Competitiveness of Dynamic Bin Packing for Online Cloud Server Allocation

Runtian Ren, Xueyan Tang, *Senior Member, IEEE*, Yusen Li, and Wentong Cai

***Abstract*— Cloud-based systems often face the problem of dispatching a stream of jobs to run on cloud servers in an online manner. Each job has a size that defines the resource demand for running the job. Each job is assigned to run on a cloud server upon its arrival and the job departs after it completes. The departure time of a job, however, is not known at the time of its arrival. Each cloud server has a fixed resource capacity and the total resource demand of all the jobs running on a server cannot exceed its capacity at all times. The objective of job dispatching is to minimize the total cost of the servers used, where the cost of renting each cloud server is proportional to its running hours by "pay-as-you-go" billing. The above job dispatching problem can be modeled as a variant of the dynamic bin packing (DBP) problem known as MinUsageTime DBP. In this paper, we study the competitiveness bounds of MinUsageTime DBP. We establish an improved lower bound on the competitive ratio of Any Fit family of packing algorithms, and a new upper bound of $\mu + 3$ on the competitive ratio of the commonly used First Fit packing algorithm, where μ is the max/min job duration ratio. Our result significantly reduces the gap between the upper and lower bounds for the MinUsageTime DBP problem to a constant value independent of μ, and shows that First Fit packing is near optimal for MinUsageTime DBP.**

***Index Terms*— Dynamic bin packing, online algorithm, competitive ratio, cloud server allocation.**

I. Introduction

DYNAMIC Bin Packing (DBP) is a long-established combinatorial problem. The standard DBP problem [7] considers a set of items, each having an arrival time and a departure time. The items are to be packed into bins in an online manner such that the total size of the items in each bin does not exceed the bin capacity at all times. A bin is opened when it receives the first item and is closed when all items in the bin depart. The objective of DBP is to minimize the maximum number of concurrently open bins in the packing process.

In this paper, we consider a novel variant of the DBP problem that focuses on the duration of each bin's usage, i.e., the period from its opening to its closing. Our objective is to pack the items into bins to minimize the accumulated bin usage time. We refer to this variant of the DBP problem as *the MinUsageTime DBP problem* [17]. This problem is motivated by the online job dispatching problem arising from many cloud-based systems in which jobs may arrive at arbitrary times. Each job needs some amount of resources for execution and is assigned to run on a cloud server upon its arrival. The departure time of the job, however, is not known at the time of its arrival. The job is not reassigned to other servers during execution due to reasons such as high migration overheads and penalty. Each cloud server has a fixed resource capacity that restricts the total amount of resources needed by all the jobs running on the server at any time. The objective of job dispatching is to minimize the total cost of the servers used. The on-demand server instances (virtual machines) rented from public clouds such as Amazon EC2 are normally charged according to their running hours by "pay-as-you-go" billing [1]. Therefore, to minimize the total renting cost, it is equivalent to minimize the total running hours of the cloud servers. Such a job dispatching problem can be modeled exactly by the MinUsageTime DBP problem defined above, where the jobs and cloud servers correspond to the items and bins respectively. MinUsageTime DBP is related to interval scheduling for minimizing the busy times of machines [8], [14], [20]. However, the latter assumes that job durations are known whereas we assume that job durations are unknown at their arrivals in MinUsageTime DBP.

A typical application of the preceding job dispatching problem is cloud gaming. In a cloud gaming system, games are run and rendered on cloud servers, while players interact with the games via networked thin clients [11], [16]. Running each game instance demands a certain amount of GPU resources. When a play request is received by the cloud gaming provider, it should be assigned to a cloud server that has enough GPU resources to run the requested game instance. Several game instances can share the same cloud server provided that the server's GPU resources are not saturated. Each game instance keeps running on the assigned server until the player stops the game. Migrating a game instance from one server to another during execution is usually not allowed due to interruption to game play. Cloud gaming providers such as GaiKai rent servers from public clouds to run game instances [23]. Then, a natural problem faced by the cloud gaming provider is how to dispatch the play requests to cloud servers to minimize the total renting cost of the servers used.

Besides economic benefits, scheduling jobs with minimum server usage time also offers many advantages in other aspects. For example, the power usage of a server typically follows a

Manuscript received February 24, 2016; revised August 17, 2016 and October 11, 2016; accepted October 27, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Bejerano. Date of publication December 9, 2016; date of current version June 14, 2017. This work was supported in part by the National Research Foundation, Prime Minister's office, Singapore, under its IDM Futures Funding Initiative, in part by the Singapore Ministry of Education Academic Research Fund Tier 2 under Grant MOE2013-T2-2-067, and in part by the NSF of China under Grant 61602266. *(Corresponding authors: Xueyan Tang; Yusen Li.)*

R. Ren, X. Tang, and W. Cai are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: renr0002@ntu.edu.sg; asxytang@ntu.edu.sg; aswtcai@ntu.edu.sg).

Y. Li is with the Department of Computer Science and Security, Nankai University, China (e-mail: liyusen@nbjl.nankai.edu.cn).

Digital Object Identifier 10.1109/TNET.2016.2630052

linear model consisting of two components: a flat component representing the power consumed by an idle server and a variable component that is proportional to the server load [3]. While the aggregate amount of the variable components for all servers is mostly fixed for processing a given workload, the aggregate amount of the fixed components largely depends on the durations for which the servers are used. As a result, minimizing the server usage time implies minimizing the total energy consumption of all servers.

A. Previous Work on MinUsageTime DBP

The MinUsageTime DBP problem was first proposed in our earlier work [17], [18]. Any online bin packing algorithm can be applied to the problem. In [17] and [18], we analyzed the competitiveness of several classical bin packing algorithms, including Any Fit family of algorithms (which open a new bin only when no current open bin can accommodate an incoming item), First Fit and Best Fit (which are two particular Any Fit algorithms). We proved that the competitive ratio of any Any Fit packing algorithm cannot be better than $\mu + 1$, where μ is the ratio of the maximum item duration to the minimum item duration. The competitive ratio of Best Fit packing is not bounded for any given μ. If all the item sizes are smaller than $\frac{1}{\beta}$ of the bin capacity ($\beta > 1$ is a constant), the competitive ratio of First Fit packing has an upper bound of $\frac{\beta}{\beta-1} \cdot \mu + \frac{3\beta}{\beta-1} + 1$. For the general case, the competitive ratio of First Fit has an upper bound of $2\mu + 7$. We further proposed a Hybrid First Fit algorithm that classifies and packs items based on their sizes to achieve a competitive ratio no larger than $\frac{8}{7}\mu + \frac{55}{7}$ [17]. We also indicated a lower bound of μ on the competitive ratio of any online packing algorithm [17]. Kamali and López-Ortiz [13] later presented a formal proof of this lower bound. They also showed that Next Fit packing has a competitive ratio bounded above by $2\mu + 1$.

B. Contributions of This Paper

In this paper, we significantly tighten the gap between the upper and lower bounds for the MinUsageTime DBP problem, reducing the gap to a constant value independent of μ. We first establish an improved lower bound on the competitive ratio of Any Fit family of packing algorithms for the MinUsageTime DBP problem. Then, we develop new approaches to the competitive analysis of the commonly used First Fit packing algorithm for MinUsageTime DBP. In an earlier version of this paper [25], we proved an upper bound of $\mu + 4$ on the competitive ratio of First Fit packing. This paper further improves the analysis and establishes a new upper bound of $\mu + 3$ on the competitive ratio of First Fit packing. This new bound is the current best upper bound for the MinUsageTime DBP problem.[1,2] While all the aforementioned upper bounds have multiplicative factors larger than 1 for μ, our new upper bound has a multiplicative factor 1 for μ. Our result indicates that First Fit packing is near optimal for the MinUsageTime DBP problem.

The rest of this paper is organized as follows. Section II reviews the related work. Section III provides some preliminaries. Section IV presents the improved lower bound on the competitive ratio of Any Fit family of packing algorithms. Section V carries out the competitive analysis of First Fit packing. Section VI compares the competitiveness of First Fit packing and Next Fit packing, and shows that the later is inherently worse. Finally, Section VII concludes the paper.

II. Related Work

The classical bin packing problem aims to pack a set of items into the minimum number of bins. It is well known that even the offline version of classical bin packing is NP-hard [9]. In the online version, each item must be placed in a bin without the knowledge of subsequent items. Once placed, the items are not allowed to move to other bins. The competitive ratios of various algorithms for classical online bin packing have been extensively studied [2], [22].

Dynamic bin packing (DBP) is a generalization of the classical bin packing problem [7]. In DBP, items may arrive and depart at arbitrary times. The objective is to minimize the maximum number of bins concurrently used in the packing. A large amount of research work has also been done to analyze the competitive ratios of various algorithms for DBP [5]–[7], [12]. However, standard DBP does not consider the duration of bin usage. In contrast, the MinUsageTime DBP problem we have defined aims to minimize the total amount of time the bins are used [17].

Interval scheduling [15] is another problem related to our MinUsageTime DBP problem. In the basic interval scheduling, each job is associated with one or several alternative time intervals for execution. The goal of scheduling is to maximize the number of jobs executed on a server that can process only a single job at any time [10], [24]. Recently, some works have studied interval scheduling with bounded parallelism, where each server can process multiple jobs simultaneously up to a fixed maximum number [8], [20]. A server is considered busy if at least one job is running on it. The objective is to minimize the total busy time of all servers to complete a given set of jobs. This target resembles the one we study in this paper. However, there is a crucial difference between interval scheduling and our MinUsageTime DBP problem. The ending times of jobs are known in interval scheduling, but the departure time of an item is not known at the time of its packing in our problem. Moreover, jobs may have arbitrary resource demands in our problem, so the maximum number of jobs that can run concurrently on a server is not fixed. Recently, Khandekar *et al.* [14] proposed a 5-approximation offline algorithm for scheduling interval jobs with arbitrary resource demands to minimize the total busy time of servers and we further developed a 4-approximation offline algorithm [21]. Maguluri and Srikant [19] conducted stochastic analysis to study the throughput of processing

[1]More precisely, our new upper bound $\mu+3$ is better than the bound $2\mu+1$ of Next Fit when $\mu > 2$.

[2]Hybrid First Fit and Next Fit packing algorithms that classify items based on their sizes can achieve competitive ratios of $\mu + 5$ [18] and $\mu + 2$ [13] respectively, but to do so, these algorithms require the max/min item duration ratio μ to be known a priori and thus are semi-online in nature.

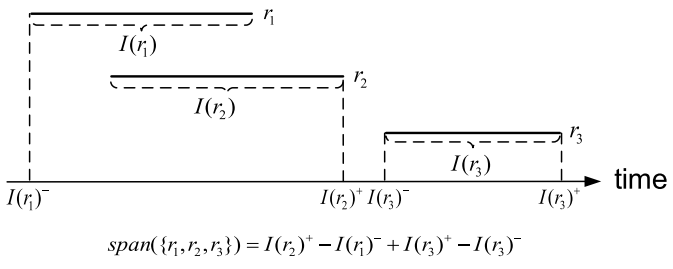


Fig. 1. Span of an item list.

jobs with unknown durations assuming that there are a fixed number of servers available and the jobs have a fixed set of resource demand choices.

III. Preliminaries

A. Notations and Definitions

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

For any item r, let $I(r)$ denote the time interval from r's arrival to its departure. We say that item r is *active* during the interval $I(r)$, and we refer to $I(r)$ as r's *active interval*. The length of $I(r)$ is called the *item duration*. Let $s(r)$ denote the size of item r. For notational convenience, for a list of items $\mathcal{R}$, we also use $s(\mathcal{R})$ to denote the total size of all the items in $\mathcal{R}$, i.e., $s(\mathcal{R}) = \sum_{r\in\mathcal{R}} s(r)$. In addition, we refer to the time duration in which at least one item in $\mathcal{R}$ is active as the *span* of $\mathcal{R}$ and denote it by $span(\mathcal{R})$ (see Figure 1). When the context is clear, we also use $span(\mathcal{R})$ to refer to the time interval(s) in which at least one item is active.

B. Packing Algorithms

In the bin packing process, a bin is *opened* when it receives the first item. When all the items in a bin depart, the bin is *closed*. At any time, the total size of all the active items in an open bin is referred to as the *bin level*.

Any Fit refers to a family of packing algorithms. An Any Fit algorithm never opens a new bin for an incoming item if the item can fit in any current open bin.

We shall focus on the following First Fit algorithm for online bin packing, which is a particular Any Fit algorithm. Each time when a new item arrives, if there are one or more open bins that can accommodate the new item, First Fit places the item in the bin which was opened earliest among these bins. If no open bin can accommodate the new item, then a new bin is opened to receive the item.

C. Competitive Ratio

The performance of an online algorithm is usually measured by its *competitive ratio*, i.e., the worst-case ratio between the solution constructed by the algorithm and an optimal solution [4].

Without loss of generality, we assume that the bins all have unit capacity. Given a list of items $\mathcal{R}$, let $OPT(\mathcal{R}, t)$ denote the minimum achievable number of bins into which all the items active at time t can be repacked. Then, the total bin usage time of an optimal offline adversary that can repack everything at any time is given by

$$OPT_{total}(\mathcal{R}) = \int_{span(\mathcal{R})} OPT(\mathcal{R}, t)\ \mathrm{d}t.$$

As shown in our earlier work [17], [18], it is easy to obtain the following lower bounds on $OPT_{total}(\mathcal{R})$:

Proposition 1: $OPT_{total}(\mathcal{R}) \geq \sum_{r\in\mathcal{R}} \Big(s(r)\cdot l(I(r))\Big)$.

Proposition 2: $OPT_{total}(\mathcal{R}) \geq span(\mathcal{R})$.

The first bound is derived by assuming that no capacity of any bin is wasted at any time, where $s(r)\cdot l(I(r))$ is the *time-space demand* of an item r. The second bound is derived from the fact that at least one bin must be used at any time when at least one item is active.

Let $A_{total}(\mathcal{R})$ denote the total bin usage time by applying an online packing algorithm A to the list of items $\mathcal{R}$. The competitive ratio of algorithm A is the maximum ratio of $A_{total}(\mathcal{R})/OPT_{total}(\mathcal{R})$ over all instances of item lists $\mathcal{R}$.

IV. A Lower Bound on the Competitive Ratio of Any Fit Family of Packing Algorithms

In this section, we establish an improved lower bound on the competitive ratio of Any Fit family of packing algorithms. For any list of items $\mathcal{R}$, let $\mu = \frac{\max_{r\in\mathcal{R}} l(I(r))}{\min_{r\in\mathcal{R}} l(I(r))}$ denote the ratio of the maximum item duration to the minimum item duration among all the items in $\mathcal{R}$. Without loss of generality, we shall assume that the minimum item duration is 1, and the maximum item duration is μ. In [18], the competitive ratio of any Any Fit packing algorithm is proved to have a lower bound of $\mu + 1$. Here, we introduce a new instance to show that the competitive ratio of Any Fit packing is at least $\frac{4\mu+3}{\mu+2}$ when $1 \leq \mu < \frac{1+\sqrt{5}}{2}$ and at least $\mu + 1$ when $\mu \geq \frac{1+\sqrt{5}}{2}$.

Let n be an integer at least 3. At time 0, let $(n-1)$ items of size $\frac{1}{n}$ arrive, followed by $(n-1)$ items of size $\frac{n-1}{n}$. As shown in Figure 2(a), Any Fit packing needs to open n bins to pack all these items. The first bin is used to pack the $(n-1)$ items of size $\frac{1}{n}$ and the other $(n-1)$ bins are used to pack one item of size $\frac{n-1}{n}$ each. At time 1, let three items of size $\frac{2}{3n}$ and $(n-2)$ items of size $\frac{1}{n}$ arrive in sequence. Each open bin has enough space left to pack only one of these items. Thus, as shown in Figure 2(b), a new bin has to be opened at time 1 to pack the item arrived last (of size $\frac{1}{n}$). As soon as these $(n+1)$ items are placed in the bins, let all the items arriving at time 0 depart (see Figure 2(c)). At time $(\mu+1)$, let all the items arriving at time 1 depart.

As shown in Figure 2, in the above packing process, n bins are open from time 0 to 1, and $(n+1)$ bins are open from time 1 to $(\mu+1)$. Therefore, the total bin usage time of Any Fit packing is $AF_{total}(\mathcal{R}) = n+(n+1)\cdot\mu$. On the other hand, in the optimal packing, from time 0 to 1, every pair of items with sizes $\frac{1}{n}$ and $\frac{n-1}{n}$ can be packed into one bin so that only $(n-1)$ bins are needed. From time 1 to $(\mu+1)$, all the items arriving at time 1 can be packed into one new bin since their total size is $3\cdot\frac{2}{3n}+(n-2)\cdot\frac{1}{n} = 1$. Therefore, the total bin usage

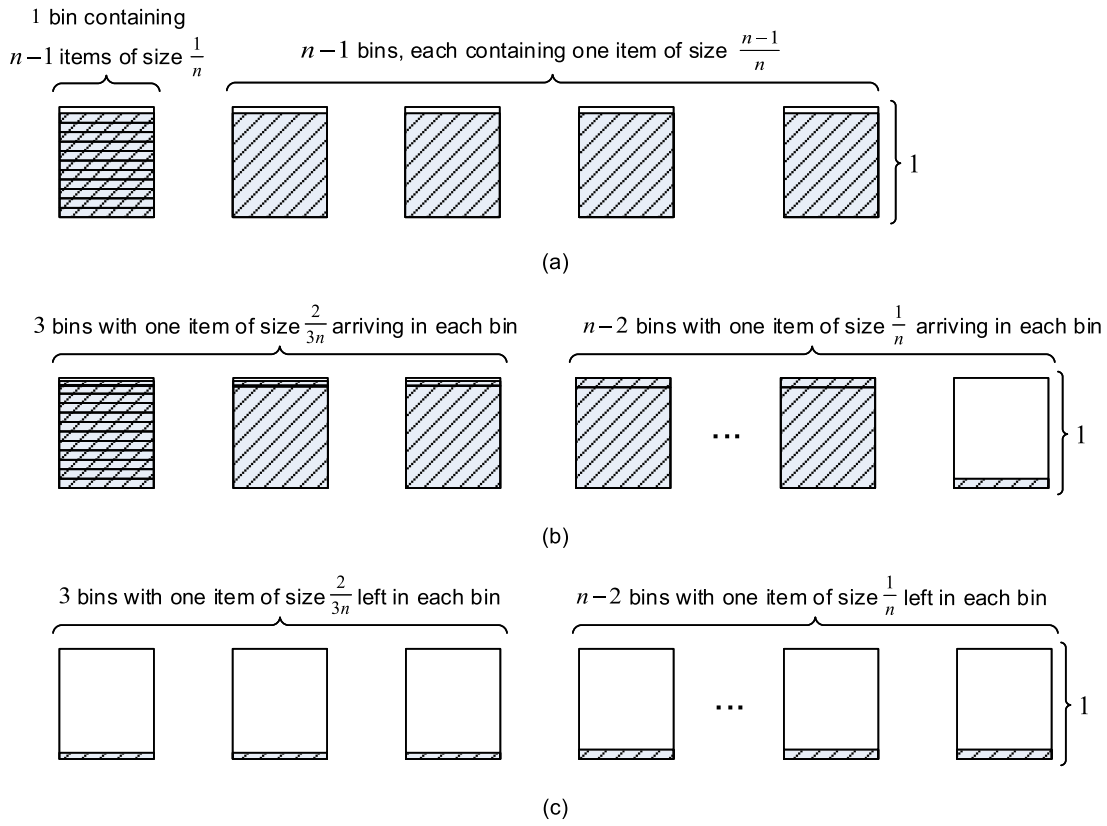


Fig. 2. Bin levels by an Any Fit packing algorithm. (a) Period $[0, 1)$. (b) At time 1. (c) Period $[1, \mu{+}1)$.

time of the optimal packing is $OPT_{total}(\mathcal{R}) = (n-1)+\mu$. It follows that

$$\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})} = \frac{n+(n+1)\cdot\mu}{(n-1)+\mu} = \mu+1-\frac{\mu^2-\mu-1}{n-1+\mu}.$$

It is easy to see that when $\mu^2-\mu-1<0$ (i.e., $1\le\mu<\frac{1+\sqrt{5}}{2}$), $\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})}>\mu+1$ and the ratio $\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})}$ decreases with n. When n is set to 3, the above ratio reaches the maximum value at $\mu+1-\frac{\mu^2-\mu-1}{n-1+\mu}=\frac{4\mu+3}{\mu+2}$.

When $\mu^2-\mu-1\ge 0$ (i.e., $\mu\ge\frac{1+\sqrt{5}}{2}$), $\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})}\le\mu+1$ and the ratio $\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})}$ increases with n. As n goes towards infinity, the above ratio can be made arbitrarily close to $\mu+1$. Hence, we have the following conclusion.

Theorem 1: For the MinUsageTime DBP problem, the competitive ratio of any Any Fit packing algorithm is at least $\frac{4\mu+3}{\mu+2}$ when $\mu<\frac{1+\sqrt{5}}{2}$ and at least $\mu+1$ otherwise.

V. An Upper Bound on the Competitive Ratio of First Fit Packing

We now analyze the competitive ratio of First Fit packing for the MinUsageTime DBP problem. The main idea is to divide the bin usage time resulting from First Fit packing into two portions. The first portion is equal to the span of the item list packed and is thus capped by the lower bound of Proposition 2. The second portion consists of the bin usage periods in which there are at least two open bins. According to the First Fit packing rule, whenever a new bin is opened for an incoming item r, the sum of r's size and the level of any open bin must exceed the bin capacity. Exploiting this observation, we break the usage period of each bin into subperiods and strategically charge the length of each subperiod to the time-space demand of relevant items. In this way, we bound the second portion of bin usage time with respect to the total time-space demand of the item list packed and hence the lower bound of Proposition 1. Combining the two portions of bin usage time, we show that the competitive ratio of First Fit packing is bounded by $\mu+3$.

Suppose a total of m bins $b_1, b_2, \ldots, b_m$ are used by First Fit packing to pack a list of items $\mathcal{R}$. For each bin b_k, let $U_k=[U_k^-, U_k^+)$ denote the usage period of b_k, i.e., the period from the time when b_k is opened to the time when b_k is closed. Then, the total bin usage time of First Fit packing is given by the total length of the usage periods of all the bins used, i.e.,

$$FF_{total}(\mathcal{R})=\sum_{k=1}^{m} l(U_k).$$

Without loss of generality, assume that the bins are indexed in the chronological order of their openings, i.e., $U_1^-\le U_2^-\le\cdots\le U_m^-$. For each bin b_k, let E_k be the latest closing time of all the bins that are opened before b_k, i.e., $E_k=\max\big\{\{U_i^+|1\le i<k\}\cup\{U_k^-\}\big\}$. We divide the usage period U_k of each bin into two parts: V_k and W_k. V_k is the period $[U_k^-, \min\{U_k^+, E_k\})$. $W_k=U_k-V_k=[\min\{U_k^+, E_k\}, U_k^+)$ is the remaining period. Figure 3 shows an example of these definitions.

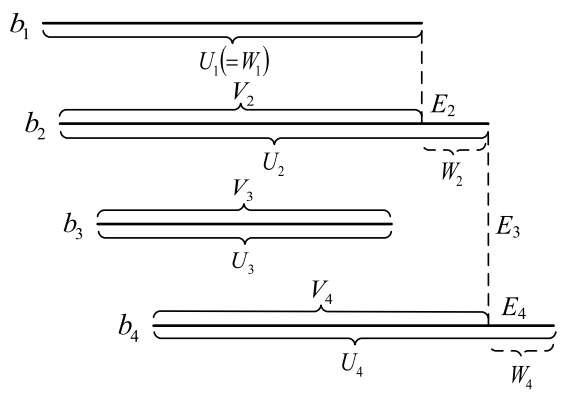


Fig. 3. An example of usage periods.

According to the definitions, we have $l(U_k) = l(V_k) + l(W_k)$. Clearly, for any two different bins b_{k_1} and b_{k_2}, $W_{k_1} \cap W_{k_2} = \emptyset$. It is also easy to see that

$$span(\mathcal{R}) = l\Big(\bigcup_{k=1}^{m} W_k\Big) = \sum_{k=1}^{m} l(W_k).$$

Moreover, $V_1 = \emptyset$ and hence $l(V_1) = 0$. Therefore,

$$\begin{aligned} FF_{total}(\mathcal{R}) &= \sum_{k=1}^{m} l(U_k) \\ &= \Big(\sum_{k=1}^{m} l(V_k)\Big) + \Big(\sum_{k=1}^{m} l(W_k)\Big) \\ &= \Big(\sum_{k\geq 2} l(V_k)\Big) + span(\mathcal{R}). \end{aligned} \tag{1}$$

For each k where $2 \leq k \leq m$, let $\mathcal{R}_k$ denote the set of all the items placed in bin b_k by First Fit packing. Let $\mathcal{R}'_k$ be a minimal subset of $\mathcal{R}_k$ such that the union of their active intervals completely covers the period V_k. That is, any subset of items $Q \subset \mathcal{R}'_k$ cannot fully cover the period V_k. It is easy to see that all the items in $\mathcal{R}'_k$ must have distinct arrival times. This is because if two items have the same arrival times, the active interval of one item must be fully contained in that of the other item and thus can be removed from $\mathcal{R}'_k$, contradicting that $\mathcal{R}'_k$ is minimal.

Let $n(k)$ denote the number of items in $\mathcal{R}'_k$ and let $r_{k,1}$, $r_{k,2}$, $\ldots$, $r_{k,n(k)}$ be the items in $\mathcal{R}'_k$ sorted according to their arrival times, i.e.,

$$I(r_{k,1})^- < I(r_{k,2})^- < \cdots < I(r_{k,n(k)})^-.$$

Then, these items must satisfy the following properties. First, it holds that

$$I(r_{k,1})^+ < I(r_{k,2})^+ < \cdots < I(r_{k,n(k)})^+.$$

Otherwise, if $I(r_{k,i})^+ \geq I(r_{k,j})^+$ for some $i < j$, the active interval of item $r_{k,j}$ is fully contained in the active interval of item $r_{k,i}$, which contradicts the definition of $\mathcal{R}'_k$. Second, for each $r_{k,i}$, we have $I(r_{k,i})^- < V_k^+$, i.e., $r_{k,i}$ arrives in the period V_k. Otherwise, $r_{k,i}$ can be removed from $\mathcal{R}'_k$ without compromising its coverage over V_k.

If $V_k = \emptyset$, we have $\mathcal{R}'_k = \emptyset$. If $V_k \neq \emptyset$, as shown in Figure 4, we can split V_k into $n(k)$ disjoint periods at the arrival times of the items in $\mathcal{R}'_k$:

$$\begin{aligned} X(r_{k,1}) &= [I(r_{k,1})^-, I(r_{k,2})^-), \\ X(r_{k,2}) &= [I(r_{k,2})^-, I(r_{k,3})^-), \\ &\cdots\cdots \\ X(r_{k,n(k)-1}) &= [I(r_{k,n(k)-1})^-, I(r_{k,n(k)})^-), \\ X(r_{k,n(k)}) &= [I(r_{k,n(k)})^-, V_k^+). \end{aligned}$$

We refer to the above periods as the X-periods of items $r_{k,1}$, $r_{k,2}$, $\cdots$, $r_{k,n(k)}$ respectively. Obviously, the total length of the X-periods is $l(V_k)$, i.e.,

$$\sum_{r\in\mathcal{R}'_k} l\big(X(r)\big) = l(V_k). \tag{2}$$

We define

$$d_k = \sum_{r\in\mathcal{R}'_k} s(r)\cdot l\big(X(r)\big).$$

Clearly, d_k is a lower bound on the total time-space demand of all the items placed in bin b_k since the X-period of each item is shorter than or equal to its active interval:

$$\begin{aligned} d_k &\leq \sum_{r\in\mathcal{R}'_k} s(r)\cdot l\big(I(r)\big) \\ &\leq \sum_{r\in\mathcal{R}_k} s(r)\cdot l\big(I(r)\big). \end{aligned} \tag{3}$$

Recall that each item r in $\mathcal{R}'_k$ is placed in bin b_k in the period V_k. By the definition of V_k, there must exist at least one open bin with an index lower than k at time $X(r)^- = I(r)^-$ when r arrives. Among all the open bins with indices lower than k at time $X(r)^-$, we define the last opened bin (the bin with the highest index) as the *supplier bin* of $X(r)$. Note that different items in $\mathcal{R}'_k$ may have different supplier bins as shown in Figure 5. By the definition of First Fit packing, the level of $X(r)$'s supplier bin at time $X(r)^-$ plus the size of item r must be larger than 1 (the bin capacity). Let $P(r)$ be the set of all the active items already packed in $X(r)$'s supplier bin at the arrival time of r. Then,

$$s(r) + \sum_{\hat{r}\in P(r)} s(\hat{r}) > 1. \tag{4}$$

We define

$$d^* = \sum_{k\geq 2}\Big(\sum_{r\in\mathcal{R}'_k}\Big(\sum_{\hat{r}\in P(r)} s(\hat{r})\cdot l\big(X(r)\big)\Big)\Big).$$

It then follows from (2) and (4) that

$$\begin{aligned} &\Big(\sum_{k\geq 2} d_k\Big) + d^* \\ &\quad= \sum_{k\geq 2}\Big(\sum_{r\in\mathcal{R}'_k}\Big(s(r) + \sum_{\hat{r}\in P(r)} s(\hat{r})\Big)\cdot l\big(X(r)\big)\Big) \\ &\quad> \sum_{k\geq 2}\Big(\sum_{r\in\mathcal{R}'_k} l\big(X(r)\big)\Big) \\ &\quad= \sum_{k\geq 2} l(V_k). \end{aligned} \tag{5}$$

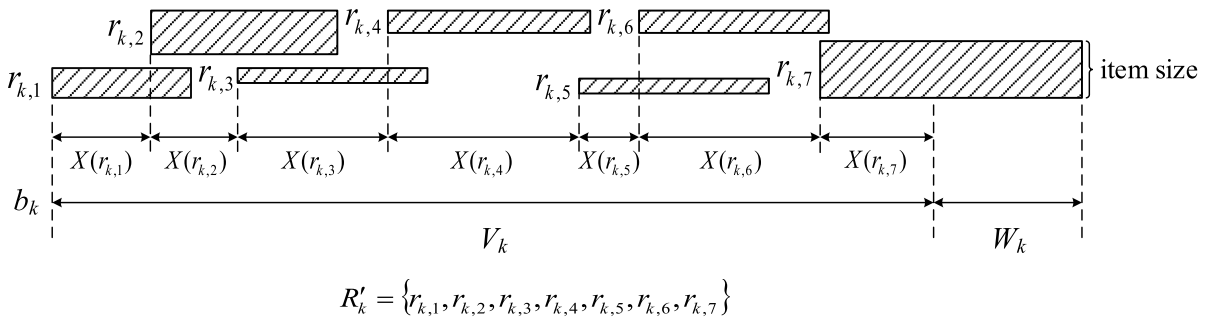


Fig. 4. Splitting V_k for bin b_k.

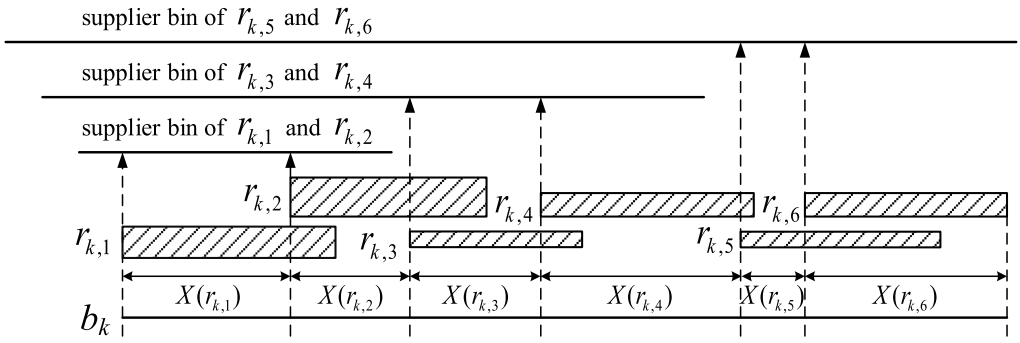


Fig. 5. An example of supplier bins.

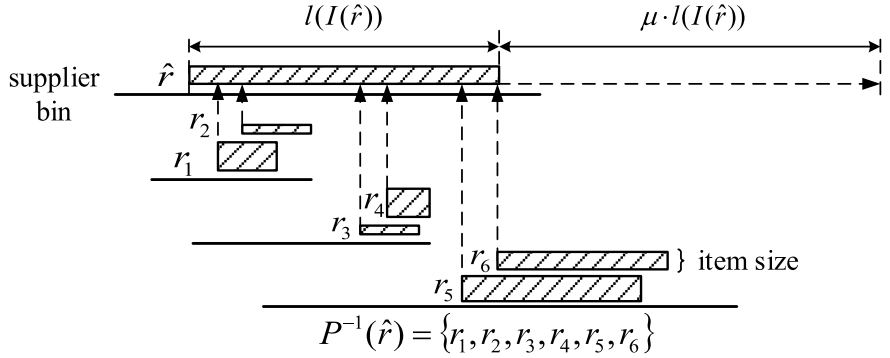


Fig. 6. An example of $P^{-1}(\hat{r})$.

We next show that d^* is bounded by $(\mu+1)\cdot\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big)$, where $\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big)$ is the total time-space demand of all the items in $\mathcal{R}$.

Proposition 3: $d^* \le (\mu+1)\cdot\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big)$.

Proof: We define

$$\widehat{\mathcal{R}} = \bigcup_{k\ge 2}\Big(\bigcup_{r\in\mathcal{R}'_k} P(r)\Big).$$

Obviously, $\widehat{\mathcal{R}} \subseteq \mathcal{R}$. For each item $\hat{r}\in\widehat{\mathcal{R}}$, let $P^{-1}(\hat{r})$ denote the set of all the items r in $\bigcup_{k\ge 2}\mathcal{R}'_k$ such that $\hat{r}\in P(r)$. Note that the items in $P^{-1}(\hat{r})$ may come from different bins. Figure 6 shows an example of $P^{-1}(\hat{r})$.

We start by studying the X-period lengths of the items in $P^{-1}(\hat{r})$. For each item $r\in P^{-1}(\hat{r})$, since item $\hat{r}$ has already been packed when r arrives, $\hat{r}$ cannot arrive later than r. Thus,

$$X(r)^- = I(r)^- \ge I(\hat{r})^-. \tag{6}$$

By definition, $I(\hat{r})$ overlaps with the X-period of each item in $P^{-1}(\hat{r})$, which suggests that $X(r)^- < I(\hat{r})^+$. Recall that each item has a duration at least 1 and at most μ. This implies that

$$l(X(r)) \le l(I(r)) \le \mu \le \mu\cdot l(I(\hat{r})).$$

Therefore,

$$\begin{aligned} X(r)^+ &= X(r)^- + l(X(r)) \\ &\le I(\hat{r})^+ + \mu\cdot l(I(\hat{r})). \end{aligned} \tag{7}$$

We now show that all the X-periods of the items in $P^{-1}(\hat{r})$ are disjoint. Consider any two items r_1 and r_2 in $P^{-1}(\hat{r})$.

If r_1 and r_2 are placed in the same bin, by the definition of X-periods, $X(r_1)$ cannot intersect with $X(r_2)$. If r_1 and r_2 are placed in two different bins, without loss of generality, suppose r_1 and r_2 are placed in bins b_{j_1} and b_{j_2} respectively where $j_1 < j_2$. Note that $X(r_1)$ and $X(r_2)$ share the same supplier bin that has an index lower than j_1 and j_2. According to the definition of supplier bin, when r_2 is packed into bin b_{j_2}, bin b_{j_1} must be closed. Otherwise, r_2's supplier bin must have an index at least j_1, which leads to a contradiction. Since b_{j_1} is closed when r_2 arrives, r_1 must have departed when r_2 arrives. This suggests that $X(r_1)$ cannot overlap with $X(r_2)$.

Therefore, it follows from (6) and (7) that

$$\begin{aligned}\sum_{r\in P^{-1}(\hat{r})} l\big(X(r)\big) &\le \max_{r\in P^{-1}(\hat{r})} X(r)^+ - \min_{r\in P^{-1}(\hat{r})} X(r)^- \\ &\le I(\hat{r})^+ + \mu\cdot l(I(\hat{r})) - I(\hat{r})^- \\ &= (\mu+1)\cdot l(I(\hat{r})).\end{aligned}$$

Thus, we have

$$\begin{aligned}d^* &= \sum_{k\ge 2}\Big(\sum_{r\in\mathcal{R}'_k}\Big(\sum_{\hat{r}\in P(r)} s(\hat{r})\cdot l\big(X(r)\big)\Big)\Big) \\ &= \sum_{r\in\cup_{k\ge 2}\mathcal{R}'_k}\Big(\sum_{\hat{r}\in P(r)} s(\hat{r})\cdot l(X(r))\Big) \\ &= \sum_{\hat{r}\in\widehat{\mathcal{R}}}\Big(s(\hat{r})\cdot\Big(\sum_{r\in P^{-1}(\hat{r})} l\big(X(r)\big)\Big)\Big) \\ &\le \sum_{\hat{r}\in\widehat{\mathcal{R}}}\Big(s(\hat{r})\cdot(\mu+1)\cdot l\big(I(\hat{r})\big)\Big) \\ &\le (\mu+1)\cdot\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big).\end{aligned}$$

Hence, the proposition is proven. □

Following from (1), (3), (5) and Proposition 3, the total bin usage time of First Fit packing satisfies

$$\begin{aligned}FF_{total}(\mathcal{R}) &= \Big(\sum_{k\ge 2} l(V_k)\Big) + span(\mathcal{R}) \\ &< \Big(\sum_{k\ge 2} d_k\Big) + d^* + span(\mathcal{R}) \\ &\le \sum_{k\ge 2}\Big(\sum_{r\in\mathcal{R}_k} s(r)\cdot l(I(r))\Big) \\ &\quad +(\mu+1)\cdot\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big) + span(\mathcal{R}) \\ &\le \sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big) \\ &\quad +(\mu+1)\cdot\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big) + span(\mathcal{R}) \\ &= (\mu+2)\cdot\sum_{r\in\mathcal{R}}\big(s(r)\cdot l(I(r))\big) + span(\mathcal{R}) \\ &\le (\mu+2)\cdot OPT_{total}(\mathcal{R}) + OPT_{total}(\mathcal{R}) \\ &= (\mu+3)\cdot OPT_{total}(\mathcal{R}),\end{aligned}$$

where the last inequality follows from the bounds given in Propositions 1 and 2. Thus, First Fit packing has a competitive ratio at most $\mu+3$.

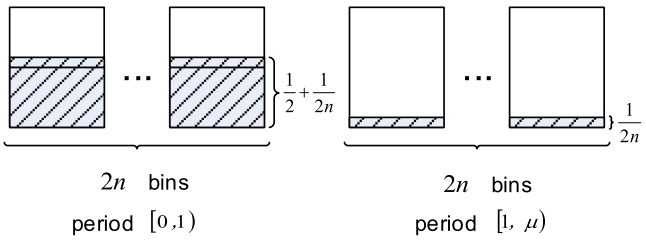


Fig. 7. Bin levels by Next Fit packing.

Theorem 2: First Fit packing is $(\mu+3)$-competitive for the MinUsageTime DBP problem.

It has been proved that for MinUsageTime DBP, the competitive ratio of any online packing algorithm cannot be better than μ [17], [18]. Thus, the result of Theorem 1 indicates that First Fit packing is near optimal for MinUsageTime DBP.

VI. Comparison Between First Fit and Next Fit

The Next Fit packing algorithm keeps exactly one bin available for receiving new items at any time. If an incoming item does not fit in the available bin, the available bin is marked unavailable and a new bin is opened (and marked available) to receive the new item. Unavailable bins are never marked available again and are closed when all the items in the bin depart.

Kamali and López-Ortiz [13] has shown that the competitive ratio of Next Fit packing has an upper bound of $2\mu+1$ for the MinUsageTime DBP problem. In this section, we show that the competitive ratio of Next Fit has a lower bound of 2μ by constructing an example. This implies that the multiplicative factor 2 of μ is inevitable in the competitive ratio of Next Fit. Therefore, First Fit is the only known packing algorithm so far whose competitive ratio has a multiplicative factor 1 for μ.

Let n be an integer no less than 3. At time 0, let $2n$ pairs of items arrive in sequence. The first item of each pair has a size $\frac{1}{2}$ and the second item has a size $\frac{1}{2n}$. At time 1, let all the items of size $\frac{1}{2}$ depart. At time μ, let all the items of size $\frac{1}{2n}$ depart.

When Next Fit packing is applied, each pair of items are placed in a separate bin because the first item of the pair (of size $\frac{1}{2}$) cannot fit in the previous open bin which has a level $\frac{1}{2}+\frac{1}{2n}$. Thus, as shown in Figure 7, $2n$ bins are opened from time 0 to μ. Therefore, the total bin usage time of Next Fit packing is $2n\mu$. On the other hand, in the optimal packing, every two items of size $\frac{1}{2}$ can be packed into one bin so that only n bins are enough to store all the items of size $\frac{1}{2}$ from time 0 to 1. All the items of size $\frac{1}{2n}$ can be packed into only one bin. Therefore, the total bin usage time of the optimal packing is $n+\mu$. The ratio between the bin usage times of Next Fit packing and optimal packing is $\frac{2n\mu}{n+\mu}$, which can be made arbitrarily close to 2μ as n goes towards infinity. Thus, the competitive ratio of Next Fit packing has a lower bound of 2μ.

VII. Concluding Remarks

The MinUsageTime DBP problem models online job dispatching to cloud servers. In this paper, we have established an improved lower bound on the competitive ratio of Any Fit family of packing algorithms. We have developed new

approaches to analyze the competitiveness of the commonly used First Fit packing algorithm for the MinUsageTime DBP problem, and established a new upper bound of $\mu + 3$ on its competitive ratio, which is the current best upper bound for the MinUsageTime DBP problem. Our result significantly reduces the gap between the upper and lower bounds for the MinUsageTime DBP problem to a constant independent of μ, and indicates that First Fit packing is near optimal for MinUsageTime DBP. One direction for future work is to extend the MinUsageTime DBP problem to the multi-dimensional version to model multiple types of resources (e.g., CPU and memory) for online cloud server allocation.

REFERENCES

[1] *Amazon EC2 Pricing*, accessed on Sep. 1, 2015[Online]. Available: http://aws.amazon.com/ec2/pricing/

[2] J. Balogh, J. Békési, and G. Galambos, "New lower bounds for certain classes of bin packing algorithms," *Approximation and Online Algorithms* (Lecture Notes in Computer Science), vol. 6534, pp. 25–36, 2011.

[3] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[4] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, vol. 53. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[5] J. W.-T. Chan, T.-W. Lam, and P. W. Wong, "Dynamic bin packing of unit fractions items," *Theor. Comput. Sci.*, vol. 409, no. 3, pp. 521–529, 2008.

[6] J. W.-T. Chan, P. W. H. Wong, and F. C. Yung, "On dynamic bin packing: An improved lower bound and resource augmentation analysis," *Computing and Combinatorics* (Lecture Notes in Computer Science), vol. 4112. 2006, pp. 309–319.

[7] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Dynamic bin packing," *SIAM J. Comput.*, vol. 12, no. 2, pp. 227–258, 1983.

[8] M. Flammini *et al.*, "Minimizing total busy time in parallel scheduling with application to optical networks," in *Proc. 23rd IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2009, pp. 1–12.

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

[10] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. San Diego, CA, USA: Academic, 1980.

[11] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 10, no. 1s, Jan. 2014, Art. no. 10.

[12] Z. Ivkovic and E. L. Lloyd, "Fully dynamic algorithms for bin packing: Being (Mostly) myopic helps," *SIAM J. Comput.*, vol. 28, no. 2, pp. 574–611, 1998.

[13] S. Kamali and A. López-Ortiz, "Efficient online strategies for renting servers in the cloud," in *SOFSEM, Theory and Practice of Computer Science* (Lecture Notes in Computer Science), vol. 8939. 2015, pp. 277–288.

[14] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir, "Real-time scheduling to minimize machine busy times," *J. Scheduling*, vol. 18, no. 6, pp. 561–573, 2015.

[15] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity," *Handbooks Oper. Res. Manage. Sci.*, vol. 4, pp. 445–522, 1993.

[16] Y. Li, Y. Deng, R. Seet, X. Tang, and W. Cai, "MASTER: Multi-platform application streaming toolkits for elastic resources," in *Proc. 23rd ACM Int. Conf. Multimedia (MM)*, 2015, pp. 805–806.

[17] Y. Li, X. Tang, and W. Cai, "On dynamic bin packing for resource allocation in the cloud," in *Proc. 26th ACM Symp. Parallelism Algorithms Archit. (SPAA)*, 2014, pp. 2–11.

[18] Y. Li, X. Tang, and W. Cai, "Dynamic bin packing for on-demand cloud resource allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 157–170, Jan. 2016.

[19] S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1938–1951, Dec. 2014.

[20] G. B. Mertzios, M. Shalom, A. Voloshin, P. W. Wong, and S. Zaks, "Optimizing busy time on parallel machines," in *Proc. 26th IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2012, pp. 238–248.

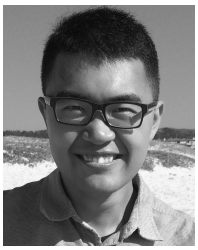
[21] R. Ren and X. Tang, "Clairvoyant dynamic bin packing for job scheduling with minimum server usage time," in *Proc. 28th ACM Symp. Parallelism Algorithms Archit. (SPAA)*, 2016, pp. 227–237.

[22] S. S. Seiden, "On the online bin packing problem," *J. ACM*, vol. 49, no. 5, pp. 640–671, 2002.

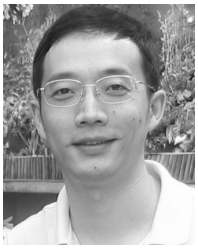
[23] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, Jul./Aug. 2013.

[24] F. C. R. Spieksma, "On the approximability of an interval scheduling problem," *J. Scheduling*, vol. 2, pp. 215–227, 1999.

[25] X. Tang, Y. Li, R. Ren, and W. Cai, "On first fit bin packing for online cloud server allocation," in *Proc. 30th IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 323–332.



Runtian Ren received the B.Sc. degree in mathematics and applied mathematics from the University of Science and Technology of China in 2014. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanyang Technological University, Singapore.



Xueyan Tang received the B.Eng. degree in computer science and engineering from Shanghai Jiao Tong University in 1998, and the Ph.D. degree in computer science from The Hong Kong University of Science and Technology in 2003. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, cloud computing, mobile and pervasive computing, and wireless sensor networks. He has served as an Associate Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and a Program Co-Chair of the IEEE ICPADS 2012 and the IEEE CloudCom 2014.

Yusen Li received the Ph.D. degree from Nanyang Technological University in 2014. He is currently an Associate Professor with the Department of Computer Science and Security, Nankai University, China. His research interests include scheduling, load balancing, and other resource management issues in distributed systems and cloud computing.

Wentong Cai is currently a Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His expertise is mainly in the areas of modeling and simulation and parallel and distributed computing. He is an Associate Editor of the *ACM Transactions on Modeling and Computer Simulation* and an Editor of the *Future Generation Computer Systems*. He has chaired a number of international conferences. Most recent ones include the 2016 International ICST Conference on Simulation Tools and Techniques, the 2015 IEEE/ACM Symposium on Distributed Simulation and Real Time Applications, and the 2014 IEEE International Conference on Cloud Computing Technology and Science.