Renting Servers for Multi-Parameter Jobs in the Cloud

Yaqiao Li
Concordia University, CSSE
yaqiao.li@concordia.ca

Mahtab Masoori
Concordia University, CSSE
mahtab.masoori@concordia.ca

Lata Narayanan
Concordia University, CSSE
lata.narayanan@concordia.ca

Denis Pankratov
Concordia University, CSSE
denis.pankratov@concordia.ca

ABSTRACT

We study the Renting Servers in the Cloud problem (*RSiC*) in multiple dimensions. In this problem, a sequence of multi-parameter jobs must be scheduled on servers that can be rented on-demand. Each job has an arrival time, a finishing time, and a multi-dimensional size vector that specifies its resource demands. Each server has a multi-dimensional capacity and jobs can be scheduled on a server as long as in each dimension the sum of sizes of jobs does not exceed the capacity of the server in that dimension. The goal is to minimize the total rental time of servers needed to process the job sequence.

AnyFit algorithms do not rent new servers to accommodate a job unless they have to. We introduce a sub-family of *AnyFit* algorithms, which we call monotone *AnyFit* algorithms. We show that monotone *AnyFit* algorithms have a tight competitive ratio of $\Theta(d\mu)$, where d is the dimension of the problem and μ is the ratio between the maximum and minimum duration of jobs in the input sequence. We also show that upper bounds for the *RSiC* problem obey the direct-sum property with respect to dimension d, that is we show how to transform 1-dimensional algorithms for *RSiC* to work in the d-dimensional setting with competitive ratio scaling by a factor of d. As a corollary, we obtain an $O(d\sqrt{\log \mu})$ upper bound for d-dimensional clairvoyant *RSiC*. We also establish a lower bound of $\widetilde{\Omega}(d\mu)$ for both deterministic and randomized algorithms for d-dimensional non-clairvoyant *RSiC*, under the assumption that $\mu \le \log d - 2$.

Lastly, we propose a natural greedy algorithm, which we call *Greedy*. This is a clairvoyant algorithm that belongs to the monotone *AnyFit* family of algorithms, thus, it has competitive ratio $\Theta(d\mu)$. Our experimental results indicate that *Greedy* performs better or as well as all other previously proposed algorithms, for almost all the settings of arrival rates and values of μ and d that we implemented.

1 INTRODUCTION

One of the most famous problems in online computation is the *bin packing* problem which has received a lot of attention among researchers [6, 7]. Given a set of items with a positive size, and a bin capacity, the objective is to pack all the items in the minimum possible number of bins such that the total size of items assigned to a bin does not exceed the bin capacity. For simplicity, it is generally assumed that the item sizes lie between 0 and 1 and the bins all have unit capacity. This problem is online in the sense that the items come in a sequential manner and the algorithm has to place a new item in a bin without having any knowledge about the upcoming items. Dynamic bin packing is a generalization of the bin packing problem [8], in which items not only have an arrival time and a size, but also a *duration*. The objective function is the same as the classic bin packing problem; minimizing the total number of bins that is used to pack all the items. Dynamic bin packing has been extensively used to model various resource allocation problems, highlighting its adaptability in optimizing packing scenarios for dynamic settings [13, 16, 25, 26].

Li et al. [19] introduced *MinUsageTime Dynamic Bin Packing*, a new variant of Dynamic Bin Packing. The problem is also known as *Renting Servers in the Cloud (RSiC)*, and is primarily motivated by job allocation to servers in the cloud. For example, users make requests for virtual machines (VMs) with specific requirements to a cloud service provider such as Microsoft Azure, which then has to assign VMs to physical servers with sufficient capacity. The power and other costs incurred by the service provider are directly proportional to the total duration that servers are kept active. Optimal assignment decisions can reduce fragmentation which can result in dramatic cost savings [10]. As another application, cloud gaming companies such as GaiKai, OnLive, and StreamMyGame rent servers from public cloud companies and are charged using a pay-as-you-go model. A customer's request to play a game is assigned to one of the rented servers that has enough capacity to serve the request. The rental cost paid by the gaming company is directly proportional to the duration of time that the servers are rented.

These situations are modeled by the *RSiC* problem, which has been studied in both non-clairvoyant and clairvoyant settings, see, e.g. [2, 15, 23]. Recently, Murhekar et al. [22] initiated the study of *RSiC* for multi-parameter jobs. In this setting, jobs with resource requirements for multiple parameters (e.g. number of GPUs, memory, network bandwidth etc.) arrive to the system in an online manner and must be assigned by the algorithm to servers with fixed capacity along each of these dimensions. In the non-clairvoyant setting, the arrival time of a job and its resource requirements, given as a *size vector*, are revealed to the algorithm when the job arrives, but its duration is only known when it departs. In the clairvoyant setting, the finishing time of a job is also known to the algorithm when it arrives. Jobs must be assigned to servers immediately after arriving, and the algorithm's decisions are irrevocable, as the cost of moving a job mid-execution to another server is assumed to be prohibitive. The objective of the *RSiC* problem is to minimize the total cost of all rented servers, where the cost of a server is proportional to the duration for which it is rented/utilized.

The performance of online algorithms is measured by the notion of *competitive ratio*. This ratio represents the worst-case comparison, considering all inputs, between the cost achieved by the online algorithm and the cost achieved by an optimal offline solution that has complete knowledge of the entire input instance in advance.

Our contributions

In this paper, we study the d-dimensional *RSiC* problem, when job sizes are d-dimensional vectors and the size of a job in any dimension is normalized to lie between 0 and 1, and the server capacity in each dimension is 1. We consider both the clairvoyant and non-clairvoyant versions of the problem. Our main results in this paper are summarized below:

- We introduce a subset of *AnyFit* algorithms called monotone *AnyFit* algorithms and propose a new clairvoyant algorithm within this category called *Greedy*, which assigns a new job to the server with enough remaining capacity that would incur the *least additional cost.* We prove that all monotone *AnyFit* algorithms including *Greedy* have a competitive ratio of $3d\mu + 1$, where μ is the ratio between the maximum and minimum duration of jobs in the input sequence. The proof uses a new technique compared to those used in [15] and [23].
  We remark here that the proof of a $6\mu + 8$ upper bound on the competitive ratio of *MTF* given in [15] also works for *Greedy* for 1-dimensional *RSiC*. However, we are able to derive a better upper bound with our technique, which moreover generalizes to any monotone *AnyFit* algorithm and d-dimensional *RSiC*. The upper bound is tight as a lower bound of $\Omega(d\mu)$ was shown on the competitive ratio of *AnyFit* algorithms in [22].
- We demonstrate a very general direct sum property of the *RSiC* problem by showing how to transform any algorithm ALG for dimension 1 to a corresponding algorithm $\text{ALG}^{\oplus d}$ for dimension d, with competitive ratio scaling *exactly* by a factor of d. As a corollary, we obtain the first clairvoyant algorithm for d-dimensional *RSiC*, with competitive ratio $\Theta(d\sqrt{\log \mu})$.
- We adapt, for the first time, an online graph coloring lower bound construction in [11] to prove lower bounds for d-dimensional *RSiC*. Specifically, we prove a lower bound of $\widetilde{\Omega}(d\mu)$ for both deterministic and randomized algorithms for d-dimensional non-clairvoyant *RSiC* when $\mu \leq \log d - 2$, and a lower bound of $\widetilde{\Omega}(d)$ for the clairvoyant case. Connections between online vector bin packing and online graph coloring have been explored in [1, 17].
- We conduct experiments in the average-case scenario, evaluating nearly all existing algorithms for *RSiC* using randomly generated synthetic data. Our findings show that the *Greedy* algorithm outperforms other algorithms, whether clairvoyant or non-clairvoyant, in the vast majority of cases.

Organization

The related work is summarized in Section 2. Formal definitions and notation are introduced in Section 3. In Section 4, we define a new subclass of the *AnyFit* algorithms for which we call the monotone *AnyFit* algorithms, and prove an upper bound of the competitive ratio using a new proof technique. In Section 5, we describe a direct sum property of algorithms for *RSiC* and obtain as a corollary an upper bound for d-dimensional clairvoyant *RSiC*. In Section 6, we prove lower bounds for deterministic and randomized algorithms for *RSiC* in both clairvoyant and non-clairvoyant settings by adapting techniques in online graph coloring. The experimental results are presented in Section 7. Finally, we draw the conclusion and list some open problems in Section 8.

2 PREVIOUS WORK

Bin packing is one of the best-studied problems in combinatorial optimization, and extensive research has been done on proving tight bounds on the competitive ratio of online algorithms for the problem [6, 7]. A well-studied class of algorithms is *AnyFit*, in which the guiding principle is to always use existing bins when possible. Well-known algorithms such as *FirstFit*, *BestFit*, and *WorstFit* fall into this class. A rather large class of *AnyFit* algorithms has been shown to have a tight competitive ratio of 1.7 for the bin packing problem [14]. The best known upper bound for bin packing is achieved by Advanced Harmonic [3] and has a competitive ratio of 1.57829. The current best lower bound is 1.54278, as shown in [4].

In the online *vector bin packing* problem, item sizes are given as d-dimensional vectors and bins have capacity 1 in each dimension. The online algorithm assigns items to bins such that the capacity constraints are respected in every dimension. Garey et al. [9] showed that Generalized *FirstFit* has a competitive ratio at most $d + 0.7$. Azar et al. [1], using connections to online graph coloring, proved a lower bound $\Omega(d^{1-\epsilon})$ for any algorithm.

Coffman et al. [8] introduced *dynamic bin packing*: items have size as well as duration, and the objective function is still to minimize the number of bins. Wong et al. [27] established the current best lower bound 2.66 of the competitive ratio for any algorithm, they also showed that *FirstFit* has an upper bound 2.897. The performance of various algorithms such as *FirstFit*, modified *FirstFit*, *BestFit* and *WorstFit* have been studied in [5, 12, 27] etc.

The 1-dimensional and d-dimensional *RSiC* problems were introduced in Li et al. [19] and Murhekar et al. [22], respectively. Except [22], all existing literature on *RSiC* studies only dimension 1. For non-clairvoyant 1-dimensional *RSiC*, Li et al. [19] proved a lower bound of $\mu + 1$ for *AnyFit* algorithms. Later, Kamali and Lopez-Ortiz [15] showed that μ is in fact a lower bound for any deterministic algorithm. The current best upper bound for *FirstFit* is $\mu + 3$, which was proved in the PhD thesis of Ren [23]. Masoori et al. [20, 21] studied the performance ratio of *FirstFit* for uniform-duration jobs and for restricted servers and showed better bounds on the competitive ratio of *FirstFit* in these settings. Kamali and Lopez-Ortiz [15] considered the *NextFit* algorithm and proved that it has competitive ratio of $2\mu + 1$. They also introduced a new *AnyFit* algorithm called the *Move-to-Front* (*MTF*), which places the next job in the *most recently used* bin. They proved that the competitive ratio of *MTF* is at most $6\mu + 7$. The clairvoyant setting has been studied in [2, 24]. In particular, Azar et al. [2] proposed the hybrid algorithm (*HA*) , and proved a tight bound of $\Theta(\sqrt{\log \mu})$ on its competitive ratio.

Murhekar et al. [22] initiated the study of non-clairvoyant d-dimensional *RSiC*. They proved that *MTF* has an upper bound $(2\mu + 1)d + 1$, which in particular improves the previous upper bound of *MTF* in [15] for $d = 1$ to $2\mu + 2$. They also generalized various upper and lower bounds of algorithms such as *FirstFit* and *NextFit* from dimension 1 to dimension d. In particular, they showed a lower bound of $d(\mu + 1)$ for *AnyFit* algorithms.

Finally, [15, 22, 24] also presented experiments on random inputs to compare different algorithms and show various non-trivial phenomena. In particular, Kamali and Lopez-Ortiz [15] demonstrated that *MTF* in general performs the best among all known non-clairvoyant algorithms at dimension 1, which was recently further confirmed by experiments for dimension d in Murhekar et al. [22].

To the best of our knowledge, the study of clairvoyant d-dimensional *RSiC* and randomized algorithms for *RSiC* is missing in the literature. This work partially fills this gap.

3 NOTATION AND PRELIMINARIES

For $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. The L_∞ norm of a vector $v \in \mathbb{R}^d_{\geq 0}$ is denoted by $\|v\|_\infty$ and equals $\max_{j \in [d]} v_j$. We shall make frequent use of the following classical inequalities:

Proposition 3.1. *For any set of vectors $v_1, v_2, \cdots, v_n \in \mathbb{R}^d_{\geq 0}$, we have the following:*

$$\left\| \sum_{i=1}^{n} v_i \right\|_\infty \leq \sum_{i=1}^{n} \|v_i\|_\infty \leq d \cdot \left\| \sum_{i=1}^{n} v_i \right\|_\infty$$

The input to a d-dimensional *RSiC* problem is $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ – a list of jobs where each job $\sigma_i \in \sigma$ is a triple (a_i, f_i, s_i), denoting the arrival time, finishing time, and size/resource demand of σ_i. We assume that the jobs are presented to the algorithm in the order of arrival, that is, $a_1 \leq a_2 \leq \cdots \leq a_n$. We refer to $f_i - a_i$ as the *duration* of the job σ_i. The duration of every job lies between 1 and μ, that is, $1 \leq f_i - a_i \leq \mu$ for every job σ_i. The *utilization* of the job σ_i, denoted by $\text{util}(\sigma_i)$, is defined as $\text{util}(\sigma_i) = (f_i - a_i) \cdot \|s_i\|_\infty$.

Each job has multi-dimensional resource demands, i.e., $s_i \in (0, 1)^d$ where s_i^j denotes the size of the job σ_i in the j^{th} dimension for $j \in [d]$. We assume that an algorithm for *RSiC* has access to a supply of identical servers of capacity 1 in each dimension, i.e, the size of each server is 1^d. Thus, for every time t the combined size of jobs in a particular dimension assigned to a particular server and active at time t must not exceed 1. We denote the sum of sizes of jobs active at time t by $s(\sigma, t)$, i.e., $s(\sigma, t) = \sum_{i: a_i \leq t < f_i} s_i$. For a server S alive at time t, we use $S(t)$ to denote the sum of sizes of all jobs that have been scheduled on S and are alive at time t. As previously mentioned, the cost associated with each server corresponds to the total duration it remains open, and the overall cost of the algorithm is the sum of the costs of all servers. The following example gives a sample input and solution.

Example 3.1. In this example we have $d = 2$. The input sequence σ consists of four jobs, $\sigma_1 = (0, 6, s_1 = [0.5, 0.2]), \sigma_2 = (1, 4, s_2 = [0.2, 0.9]), \sigma_3 = (3, 9, s_3 = [0.2, 0.3]), \sigma_4 = (5, 8, s_4 = [0.6, 0.1])$. Figure 1 shows a possible assignment of these jobs to servers. The algorithm opens the first server for σ_1. However, upon the arrival of σ_2, a new server is opened since the first server lacks the capacity in the second dimension to accommodate it. When the third job σ_3 arrives, it is assigned to the first server. However, for σ_4, a new server must be opened as the first server does not have enough capacity in the first dimension, and the second server is already closed.

The cost of each server is determined by its opening and closing times. The first server's cost is 9 (opening at 0 and finishing at 9), while the second and third servers both have a cost of 3. The total cost of the algorithm is the sum of the costs of all servers, resulting in a total cost of 15.

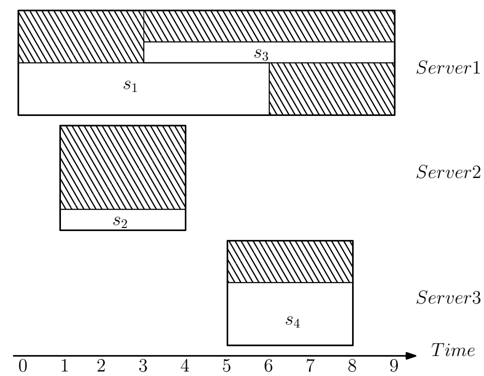

Figure 1: Online assignment of jobs into servers described in Example 3.1. Note that in this figure, we show the jobs according to the size of the first dimension.

We shall sometimes also use r to denote an arbitrary job in σ. In this case, we use the notation $r = (a(r), f(r), s(r))$ and we use $s(r)_j$ to denote the j^{th} coordinate of the size vector $s(r)$. The two key parameters of an instance σ, usually called the *span* and the *utilization* [19], are defined respectively below,

$$\text{span}(\sigma) = \left| \cup_{i \in [n]} [a_i, f_i) \right|, \quad \text{util}(\sigma) = \sum_{i=1}^{n} \text{util}(\sigma_i).$$

Without loss of generality, we assume that $\cup_{i \in [n]} [a_i, f_i) = [0, T)$, that is, span arises from a single uninterrupted interval, and the first job arrives at time 0.

For an algorithm ALG and $t \in [0, T)$ we use $\text{ALG}(\sigma, t)$ to denote the number of servers opened by ALG that are active at time t. We use $\text{ALG}(\sigma)$ to denote the total cost of ALG on input σ, i.e., the sum of durations of servers opened by ALG. Similar notation is used for OPT. Observe that

Proposition 3.2.

$$\text{OPT}(\sigma) = \int_0^T \text{OPT}(\sigma, t) dt, \text{ and}$$

$$\text{ALG}(\sigma) = \int_0^T \text{ALG}(\sigma, t) dt.$$

We note the following general lower bound on $\text{OPT}(\sigma)$.

Lemma 3.3. $\int_0^T \lceil \|s(\sigma, t)\|_\infty \rceil dt \leq \text{OPT}(\sigma)$

Proof. As the capacity of each server is 1 for every dimension $j \in [d]$; any algorithm needs at least $\lceil \|s(\sigma, t)\|_\infty \rceil$ servers to pack the total load at any time t. Therefore, $\text{OPT}(\sigma, t) \geq \lceil \|s(\sigma, t)\|_\infty \rceil$. Using Proposition 3.2, we can conclude:

$$\int_0^T \lceil \|s(\sigma, t)\|_\infty \rceil dt \leq \int_0^T \text{OPT}(\sigma, t) dt = \text{OPT}(\sigma). \qquad \square$$

Corollary 3.4 ([22]). $\mathrm{OPT}(\sigma) \geq \max\{\mathrm{span}(\sigma), \mathrm{util}(\sigma)/d\}$.

An online algorithm ALG is said to be *asymptotically* ρ-competitive if there exists a constant $c > 0$ such that for all input sequences σ:

$$\mathrm{ALG}(\sigma) \leq \rho \cdot \mathrm{OPT}(\sigma) + c. \tag{1}$$

The infimum over all such ρ is denoted by $\rho(\mathrm{ALG})$ and is called the *competitive ratio* of ALG. If $c = 0$ then the algorithm is called *strictly* ρ-competitive.

Lastly, we use notation $\mathbb{1}(C)$ for the indicator function that evaluates to 1 if the condition C is satisfied, and it evaluates to 0 otherwise.

4 MONOTONE *ANYFIT* ALGORITHMS

An algorithm for *RSiC* is said to be an *AnyFit* algorithm if it opens a new server only in case that a new incoming job cannot be accommodated on any of the currently active servers. Most *AnyFit* algorithms use an ordering of active servers, and assign the next job to the first server in the ordering with enough available space to accommodate the job. In this case, we say that the algorithm *employs an ordering*. For example, *FirstFit* orders servers based on their opening times, *BestFit* orders servers based on their remaining capacity, and *MTF* moves the server to which a job is assigned to the first position in the ordering. Observe that the ordering of servers could be fixed as in *FirstFit* and *LastFit*, or it could change when a job arrives, as in *BestFit*, *WorstFit*, and *MTF*, as well as when a job leaves, as in *BestFit* and *WorstFit*.

Consider an algorithm ALG that employs an ordering. We say that a server S is higher in the ordering than S' at time t if S appears closer to the beginning of the ordering than S'. Consider $t < t'$ and define $A(t, t')$ to be the set of servers that are alive at t and t'.

Definition 4.1. An *AnyFit* algorithm ALG is called *monotone* if

- it employs an ordering, and
- for every $t < t'$ and every server $S \in A(t, t')$: if S did not receive any new jobs during the interval (t, t') then every server in $A(t, t')$ that is higher than S in the ordering at time t is still higher than S in the ordering at time t'.

Note that for a monotone *AnyFit* algorithm a server S can move up in the ordering between t and t' only if either some server that was higher than S at time t was released before t', or S received a job during the interval (t, t'). It is clear that *AnyFit* algorithms employing a static ordering such as *FirstFit* and *LastFit* have the monotone property. Observe that in *MTF*, a server that receives a job moves to the first position in the ordering, and the relative position of other servers stays the same. Since the only way for a server to move ahead of other servers in the ordering is for it to receive a job, *MTF* obeys the monotone property. However, in *BestFit*, a server may move down in the ordering when a job departs, causing the server to have more available space. Thus *BestFit* does not obey the monotone property.

We propose a new monotone *AnyFit* algorithm, that surprisingly has not been studied earlier:

***Greedy*:** Order servers in decreasing order of their *finishing times*, that is, the maximum of the finishing times of jobs currently in the server. Assign the newly arrived job to the first server in the order that has sufficient capacity. If no such server exists, open a new server and assign the job to it.

Greedy is a natural and easy-to-implement algorithm that uses the greedy heuristic of assigning the incoming job to the server that will incur the *least additional cost* to the algorithm. Observe that a server S moves ahead of another server S' in the ordering if and only if S receives a job that causes its finishing time to be higher than that of S'. Therefore *Greedy* obeys the monotone property specified in Definition 4.1. We note that it is a clairvoyant algorithm.

We need the following lemma before we prove the main result of this section. We denote the sum of L_∞ norms of sizes of jobs with arrival time in the interval (t, t') for $t < t'$ by $s_\infty(\sigma, t, t')$, i.e., $s_\infty(\sigma, t, t') = \sum_{i: t < a_i < t'} \|s_i\|_\infty$.

Lemma 4.1. $\int_0^T s_\infty(\sigma, t - \alpha, t)dt = \alpha \sum_{i=1}^n \|s_i\|_\infty$.

Proof.

$$\begin{aligned}
\int_0^T s_\infty(\sigma, t-\alpha, t)dt &= \int_0^T \sum_{i=1}^n \mathbb{1}(t - \alpha < a_i < t)\, \|s_i\|_\infty \, dt \\
&= \sum_{i=1}^n \int_0^T \mathbb{1}(t - \alpha < a_i < t)\, \|s_i\|_\infty \, dt \\
&= \sum_{i=1}^n \|s_i\|_\infty \int_0^T \mathbb{1}(t - \alpha < a_i < t) dt \\
&= \sum_{i=1}^n \alpha \, \|s_i\|_\infty .
\end{aligned}$$

□

Now, we are ready to prove the main result of this section.

Theorem 4.2. *Let* ALG *be a monotone AnyFit algorithm. Then we have* $\rho(\mathrm{ALG}) \leq 3\mu d + 1$.

Proof. We claim that for an arbitrary t it holds that

$$\mathrm{ALG}(\sigma, t) \leq s_\infty(\sigma, t - 2\mu, t) + s_\infty(\sigma, t - \mu, t) + 1. \tag{2}$$

Observe that each server in $A(t - \mu, t)$ must have received a job during the time interval $(t - \mu, t)$, otherwise a server alive at time $t - \mu$ would have been released by time t, since the duration of each job is at most μ. Suppose there are q servers in $A(t - \mu, t)$ named $A_1, A_2, \ldots, A_q$, ordered according to the ordering of ALG at time $t - \mu$. Let t_i be the earliest time in $(t - \mu, t)$ when a job with the size vector s_i arrived in server A_i. Let $B(t - \mu, t)$ denote the set of new servers that were opened during time $(t - \mu, t)$ that are still alive at time t. Suppose there are p such servers called $B_1, B_2, \ldots, B_p$ ordered by their opening times $t'_1, t'_2, \ldots, t'_p$. Let s'_i be the size vector of the first job placed into B_i. See Figure 2 for an illustration. Note that we have $\mathrm{ALG}(\sigma, t) = p + q$.

Consider some $i \in \{2, \ldots, q\}$. Observe that A_{i-1} preceded A_i in the ordering of ALG at time $t - \mu$, no job arrived in A_i during time interval $(t - \mu, t_i)$, and ALG is monotone. Thus, A_{i-1} precedes A_i in the ordering of ALG immediately prior to arrival of job s_i. Thus, ALG must have tried placing s_i into server A_{i-1} at time t_i, but could not fit it in (since s_i was ultimately placed into A_i). This happened because in some coordinate the total size of jobs in server A_{i-1} plus the size of the job s_i in that coordinate exceeded the capacity. Thus,

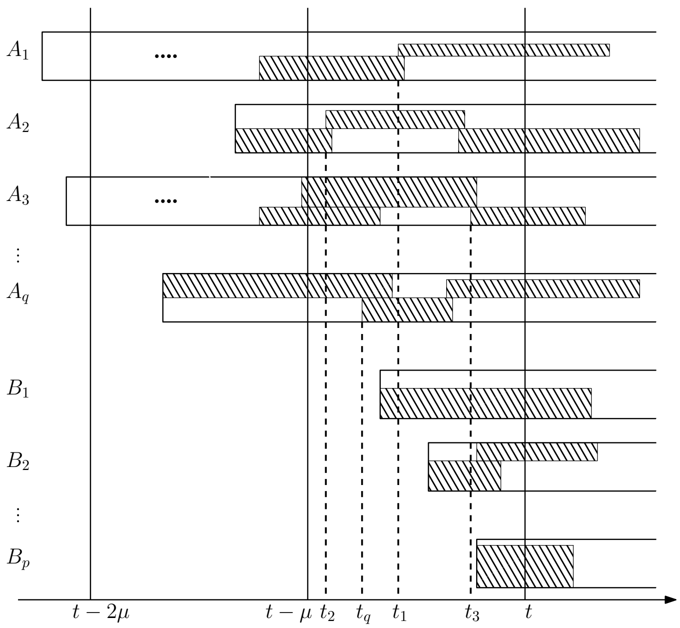

Figure 2: Configuration of servers in interval $[t-2\mu, t]$. Note that the A_i servers are ordered according to the ordering of ALG at time $t-\mu$ while the B_i servers are ordered by opening time.

we can conclude that

$$\|s_i + A_{i-1}(t_i)\|_\infty > 1. \tag{3}$$

For the B_i servers, since ALG is an *AnyFit* algorithm (it opens a new server only if it has to), we have:

$$\left\|s'_1 + A_q(t'_1)\right\|_\infty > 1 \text{ and } \left\|s'_i + B_{i-1}(t'_i)\right\|_\infty > 1 \text{ for } 2 \le i \le p \tag{4}$$

Also, note that

$$\sum_{i=2}^{q} \|A_{i-1}(t_i)\|_\infty + \left\|A_q(t'_1)\right\|_\infty + \sum_{i=2}^{p} \left\|B_{i-1}(t'_i)\right\|_\infty \le s_\infty(\sigma, t-2\mu, t), \tag{5}$$

since all jobs that are alive in server A_{i-1} at time t_i, as well as in A_q at time t'_1, must have arrived between $t-2\mu$ and t, and the A_i and the B_j servers partition the set of relevant jobs. In addition, we have

$$\sum_{i=2}^{q} \|s_i\|_\infty + \left\|s'_1\right\|_\infty + \sum_{i=2}^{p} \left\|s'_i\right\|_\infty \le s_\infty(\sigma, t-\mu, t), \tag{6}$$

since the s_i and the s'_i jobs have arrival times between $t-\mu$ and t, and the jobs are distinct.

Combining the observations in (3) and (4), we obtain

$$\begin{aligned} q+p-1 &< \sum_{i=2}^{q} \|s_i + A_{i-1}(t_i)\|_\infty + \left\|s'_1 + A_q(t'_1)\right\|_\infty \\ &\quad + \sum_{i=2}^{p} \left\|s'_i + B_{i-1}(t'_i)\right\|_\infty \\ &\le s_\infty(\sigma, t-2\mu, t) + s_\infty(\sigma, t-\mu, t), \end{aligned}$$

where the second inequality follows from Proposition 3.1 and application of (5) and (6). This establishes Inequality (2).

To finish the proof of the theorem, we integrate this inequality over possible values of t, i.e.:

$$\begin{aligned} \mathrm{ALG}(\sigma) &= \int_0^T \mathrm{ALG}(\sigma, t)dt \\ &\le \int_0^T (s_\infty(\sigma, t-2\mu, t) + s_\infty(\sigma, t-\mu, t) + 1)dt \\ &= 2\mu \sum_{i=1}^{n} \|s_i\|_\infty + \mu \sum_{i=1}^{n} \|s_i\|_\infty + \mathrm{span}(\sigma) \\ &\le 3\mu \,\mathrm{util}(\sigma) + \mathrm{span}(\sigma) \\ &\le 3\mu d \,\mathrm{OPT}(\sigma) + \mathrm{OPT}(\sigma), \end{aligned}$$

where the second equality follows from two applications of Lemma 4.1, and the last inequality is an application of Proposition 3.4. □

We note that there is not much room for improvement of the bound in the above theorem, because Murhekar et al. [22] proved a lower bound of $(\mu+1)d$ for any *AnyFit* algorithm.

Corollary 4.3.

$$(\mu+1)d \le \rho(\mathit{Greedy}), \rho(\mathit{LastFit}) \le 3d\mu + 1$$

As mentioned earlier, the proof of a $6\mu + 8$ upper bound on the competitive ratio of *MTF* given in [15] also works for *Greedy* in the 1-dimensional case. However, Corollary 4.3 improves this bound. We conjecture that the correct bound on the competitive ratio of *Greedy* is $\mu d + O(d)$.

5 A DIRECT-SUM PROPERTY OF *RSIC*

Given an arbitrary algorithm ALG for 1-dimensional *RSiC*, we define an algorithm, call it $\mathrm{ALG}^{\oplus d}$, that works for d-dimensional *RSiC*, as follows. Let σ be an input instance for the d-dimensional problem. We partition σ as follows:

$$\sigma = \sigma^{(1)} \cup \cdots \cup \sigma^{(d)},$$

where $\sigma^{(j)}$ is the subset of jobs r for which $\|s(r)\|_\infty$ is achieved at the j-th dimension. When $\|s(r)\|_\infty$ is achieved in more than one dimension, we break the tie arbitrarily. It is easy to see that this partitioning can be done online. Each $\sigma^{(j)}$ will be assigned to a different set of servers.

The algorithm $\mathrm{ALG}^{\oplus d}$ is defined as follows: on the arrival of a job r, decide online a unique dimension j in which its size is maximum, and assign $r \in \sigma^{(j)}$. Next apply the algorithm ALG (for 1-dimensional *RSiC*) to process $\sigma^{(j)}$, pretending that the instance is 1-dimensional by only looking at the size at the j-th coordinate, and ignoring the sizes of other dimensions, and assigning to servers that only contain jobs in $\sigma^{(j)}$.

Theorem 5.1. *Let* ALG *be an arbitrary deterministic algorithm for* 1*-dimensional RSiC. Then,* $\mathrm{ALG}^{\oplus d}$ *works correctly for any* d*-dimensional RSiC, and* $\rho(\mathrm{ALG}^{\oplus d}) = d \cdot \rho(\mathrm{ALG})$*. Moreover, the guarantee on the competitive ratio holds for both strict and asymptotic competitive ratios.*

Proof. Firstly, we show that the algorithm $\mathrm{ALG}^{\oplus d}$ does not violate the size constraint, i.e., the total size of all jobs in every server does not exceed 1^d. To see this, consider an arbitrary job $r \in \sigma$ and suppose it is put into a bin B by $\mathrm{ALG}^{\oplus d}$. By the definition of $\mathrm{ALG}^{\oplus d}$, we know before r is put into B, either server B is empty (i.e., has not been created yet) in which case after r is assigned to server B the size constraint is trivially respected, or server B is nonempty. In the latter case, it only contains jobs in $\sigma^{(j)}$. In this case, we have

$$\|s(r) + B(t)\|_\infty = s(r)_j + B(t)_j \leq 1,$$

where the equality follows from the fact that every job in server B is in $\sigma^{(j)}$ and also $r \in \sigma^{(j)}$. The inequality follows by the fact that we apply algorithm ALG on $r \in \sigma^{(j)}$.

Next, we show $\rho(\mathrm{ALG}^{\oplus d}) \leq d \cdot \rho(\mathrm{ALG})$. By an abuse of notation, let $\mathrm{ALG}(\sigma^{(j)})$ denote the cost of $\mathrm{ALG}^{\oplus d}(\sigma)$ on the subset of inputs $\sigma^{(j)}$. Since $\sigma^{(j)} \subseteq \sigma$, one has $\mathrm{OPT}(\sigma^{(j)}) \leq \mathrm{OPT}(\sigma)$ for every j. Let $\mathrm{OPT}'(\sigma^{(j)})$ denote the cost of the optimal solution that processes $\sigma^{(j)}$ by only focusing on the size of the j-th dimension. Then, for every $\rho > \rho(\mathrm{ALG})$ there exists a $c > 0$ such that $\mathrm{ALG}(\sigma^{(j)}) \leq \rho \cdot \mathrm{OPT}'(\sigma^{(j)}) + c$ for every j. Observe that we have $\mathrm{OPT}'(\sigma^{(j)}) = \mathrm{OPT}(\sigma^{(j)})$. This is because every job in $\sigma^{(j)}$ satisfies that the size at the j-th dimension is the largest. With these, and by the definition of $\mathrm{ALG}^{\oplus d}$, we have

$$\begin{aligned}
\mathrm{ALG}^{\oplus d}(\sigma) &= \sum_{j \in [d]} \mathrm{ALG}(\sigma^{(j)}) \\
&\leq \sum_{j \in [d]} \left(\rho \cdot \mathrm{OPT}'(\sigma^{(j)}) + c\right) \\
&= \sum_{j \in [d]} \left(\rho \cdot \mathrm{OPT}(\sigma^{(j)}) + c\right) \\
&\leq \sum_{j \in [d]} (\rho \cdot \mathrm{OPT}(\sigma) + c) = d \cdot \rho \cdot \mathrm{OPT}(\sigma) + cd.
\end{aligned}$$

Since cd is a constant independent of input, and this inequality holds for all $\rho > \rho(\mathrm{ALG})$, it follows that $\rho(\mathrm{ALG}^{\oplus d}) \leq d\rho(\mathrm{ALG})$. Moreover, if $c = 0$ then $cd = 0$, so the competitive ratio guarantee preserves strictness.

Lastly, we show $\rho(\mathrm{ALG}^{\oplus d}) \geq d \cdot \rho(\mathrm{ALG})$. Let H be an arbitrary 1-dimensional instance, from which we construct a d-dimensional instance σ as follows. For every job $h = (a(h), f(h), s(h)) \in H$, create d jobs in σ that have the same arrival and finishing time as h, and the size vectors are the d column vectors of the matrix $s(h) \cdot I_d$ where I_d is the $d \times d$ identity matrix. Clearly, every $\sigma^{(j)}$ is simply a copy of H in dimension j, while having 0's in all other dimensions. Hence, $\mathrm{ALG}^{\oplus d}(\sigma) = d \cdot \mathrm{ALG}(H)$. Furthermore, Observe that $\mathrm{OPT}(H) = \mathrm{OPT}(\sigma)$, where here by an abuse of notation we use $\mathrm{OPT}(H)$ to denote the cost of the optimal algorithm for the 1-dimensional instance H, and $\mathrm{OPT}(\sigma)$ to denote the cost of the optimal algorithm for the d-dimensional instance σ. Hence,

$$\rho(\mathrm{ALG}^{\oplus d}) \geq \frac{\mathrm{ALG}^{\oplus d}(\sigma)}{\mathrm{OPT}(\sigma)} = \frac{d \cdot \mathrm{ALG}(H)}{\mathrm{OPT}(\sigma)} = d \cdot \frac{\mathrm{ALG}(H)}{\mathrm{OPT}(H)}.$$

Because H is arbitrary, the desired lower bound follows. □

In [2], the Hybrid Algorithm *HA* is defined for 1-dimensional clairvoyant *RSiC*, and is shown to have a competitive ratio $\Theta(\sqrt{\log \mu})$.

Corollary 5.2. *The algorithm $HA^{\oplus d}$ for d-dimensional clairvoyant RSiC has a competitive ratio $\Theta(d\sqrt{\log \mu})$.*

6 LOWER BOUNDS VIA ONLINE GRAPH COLORING

In this section, we show lower bounds on the competitive ratio of any algorithm for d-dimensional *RSiC*. We consider deterministic algorithms in Section 6.1 for both the clairvoyant and non-clairvoyant versions of the problem. In Section 6.2, we give lower bounds for randomized algorithms for both versions of the problem.

6.1 Deterministic algorithms

For 1-dimensional non-clairvoyant *RSiC*, it is shown [15, Theorem 1] that μ is a lower bound for *any* deterministic algorithm. By adapting the online graph coloring lower bound construction from Halldórsson-Szegedy [11], we show a $\widetilde{\Omega}(d\mu)$ lower bound for d-dimensional non-clairvoyant *RSiC*. For readers familiar with the online graph construction in [11], a size vector in the proof of Theorem 6.1 below corresponds to a vertex in the online graph, and the size vectors of the jobs are chosen in the way such that jobs that fit into one server correspond to an independent set of vertices of the online graph. The idea of applying online graph coloring to online vector bin packing has been discussed in [1, 17].

Theorem 6.1. *There exists a constant d_0 (which can take $d_0 = 280$) such that for every dimension $d \geq d_0$ and $\mu \leq \log d - 2$, every deterministic algorithm for non-clairvoyant RSiC has a competitive ratio $\geq \frac{d}{\log^3 d} \cdot \mu$. For clairvoyant RSiC, the lower bound $\Omega(\max\{\sqrt{\log \mu}, \frac{d}{\log^2 d}\})$ holds.*

Proof. Let ALG be an arbitrary deterministic algorithm for d-dimensional non-clairvoyant *RSiC*. Choose k to be the largest integer for which $d \geq d' = \binom{2k}{k} \cdot k \geq 2^{2k}$. Let ADV denote the adversary, which will adaptively construct an instance $\sigma = (\sigma_1, \ldots, \sigma_{d'})$ consisting of d' jobs, such that ALG uses at least d'/k servers on σ, while OPT uses at most $2k$ servers. In fact, ADV will construct a solution using no more than $2k$ servers. Note that all jobs arrive at the same time 0. At the end of d' steps, once all jobs have been processed by ALG, the adversary specifies the finish times of all jobs.

To avoid confusion, we call the servers used by ALG *bins*, while the servers used by ADV will be called *servers*. We name the $2k$ servers that will be used by ADV as $1, 2, \ldots, 2k$. For an arbitrary job $r \in \sigma$, let $\mathrm{ADV}(r) \in [2k]$ denote the server to which ADV assigns r. Let B be an arbitrary bin used by ALG, and let $\hat{B}_{\leq i}$ denote the set of jobs in B after the first i jobs have been processed, and let $B_{\leq i}$ denote the sum of the sizes of these jobs. Define

$$\mathrm{ADV}(B, i) := \{\mathrm{ADV}(r) : r \in \hat{B}_{\leq i}\},$$

i.e., it is the set of servers that the adversary used to assign jobs present in bin B after the first i jobs have been processed.

In step i, the adversary specifies the job σ_i, whose size is defined as follows. Let X denote the set of servers used by ALG that have exactly k jobs placed in them after the first $i - 1$ jobs have been processed. We define

$$\mathcal{F}_i = \{\mathrm{ADV}(B, i-1) : B \in X\}. \tag{7}$$

Note that $\mathcal{F}_i$ is a set of subsets of $[2k]$; each element of $\mathcal{F}_i$ is the set of servers in which the adversary placed the jobs of k-sized bins of ALG. The adversary chooses an arbitrary subset A_i of size k

$$A_i \in \binom{[2k]}{k} - \mathcal{F}_i, \tag{8}$$

Since each distinct subset of $[2k]$ in $\mathcal{F}_i$ corresponds to k jobs in the input sequence σ, and recalling that the length of σ is $d' = \binom{2k}{k}k$, the set A_i always exists.

The adversary now defines $s_i \in [0,1]^{d'}$ as follows:

$$s_i^j = \begin{cases} 0, & \text{if } j < i \text{ and } \mathrm{ADV}(\sigma_j) \in A_i, \\ 1/d, & \text{if } j < i \text{ and } \mathrm{ADV}(\sigma_j) \notin A_i, \\ 1, & \text{if } j = i, \\ 0, & \text{if } j > i. \end{cases} \tag{9}$$

The adversary assigns σ_i to an arbitrary server $Q \in A_i$. We show below that this is a valid assignment, that is, it does not violate capacity constraints.

Claim 6.2. *Let $Q \in A_i$ denote an arbitrary server in A_i. Then*

$$\|s_i + Q(i-1)\|_\infty \le 1 \tag{10}$$

Proof. Indeed, by (9), $\sigma_j \in \hat{Q}(i-1)$ implies $s_i^j = 0$. Also, because $j < i$, again by (9) one has $s_j^i = 0$. The claim follows. □

The adversary now presents job σ_i to ALG and suppose ALG assigns σ_i to bin B. Next, we show that after the assignment, bin B (and indeed any bin of ALG) contains at most k jobs.

Claim 6.3. *$|\hat{A}(i)| \le k$ for every bin A in use by* ALG *after i jobs have been processed.*

Proof. Assume this is true inductively after $i-1$ jobs have been processed. Since σ_i was added to bin B, we only need to argue that bin B had at most $k-1$ jobs after $i-1$ jobs were processed. Suppose instead that $|\hat{B}(i-1)| = k$. Then $\mathrm{ADV}(B, i-1) \in \mathcal{F}_i$, and so $A_i \neq \mathrm{ADV}(B, i-1)$.

Since both A_i and $\mathrm{ADV}(B, i-1)$ are of size k, there must exist at least one job $\sigma_j \in \hat{B}(i-1)$ for which $\mathrm{ADV}(\sigma_j) \notin A_i$. This implies $s_i^j + s_j^j = 1/d + 1 > 1$. Hence, s_i does not fit into bin B, which contradicts the choice of B by ALG. □

Claim 6.2 shows that the optimal assignment uses at most $2k$ servers while Claim 6.3 shows that ALG needs at least d'/k bins to assign the jobs in σ.

We are now ready to determine the choice of f_i, the finishing time of job σ_i. Observe that there exists at least one server of the adversary, call it Q, such that jobs in $\hat{Q}_{\le d'}$ appear in at least $(d'/k)/(2k) = d'/2k^2$ distinct bins of ALG. The adversary sets the duration of jobs $\hat{Q}_{\le d'}$ to be μ, and the duration of all other jobs to be 1.

Hence, the cost of OPT is at most $2k - 1 + \mu$, while the cost of ALG is at least $d'\mu/2k^2$. Using the fact that for $k \ge 4$, we have $d' = \binom{2k}{k} \cdot k \ge 2^{2k}$, the competitive ratio of ALG is at least

$$\frac{d'\mu/2k^2}{2k - 1 + \mu} \ge \frac{d'}{8k^3} \cdot \mu \ge \frac{d'}{\log^3 d'} \cdot \mu,$$

provided $\mu \le 2k$, and $k \ge 4$. Next, observe that

$$d < \binom{2k+1}{k+1} \cdot (k+1) \le \binom{2k+1}{k+1} \cdot k + 2^{2k+1} \le 2\binom{2k}{k} \cdot k + 2^{2k+1}$$
$$\le 2d' + 2d' = 4d'$$

Hence, $\frac{d'}{\log^3 d'} \cdot \mu \ge \frac{d}{4\log^3 d} \cdot \mu$ which gives the desired bound on the competitive ratio for dimension $d > 280$ (which ensures that $k \ge 4$), and $\mu \le \log(d/4) = \log d - 2$.

Finally, if the model is clairvoyant, the adversary sets all jobs to be of duration 1, and obtains a lower bound of competitive ratio $(d/k)/(2k) = d/2k^2 = \Omega(d/\log^2 d)$ that is independent of μ. The other lower bound $\Omega(\sqrt{\log \mu})$ comes from the lower bound for 1-dimensional clairvoyant *RSiC* established in [2]. □

6.2 Randomized algorithms

Theorem 6.4. *In 1-dimensional non-clairvoyant RSiC, any randomized algorithm has a competitive ratio at least $\frac{1-e^{-1}}{2} \cdot \mu$.*

Proof. We use Yao's principle. Consider the following distributional input: k^2 jobs each of size $1/k$, uniformly at random pick k among k^2 jobs to be of duration μ, and let the rest $k^2 - k$ jobs to be of duration 1. Let ALG be an arbitrary deterministic algorithm. We show that in expectation ALG has cost $\Omega(k\mu)$. Since OPT has cost $\le k - 1 + \mu$, this gives the competitive ratio $\Omega(\mu)$ as desired.

Let $A_1, \ldots, A_m$ be the m servers that ALG uses for the above instance. Let $|A_i|$ denote the number of jobs in A_i, then $1 \le |A_i| \le k$. We partition these m servers into p groups $B_1, \ldots, B_{p-1}, B_p$ such that the number of jobs in each group B_i contains $\ge k$ jobs and $< 2k$ jobs, except perhaps the last group B_p which may contain less than k jobs. Note that such a partition exists by simply partitioning greedily. Hence, $p \ge k^2/2k = k/2$. Let $|B_i|$ denote the number of jobs in group B_i. Then, for every $1 \le i \le p-1$,

$$\Pr[B_i \text{ contains at least one job of duration } \mu] = 1 - \frac{\binom{k^2 - |B_i|}{k}}{\binom{k^2}{k}}$$

$$\ge 1 - \frac{\binom{k^2-k}{k}}{\binom{k^2}{k}} \ge 1 - (1 - \frac{1}{k})^k \ge 1 - e^{-1}.$$

Let $X_i \in \{0,1\}$ be a random variable denoting whether group B_i contains some job of duration μ or not. Then, by the linearity of expectation, the expected cost of ALG is at least

$$\mu \cdot \mathbb{E}\left[\sum_{i=1}^{p-1} X_i\right] = \mu \cdot \sum_{i=1}^{p-1} \mathbb{E}[X_i] \ge \mu \cdot (p-1)(1 - e^{-1})$$
$$\ge \mu \cdot (k/2 - 1)(1 - e^{-1}).$$

Hence, the competitive ratio is at least $\frac{1-e^{-1}}{2} \cdot \frac{k-2}{k-1+\mu} \cdot \mu$. For every μ, since we can pick k to be arbitrarily large, we get the ratio is at least $\frac{1-e^{-1}}{2} \cdot \mu$ as claimed. □

Theorem 6.5. *There exists a constant d_0 (can take $d_0 = 280$) such that for every dimension $d \ge d_0$ and $\mu \le \log d - 2$, every randomized algorithm for d-dimensional non-clairvoyant RSiC has a competitive ratio $\ge \Omega(\frac{d}{\log^4 d} \cdot \mu)$. For d-dimensional clairvoyant RSiC, the lower bound $\Omega(\frac{d}{\log^2 d})$ holds.*

Proof. We apply Yao's principle. In [11], it is shown that the online graph lower bound construction can be modified to work against randomized algorithms, with only a loss of a constant. We refer the interested reader to the original paper [11] for details. Here we only give the necessary modifications, based on the proof given in Theorem 6.1. The oblivious adversary constructs a distributional input, as follows.

- The adversary uniformly at random picks a server $Q \in [2k]$;
- In step (8), the adversary instead chooses a random subset $A_i \subseteq [2k]$ of size k;
- the vector s_i is defined in the same way;
- the adversary assigns s_i to a random server $X \in A_i$;
- if $X = Q$, the adversary sets the duration of s_i to be μ, otherwise to be 1.

This finishes the construction of the distributional input. Note that the adversary is oblivious.

By the same argument in [11], under this distributional input, one can show that any deterministic algorithm uses in expectation $\Omega(d/k)$ bins. Note that adversary still uses at most $2k$ servers, with cost at most $2k - 1 + \mu$. Hence, as in the deterministic proof, there exists at least one adversary's server Q^*, such that jobs in this server appear in at least $\Omega(d/k)/2k = \Omega(d/k^2)$ algorithm's bins. Since the adversary randomly picks a server Q and sets jobs in it to be of duration μ, we conclude that in expectation the algorithm has cost at least $\geq \frac{1}{2k} \cdot \Omega(d/k^2) \cdot \mu = \Omega(d\mu/k^3)$. Hence, the expected competitive ratio is at least $\frac{\Omega(d\mu/k^3)}{2k-1+\mu} \geq \Omega(d\mu/k^4)$, where $k = \Theta(\log d)$, as claimed. For the clairvoyant case, the lower bound follows by setting all jobs to be of duration 1. □

7 EXPERIMENTS

In this section, we provide a thorough evaluation of the average-case performance of almost all existing non-clairvoyant as well as clairvoyant algorithms for the *RSiC* problem.

7.1 Experimental setup

We evaluate the performance of different algorithms using randomly generated input sequences for d-dimensional *RSiC*, for $d \in \{1, 2, 5\}$, closely adhering to the experimental setup detailed in [15] for the 1-dimensional case. In the experiments, we assume that each server has size E^d where $E = 1000$, and each job is assumed to have a size in $\{1, 2, \cdots, E\}^d$. For a given integral span value T; for $T \in \{1000, 5000, 10000\}$, we assume that each job arrives at an integral time step within the interval $[0, T - \mu]$ and has an integral duration in $[1, \mu]$, for $\mu \in \{1, 2, 5, 10, 100\}$. Each experimental instance comprises a sequence of $n = 10000$ jobs, with the size and duration of each job selected randomly from their respective ranges, assuming a uniform distribution. The reported upper bound on competitve ratio of each studied algorithm is computed as the ratio of the average cost of the algorithm over 100 input sequences, and the average of the lower bound on OPT given by Lemma 3.3 for these instances.

All our experiments were executed on a personal laptop with a Dual-core 2.3 GHz Intel Core i5 CPU. The laptop had 8 GB of RAM. The laptop was running Mac OS version 12.6.4. The code was written in C++ using VS code version 1.38.1.

7.2 Implemented algorithms

We implemented both clairvoyant and non-clairvoyant algorithms. The non-clairvoyant algorithms we implemented are:

- ***NextFit***: which keeps only one open server at each time.
- ***ModifiedNextFit***: which assigns jobs with sizes greater than a specific threshold separately from the other jobs using the *NextFit* algorithm.
- ***FirstFit***: monotone *AnyFit* that orders servers in increasing order of opening time.
- ***LastFit***: monotone *AnyFit* that orders servers in decreasing order of opening time.
- ***ModifiedFirstFit***: which assigns jobs with sizes greater than a specific threshold separately from the other jobs using the *FirstFit* algorithm.
- ***BestFit***: *AnyFit* that orders servers in increasing order of remaining capacity.
- ***WorstFit***: *AnyFit* that orders servers in decreasing order of remaining capacity.
- ***RandomFit***: *AnyFit* algorithm that orders servers randomly.
- ***MTF***: monotone *AnyFit* that orders servers in decreasing order of the last time a job was assigned to it.

In our experiments, similar to [15], we adopt the parameters for *ModifiedNextFit* and *ModifiedFirstFit* as $E^d/(\mu + 1)$ and $E^d/(\mu + 7)$, respectively. This choice of values is designed to optimize the competitive ratio of these algorithms, as indicated in [15, 19].

The clairvoyant algorithms we implemented are:

- ***Departure Strategy*** [24]: the span is split into intervals of length τ each, where $\tau > 0$ is a constant. Classifies jobs into categories according to their departure times. Each category contains all jobs that depart in a time interval of length τ.
- ***Duration Strategy*** [24]: classifies the jobs into categories such that the max/min job duration ratio for each category is a given constant α. Given a base job duration b, each category includes all the jobs with durations between $b\alpha^{i-1}$ and $b\alpha^i$ for an integer i.
- ***Hybrid Algorithm (HA)*** [2]: classifies jobs according to their length and their arrivals. Suppose the maximum duration of jobs in the input sequence is μ. Then all the jobs whose lengths are in range $[2^{i-1}, 2^i]$ for integer $1 \leq i \leq \lceil \log \mu \rceil + 1$ and whose arrival times are in time interval $[(c-1)2^i, c2^i)$ for an integer c are put into the same category.
- ***Greedy***: monotone *AnyFit* as defined in Section 4.
- **New Hybrid**: $HA^{\oplus d}$ as defined in Section 5.

7.3 Experimental results

Our experimental results for $d \in \{1, 2, 5\}$ are shown in Tables 1, 2 and 3, respectively. We validated our results against those in [18] for $d = 1$ and for the algorithms implemented there. Our results are slightly different as we use a better lower bound to compute the competitive ratio; when using the same lower bound as [18], our results match exactly.

	T=1000					T=5000					T=10000				
	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$
	Non-clairvoyant														
NextFit	1.27	1.37	1.45	1.49	1.52	1.12	1.20	1.32	1.40	1.51	1.06	1.10	1.20	1.31	1.50
MNF	1.31	1.39	1.43	1.48	1.52	1.19	1.29	1.41	1.47	1.52	1.11	1.19	1.31	1.39	1.51
WorstFit	1.41	1.39	1.36	1.33	1.29	1.16	1.20	1.26	1.28	1.29	1.06	1.09	1.16	1.22	1.29
FirstFit	1.42	1.36	1.30	1.27	1.22	1.17	1.20	1.24	1.25	1.23	1.07	1.10	1.16	1.21	1.24
MFF	1.51	1.44	1.35	1.30	1.23	1.25	1.29	1.33	1.32	1.25	1.13	1.17	1.24	1.28	1.25
BestFit	1.51	1.41	1.31	1.24	**1.11**	1.17	1.21	1.25	1.26	**1.16**	1.07	1.10	1.17	1.22	1.19
LastFit	1.35	1.34	1.29	1.25	1.17	1.14	1.18	1.23	1.24	1.19	1.05	1.08	1.15	1.20	1.21
Random Fit	1.49	1.41	1.34	1.28	1.18	1.17	1.21	1.26	1.27	1.21	1.07	1.10	1.17	1.22	1.23
MTF	1.32	1.32	1.28	1.24	1.16	1.13	1.17	1.22	1.24	1.19	1.05	1.08	1.15	1.20	1.20
	Clairvoyant														
Departure Strategy	1.42	1.36	1.30	1.27	1.17	1.17	1.20	1.24	1.25	1.21	1.07	1.10	1.16	1.21	1.23
Duration Strategy	1.42	1.40	1.35	1.31	1.23	1.17	1.20	1.24	1.25	1.24	1.07	1.16	1.26	1.33	1.29
Hybrid Algorithm	**1.12**	**1.25**	1.32	1.33	1.25	**1.03**	1.22	1.36	1.39	1.31	**1.01**	1.15	1.30	1.39	1.34
New Hybrid	**1.12**	**1.25**	1.32	1.33	1.25	**1.03**	1.22	1.36	1.40	1.31	**1.01**	1.15	1.30	1.39	1.34
Greedy	1.28	1.27	**1.22**	**1.19**	1.13	1.12	**1.15**	**1.19**	**1.20**	**1.16**	1.05	**1.07**	**1.13**	**1.17**	**1.17**

Table 1: Experimental results for the *RSiC* problem when $d = 1$.

	T=1000					T=5000					T=10000				
	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$
	Non-clairvoyant														
NextFit	1.40	1.49	1.59	1.65	1.73	1.12	1.20	1.36	1.48	1.69	1.05	1.09	1.21	1.35	1.65
MNF	1.44	1.52	1.61	1.65	1.73	1.17	1.25	1.39	1.49	1.69	1.09	1.13	1.23	1.36	1.65
WorstFit	1.46	1.45	1.44	1.42	1.38	1.14	1.19	1.29	1.35	1.39	1.05	1.08	1.17	1.26	1.38
FirstFit	1.49	1.45	1.42	1.40	1.35	1.15	1.20	1.29	1.34	1.37	1.06	1.09	1.17	1.26	1.37
MFF	1.50	1.47	1.43	1.41	1.35	1.16	1.21	1.30	1.35	1.37	1.06	1.09	1.18	1.26	1.37
BestFit	1.48	1.44	1.40	1.37	1.26	1.14	1.19	1.28	1.33	1.31	1.05	1.08	1.17	1.25	1.33
LastFit	1.39	1.40	1.39	1.36	1.29	1.12	1.17	1.27	1.32	1.32	1.05	1.08	1.16	1.24	1.33
Random Fit	1.48	1.45	1.42	1.39	1.30	1.14	1.19	1.28	1.34	1.34	1.05	1.08	1.17	1.26	1.35
MTF	1.38	1.39	1.38	1.36	1.28	1.12	1.17	1.27	1.32	1.32	1.05	1.07	1.16	1.24	1.33
	Clairvoyant														
Departure Strategy	1.48	1.45	1.42	1.40	1.30	1.15	1.20	1.29	1.34	1.35	1.06	1.09	1.17	1.26	1.37
Duration Strategy	1.48	1.49	1.48	1.47	1.38	1.15	1.20	1.29	1.34	1.37	1.06	1.12	1.23	1.34	1.44
Hybrid Algorithm	**1.23**	1.37	1.45	1.47	1.38	**1.07**	1.21	1.36	1.45	1.45	**1.02**	1.11	1.25	1.37	1.48
New Hybrid	1.42	1.54	1.62	1.65	1.64	1.17	1.29	1.46	1.57	1.65	1.09	1.16	1.31	1.46	1.65
Greedy	1.36	**1.36**	**1.34**	**1.32**	**1.24**	1.12	**1.16**	**1.25**	**1.30**	**1.29**	1.04	**1.07**	**1.15**	**1.23**	**1.30**

Table 2: Experimental results for the *RSiC* problem when $d = 2$.

	T=1000					T=5000					T=10000				
	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$	$\mu = 100$
	Non-clairvoyant														
NextFit	1.49	1.58	1.70	1.77	1.90	1.11	1.19	1.37	1.52	1.82	**1.04**	1.08	1.19	1.35	1.76
MNF	1.50	1.58	1.70	1.77	1.90	1.12	1.19	1.37	1.52	1.82	**1.04**	1.08	1.19	1.35	1.76
WorstFit	1.47	1.52	1.59	1.61	1.61	1.11	**1.18**	**1.33**	1.45	1.61	**1.04**	**1.07**	1.18	**1.31**	1.60
FirstFit	1.48	1.53	1.59	1.62	1.62	1.11	**1.18**	1.34	1.46	1.62	**1.04**	**1.07**	1.18	**1.31**	1.60
MFF	1.48	1.53	1.59	1.62	1.62	1.11	**1.18**	1.34	1.46	1.62	**1.04**	**1.07**	1.18	**1.31**	1.60
BestFit	1.47	1.52	1.58	1.60	1.57	1.11	**1.18**	**1.33**	1.45	1.60	**1.04**	**1.07**	1.18	**1.31**	1.59
LastFit	1.45	1.51	1.57	1.60	1.57	1.11	**1.18**	**1.33**	1.45	1.60	**1.04**	**1.07**	1.18	**1.31**	**1.58**
Random Fit	1.47	1.52	1.58	1.61	1.59	1.11	**1.18**	**1.33**	1.45	1.61	**1.04**	**1.07**	1.18	**1.31**	1.59
MTF	1.45	1.51	1.57	1.60	1.57	1.11	**1.18**	**1.33**	1.45	1.60	**1.04**	**1.07**	1.18	**1.31**	1.59
	Clairvoyant														
Departure Strategy	1.48	1.53	1.59	1.62	1.61	1.11	**1.18**	1.34	1.46	1.63	**1.04**	**1.07**	1.18	**1.31**	1.61
Duration Strategy	1.48	1.55	1.63	1.67	1.66	1.11	**1.18**	1.34	1.46	1.63	**1.04**	1.08	1.19	1.34	1.65
Hybrid Algorithm	**1.42**	1.53	1.63	1.68	1.68	**1.10**	1.19	1.37	1.51	1.70	**1.04**	1.08	1.20	1.35	1.68
New Hybrid	1.52	1.61	1.72	1.79	1.91	1.13	1.21	1.39	1.55	1.85	1.05	1.09	1.21	1.37	1.79
Greedy	1.45	**1.50**	**1.56**	**1.59**	**1.55**	1.11	**1.18**	**1.33**	**1.44**	**1.59**	**1.04**	**1.07**	**1.17**	**1.31**	**1.58**

Table 3: Experimental results for the *RSiC* problem when $d = 5$.

First we note that for all algorithms, the experimentally derived competitive ratio on random inputs is much better than the worst-case bounds derived theoretically. This is not surprising as the worst-case inputs are carefully constructed to beat the given algorithm and are unlikely to occur in practice. Comparing the three tables, we see that the competitive ratio for every algorithm and every value of T and μ increases with increasing d. In general, the competitive ratio also increases with μ, keeping other parameters constant.

We note that many algorithms have versions that separate servers into separate categories, and assign jobs to servers in a particular category based on their sizes. In general, such modifications of algorithms do not perform better than the original versions on random inputs, even though they have better worst-case competitive ratios. For example, *MFF* performs worse than *FirstFit*, *MNF* performs worse than *NextFit*, and New Hybrid performs worse than *HA* in our experiments. We can also see that all clairvoyant algorithms except *Greedy* classify jobs and servers into different categories, and do not have good performance.

An interesting finding is that *Greedy* has the best performance in almost all cases, among all clairvoyant and non-clairvoyant algorithms. Clearly, *Greedy* being a clairvoyant algorithm uses the information on finishing time of jobs to its advantage. However, the other clairvoyant algorithms generally do not exhibit good performance, with the exception of *HA* for the case $\mu = 1$. It is important to note that *Greedy* is the only monotone *AnyFit* algorithm among the clairvoyant algorithms we have implemented. Note that, the other clairvoyant algorithms do not belong to the monotone *AnyFit* algorithms category.

Among non-clairvoyant algorithms, the best algorithms are generally *MTF* and *LastFit*, which are also both monotone *AnyFit* algorithms. Surprisingly, as in [15], in our experiments *BestFit* is one of the better algorithms, especially for higher values of μ, where its performance ratio equals or betters that of *MTF* and *LastFit*[1]. Recall that the worst-case competitive ratio of *BestFit* is unbounded as shown in [18].

In [15] and [22], two key factors, namely *alignment* and *packing* are identified as contributing to the performance of an algorithm for *RSiC*. The first factor is about how effectively jobs are aligned into servers in terms of their durations, while the second factor evaluates how tightly the jobs are packed together in servers. *AnyFit* algorithms (except *WorstFit*) do well in terms of packing, but do not consider alignment. *NextFit* tries to align jobs but does not do as well with packing. The authors of [15] stipulate that by assigning the next job to the server that has the *most recently arrived job*, *MTF* succeeds in terms of aligning jobs well in terms of time, while it also succeeds in packing since it is an *AnyFit* algorithm.

Greedy is also an *AnyFit* algorithm and therefore does well in terms of packing. By assigning to the server which contains the job that will *finish the last*, we may say that *Greedy* aligns even better than *MTF*.

A final interesting finding is that the difference in performance between the algorithms appears to narrow for $d = 5$. Further research is needed to understand this phenomenon, but one reason could be that because the sizes of jobs in different dimensions are chosen independently, it is harder for any algorithm to achieve a good packing, which diminishes the difference between the algorithms.

8 CONCLUSION AND OPEN PROBLEMS

In this paper, we studied the d-dimensional *RSiC* problem. We introduced a sub-family of *AnyFit* algorithms called monotone algorithms, and showed that monotone *AnyFit* algorithms achieve competitive ratio $\Theta(d\mu)$. We also introduced a natural clairvoyant algorithm *Greedy* that is a monotone *AnyFit* algorithm. We showed how to lift 1-dimensional algorithms for *RSiC* to work in d dimensions via a general direct-sum theorem. We also established a $\tilde{\Omega}(d\mu)$ lower bound for the non-clairvoyant setting under the assumption that $\mu \leq \log d - 2$. Finally, we conducted experiments that demonstrate that the newly introduced *Greedy* algorithm is among the best performing algorithms (both clairvoyant and non-clairvoyant) in the average-case scenario.

The following is a list of open problems that are suggested by this work:

- Close the gap between upper and lower bounds in Corollary 4.3. In particular, we conjecture that $\rho(Greedy) \leq d\mu + O(d)$.
- Does there exist a randomized algorithm for 1-dimensional clairvoyant *RSiC* with competitive ratio $O(1)$?
- Can one remove the constraint $\mu \leq \log d - 2$ in Theorems 6.1 and 6.5?
- For d-dimensional clairvoyant *RSiC*, is $\Omega(d\sqrt{\log \mu})$ a lower bound for *any* algorithm?
- Do the results from the experimental section carry over to real-life industrial instances?

[1]While *BestFit* also has excellent performance in the experiments of [22], it does not beat *MTF* for any value of d or μ. However, their experimental setup is somewhat different to ours and that in [15].

REFERENCES

[1] Y. Azar, I. R. Cohen, S. Kamara, and B. Shepherd. Tight bounds for online vector bin packing. In *Proceedings of the forty-fifth annual ACM Symposium on Theory of Computing (STOC)*, pages 961–970, 2013.

[2] Y. Azar and D. Vainstein. Tight bounds for clairvoyant dynamic bin packing. *ACM Transactions on Parallel Computing (TOPC)*, 6(3):1–21, 2019.

[3] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. *arXiv preprint arXiv:1707.01728*, 2017.

[4] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83:2047–2062, 2021.

[5] J. W.-T. Chan, T.-W. Lam, and P. W. Wong. Dynamic bin packing of unit fractions items. *Theoretical Computer Science*, 409(3):521–529, 2008.

[6] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, D. Vigo, et al. Bin packing approximation algorithms: Survey and classification. In *Handbook of Combinatorial Optimization*, pages 455–531. Springer, 2013.

[7] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin-packing—an updated survey. *Algorithm design for computer system design*, pages 49–106, 1984.

[8] E. G. Coffman, Jr, M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM Journal on Computing*, 12(2):227–258, 1983.

[9] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C.-C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory, Series A*, 21(3):257–298, 1976.

[10] O. Hadary, L. Marshall, I. Menache, A. Pan, E. E. Greeff, D. Dion, S. Dorminey, S. Joshi, Y. Chen, M. Russinovich, et al. Protean:{VM} allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 845–861, 2020.

[11] M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. In *Proceedings of the third annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 211–216. Society for Industrial and Applied Mathematics, 1992.

[12] X. Han, C. Peng, D. Ye, D. Zhang, and Y. Lan. Dynamic bin packing with unit fraction items revisited. *Information Processing Letters*, 110(23):1049–1054, 2010.

[13] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. Joint vm placement and routing for data center traffic engineering. In *2012 Proceedings IEEE INFOCOM*, pages 2876–2880. IEEE, 2012.

[14] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on computing*, 3(4):299–325, 1974.

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

[16] R. Karwayun. A dynamic energy efficient resource allocation scheme for heterogeneous clouds using bin-packing heuristics. *Algorithms*, 9:15–24, 2018.

[17] Y. Li and D. Pankratov. Online vector bin packing and hypergraph coloring illuminated: Simpler proofs and new connections. *arXiv preprint arXiv:2306.11241*, 2023.

[18] 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.

[19] 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, 2015.

[20] M. Masoori, L. Narayanan, and D. Pankratov. Renting servers in the cloud: The case of equal duration jobs. *arXiv preprint arXiv:2108.12486*, 2021.

[21] M. Masoori, L. Narayanan, and D. Pankratov. Renting servers in the cloud: Parameterized analysis of FirstFit. In *Proceedings of the 25th International Conference on Distributed Computing and Networking (ICDCN)*, page 199–208, 2024.

[22] A. Murhekar, D. Arbour, T. Mai, and A. B. Rao. Brief announcement: Dynamic vector bin packing for online resource allocation in the cloud. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 307–310, 2023.

[23] R. Ren. *Combinatorial algorithms for scheduling jobs to minimize server usage time*. PhD thesis, 2018.

[24] 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.

[25] A. L. Stolyar. An infinite server system with general packing constraints. *Operations Research*, 61(5):1200–1217, 2013.

[26] A. L. Stolyar and Y. Zhong. A large-scale service system with packing constraints: Minimizing the number of occupied servers. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):41–52, 2013.

[27] P. W. Wong, F. C. Yung, and M. Burcea. An 8/3 lower bound for online dynamic bin packing. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 44–53. Springer, 2012.