non-upgraded* part V. Note that jobs are considered divisible along the size dimension in such partitioning. There may be a job crossing the upgraded and non-upgraded parts, i.e., a portion of this job is in the upgraded part while the remaining portion is in the non-upgraded part. Let $S(U)$ and $S(V)$ denote the total size of the jobs in U and V respectively. By definition, we have $H_k \subset U$ and $S(U) \leq g_k$. We refer to the highest-indexed machine type among the exact machine types of all the jobs in the non-upgraded part V as the boundary machine type z_b (clearly $z_b < k$). In the case that $V = \emptyset$, we define $z_b = 0$.

If a new job J_0 (whose exact machine type $m(J_0)$ is lower than or equal to k) is added to the job set $\mathcal{J}^{1d}$, U and V as well as the cost of the best machine configuration will change. Let $z_0 \in T(k)$ be the index such that the exact machine type $m(J_0)$ is in the tree rooted at type z_0. It is easy to observe that:

(i) If $S(H_k) \geq g_k$, J_0 is always accommodated by type-z_0 machines. Hence, the cost increase is $s(J_0) \cdot \frac{r_{z_0}}{g_{z_0}}$.

(ii) If $S(H_k) < g_k$ and $z_0 \leq z_b$, J_0 is added to the non-upgraded part V and accommodated by type-z_0 machines. Hence, the cost increase is $s(J_0) \cdot \frac{r_{z_0}}{g_{z_0}}$.

(iii) If $S(H_k) < g_k$ and $z_b < z_0 < k$, J_0 is added to the upgraded part U. This may result in the total size of the upgraded part U exceeding the capacity g_k of the type-k machine. Thus, an amount of job size $\max\{S(U) + s(J_0) - g_k, 0\}$ will be moved from the upgraded part U to the non-upgraded part V. Since the cost of the upgraded part U does not change (which equals r_k), the cost increase is given by the cost of the moved portion after the movement. Note that the size of the moved portion is bounded by $s(J_0)$ because $S(U) \leq g_k$. Hence, the moved portion will be accommodated by machine types from $T(k)$ that are no higher than z_0, which have normalized cost rates at most $\frac{r_{z_0}}{g_{z_0}}$. Therefore, the cost increase is bounded by $s(J_0) \cdot \frac{r_{z_0}}{g_{z_0}}$.

(iv) If $S(H_k) < g_k$ and $z_b < z_0 = k$, J_0 is added to the upgraded part U. If $S(H_k) + s(J_0) \leq g_k$, the number of type-k machines used remains at 1. Same as the above case (iii), an amount of job size $\max\{S(U) + s(J_0) - g_k, 0\}$ will be moved from the upgraded part U to the non-upgraded part V and the cost increase is bounded by $s(J_0) \cdot \frac{r_{z_0}}{g_{z_0}} = s(J_0) \cdot \frac{r_k}{g_k}$.

(v) If $S(H_k) < g_k$, $z_b < z_0 = k$ and $S(H_k) + s(J_0) > g_k$, the number of type-k machines used will be increased to $\frac{S(H_k)+s(J_0)}{g_k}$. Only the jobs in the upgraded part U that are from H_z's where $z \in T(k) \setminus \{k\}$ will be moved to the non-upgraded part V. Thus, the amount of job size moved from U to V is $S(U) - S(H_k)$. The cost increase is given by the cost of the moved portion after the movement plus the cost of the additional type-k machines. The former is bounded by $(S(U) - S(H_k)) \cdot \frac{r_k}{g_k}$ (since the moved portion will be accommodated by machine types with normalized cost rates at most $\frac{r_{z_0}}{g_{z_0}} = \frac{r_k}{g_k}$), while the latter is $\left(\frac{S(H_k)+s(J_0)}{g_k} - 1\right) \cdot r_k$. Thus, the cost increase is bounded by $(S(U) - S(H_k)) \cdot \frac{r_k}{g_k} + \left(\frac{S(H_k)+s(J_0)}{g_k} - 1\right) \cdot r_k = (S(U) + s(J_0) - g_k) \cdot \frac{r_k}{g_k} \leq s(J_0) \cdot \frac{r_k}{g_k} = s(J_0) \cdot \frac{r_{z_0}}{g_{z_0}}$. Alternatively, we can also conceptually consider the part of J_0 accommodated by the additional type-k machines, i.e., $S(H_k) + s(J_0) - g_k$, as being "moved" to machine type $z_0 = k$. Then, the collective "moved" amount is $(S(U) - S(H_k)) + (S(H_k) + s(J_0) - g_k) = S(U) + s(J_0) - g_k$. In this way, cases (iii), (iv) and (v) can be unified as moving an amount of job size $\max\{S(U) + s(J_0) - g_k, 0\}$ to machine types with normalized cost rates at most $\frac{r_{z_0}}{g_{z_0}}$ and the cost increase is given by the cost of the moved portion after the movement.

(vi) In summary of all the cases above, the cost increase is bounded by $s(J_0) \cdot \frac{r_{z_0}}{g_{z_0}}$.

Proof of Lemma 5.7. It suffices to prove the special case of the lemma in which $\mathcal{Y} \setminus \mathcal{X}$ is a singleton, say J_0. Denote by k_1 the highest-indexed machine type used by the modified machine configuration for $\mathcal{X}$. Denote by k_2 the highest-indexed machine type used by the modified machine configuration for $\mathcal{Y} = \mathcal{X} \cup \{J_0\}$. Obviously, $m(J_0) \leq k_2$. If $m(J_0) \geq k_1$, we immediately have $k_2 \geq m(J_0) \geq k_1$. Thus, to show that $k_2 \geq k_1$, it suffices to consider the case when $m(J_0) < k_1$.

By Theorem 3.1, k_1 is in $P(k_0)$ where $k_0 := \max\{m(J) : J \in \mathcal{X}\}$ is the highest-indexed exact machine type among all the jobs in $\mathcal{X}$. If $m(J_0) \leq k_0$, k_2 is also in $P(k_0)$. On the other hand, if $m(J_0) > k_0$, k_2 is in $P(m(J_0))$. Since $k_0 < m(J_0) < k_1$ and k_0 is in the tree rooted at k_1, by Proposition 3.1.3, $m(J_0)$ is also in the tree rooted at k_1. Thus, k_1 is also in $P(m(J_0))$. In summary, k_1 and k_2 are either both in $P(k_0)$ or both in $P(m(J_0))$. Therefore, either $k_1 = k_2$ or one of them is an ancestor of the other in the cost-per-capacity graph.

We prove $k_2 \geq k_1$ by contradiction. Let w_1 be the optimal machine configuration for jobs $\mathcal{X}$, in which the highest-indexed machine type used is k_1; and w_1' be the best machine configuration for jobs $\mathcal{X} \cup \{J_0\}$, assuming the highest-indexed machine type used is k_1. Let w_2 be the best machine configuration for jobs $\mathcal{X}$, assuming the highest-indexed machine type used is k_2; and w_2' be the optimal machine configuration for jobs $\mathcal{X} \cup \{J_0\}$, in which the highest-indexed machine type used is k_2.

If $k_1 > k_2$, we are to prove that the cost difference between w_1 and w_1' is no more than the cost difference between w_2 and w_2'. As a result, the cost of the machine configuration w_1' is lower than or equal to the cost of the optimal machine configuration w_2' for scheduling jobs $\mathcal{X} \cup \{J_0\}$, and hence w_1' is optimal. This contradicts to the choice of the highest-indexed machine type k_2 used by the modified machine configuration for jobs $\mathcal{X} \cup \{J_0\}$, which is defined to be the highest among the highest-indexed machine types of all the optimal machine configurations for jobs $\mathcal{X} \cup \{J_0\}$.

Let $z_2 \in T(k_2)$ be the index such that the exact machine type $m(J_0)$ is in the tree rooted at type z_2, and let $z_1 \in T(k_1)$ be the index such that the exact machine type $m(J_0)$ is in the tree rooted at type z_1. Note that $k_1 > k_2$ implies that k_1 is an ancestor of k_2 in the cost-per-capacity graph. Thus, we have either $z_2 = z_1 < k_1$ (if $m(J_0) \notin A(k_1)$) or $z_2 < z_1 = k_1$ (if $m(J_0) \in A(k_1)$). In either case, it holds that $\frac{r_{z_1}}{g_{z_1}} \leq \frac{r_{z_2}}{g_{z_2}}$.

Let H_{k_2} denote the set of jobs in $\mathcal{X}$ whose exact machine types are in the tree rooted at type k_2. If $S(H_{k_2}) \geq g_{k_2}$, by observation (i) above, the cost difference between w_2 and w_2' is $s(J_0) \cdot \frac{r_{z_2}}{g_{z_2}}$. On the other hand, by observation (vi) above, the cost difference between w_1 and w_1' is at most $s(J_0) \cdot \frac{r_{z_1}}{g_{z_1}} \leq s(J_0) \cdot \frac{r_{z_2}}{g_{z_2}}$.

If $S(H_{k_2}) < g_{k_2}$, since $k_1 \in P(k_2) \setminus \{k_2\}$, we must have $S(H_{k_1}) < g_{k_1}$ by Proposition 3.2.3, where H_{k_1} denotes the set of jobs in $\mathcal{X}$ whose exact machine types are in the tree rooted at type k_1. Let U_1 and V_1 be the upgraded and non-upgraded parts of w_1, and let U_2 and V_2 be the upgraded and non-upgraded parts of w_2. It must hold that the upgraded part U_1 has a total size no less than the upgraded part U_2, i.e., $S(U_1) \geq S(U_2)$, because $g_{k_1} > g_{k_2}$. This implies that the boundary machine type z_{b_1} of w_1 is no higher than the boundary machine type z_{b_2} of w_2, i.e., $z_{b_1} \leq z_{b_2}$. If $z_2 \leq z_{b_2}$, by observation (ii) above, the cost difference between w_2 and w_2' is $s(J_0) \cdot \frac{r_{z_2}}{g_{z_2}}$, and by observation (vi) above, the cost difference between w_1 and w_1' is at most $s(J_0) \cdot \frac{r_{z_1}}{g_{z_1}} \leq s(J_0) \cdot \frac{r_{z_2}}{g_{z_2}}$. If $z_{b_2} < z_2$, we have $z_{b_1} \leq z_{b_2} < z_2 \leq z_1$. By observations (iii), (iv) and (v) above, J_0 is added to U_1 and U_2 of w_1 and w_2 respectively, while an amount of job size $\max\{S(U_1) + s(J_0) - g_{k_1}, 0\}$ and $\max\{S(U_2) + S(J_0) - g_{k_2}, 0\}$ needs to be moved from U_1 to V_1 and from U_2 to V_2 respectively. Thus, the cost difference between w_1 and w_1' (and between w_2 and w_2') is the total cost of the moved portion after the movement. It is easy to see that the moved portions satisfy $\max\{S(U_1) + s(J_0) - g_{k_1}, 0\} \leq \max\{S(U_2) + s(J_0) - g_{k_2}, 0\}$. In fact, if $S(U_2) < g_{k_2}$, we must have $S(U_2) = S(\mathcal{X}) = S(U_1)$ and the claim follows from $g_{k_1} > g_{k_2}$. On the other hand, if $S(U_2) = g_{k_2}$, the claim follows from $\max\{S(U_1) + s(J_0) - g_{k_1}, 0\} \leq s(J_0) = \max\{S(U_2) + s(J_0) - g_{k_2}, 0\}$. Moreover, since $z_{b_1} \leq z_{b_2}$, each unit moved from U_1 to V_1 must be moved to a machine type indexed no higher than that of each unit moved from U_2 to V_2, so the former must have a cost-per-capacity rate no higher than the latter. Hence, the cost increase from w_1 to w_1' must be no more than the cost increase from w_2 to w_2'.

□