Clairvoyant Dynamic Bin Packing for Job Scheduling with Minimum Server Usage Time

Runtian Ren and Xueyan Tang
School of Computer Science and Engineering
Nanyang Technological University
Singapore 639798
{RENR0002, asxytang}@ntu.edu.sg

ABSTRACT

The MinUsageTime Dynamic Bin Packing (DBP) problem targets at minimizing the accumulated usage time of all the bins in the packing process. It models the server acquisition and job scheduling issues in many cloud-based systems. Earlier work has studied MinUsageTime DBP in the non-clairvoyant setting where the departure time of each item is not known at the time of its arrival. In this paper, we investigate MinUsageTime DBP in the clairvoyant setting where the departure time of each item is known for packing purposes. We study both the offline and online versions of Clairvoyant MinUsageTime DBP. We present two approximation algorithms for the offline problem, including a 5-approximation Duration Descending First Fit algorithm and a 4-approximation Dual Coloring algorithm. For the online problem, we establish a lower bound of $\frac{1+\sqrt{5}}{2}$ on the competitive ratio of any online packing algorithm. We propose two strategies of item classification for online packing, including a classify-by-departure-time strategy and a classify-by-duration strategy. We analyze the competitiveness of these strategies when they are applied to the classical First Fit packing algorithm. It is shown that both strategies can substantially reduce the competitive ratio for Clairvoyant MinUsageTime DBP compared to the original First Fit algorithm.

Keywords

Dynamic bin packing, job scheduling

1. INTRODUCTION

Job scheduling is a fundamental issue in many cloud-based systems. In a typical scenario, each cloud server has a fixed capacity of computational resources. Each job requires a certain amount of computational resources to run on a cloud server from its arrival time to its departure time. Since the total resource demand of the jobs running on a server cannot exceed its capacity, a limited number of jobs can be processed concurrently on each cloud server. Thus, a sufficient number of cloud servers need to be acquired to handle a given set of jobs. On the other hand, the cost of renting a cloud server is normally proportional to its running hours by "pay-as-you-go" billing [1]. Therefore, it is important to acquire cloud servers and schedule jobs on the cloud servers in an effective way to minimize the total renting cost of the servers. Such a server acquisition and job scheduling problem can be modeled as the MinUsageTime Dynamic Bin Packing (DBP) problem, which was first defined by Li *et al.* [19, 17]. The MinUsageTime DBP problem aims at minimizing the total usage time of all the bins in the packing process. The jobs and cloud servers in the aforementioned job scheduling problem correspond to the items and bins respectively in MinUsageTime DBP. Li *et al.* [19, 17, 24] studied MinUsageTime DBP in the non-clairvoyant setting where the departure time of each item is not known at the time of its arrival. They analyzed the competitiveness of several classical bin packing algorithms for online Non-Clairvoyant MinUsageTime DBP, including an 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).

In this paper, we study the MinUsageTime DBP problem in the clairvoyant setting where the departure time of each item is known for packing purposes. This is possible in real applications such as cloud gaming where the ending times of game sessions can be predicted with reasonable accuracy for certain games [18], and data analytics systems where jobs are mostly recurring [21, 12]. We consider both the offline and online versions of the Clairvoyant MinUsageTime DBP problem.[1] In the offline version, the information of all the items to pack (including item sizes, arrival times and departure times) is assumed known, whereas in the online version, the items must be placed into bins as they arrive without any knowledge of future item arrivals. Clairvoyant MinUsageTime DBP can also be viewed as a generalized interval scheduling problem. In the interval scheduling problem with bounded parallelism [10], a set of interval jobs are to be processed. Each job should be assigned to run on a machine from its arrival time to its departure time. Each machine can simultaneously process at most a fixed number of g jobs, where g stands for the machine capacity. A machine is considered busy when there is at least one job running on it. The interval scheduling problem aims at minimizing the total busy time of all the machines for processing the given set of interval jobs. Our Clairvoyant MinUsageTime DBP problem generalizes the above interval scheduling problem in that items are allowed to have arbitrary sizes, so the maximum number of items that can be placed in a bin is not fixed.

Previous work on MinUsageTime DBP. Li *et al.* [19, 17] analyzed the competitiveness of several classical bin packing algo-

SPAA '16, July 11-13, 2016, Pacific Grove, CA, USA
© 2016 ACM. ISBN 978-1-4503-4210-0/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2935764.2935775

[1]We remark that it does not really make sense to study the offline version of the Non-Clairvoyant MinUsageTime DBP problem since by definition, item departure times are not known for packing purposes in the non-clairvoyant setting.

rithms for Non-Clairvoyant MinUsageTime DBP. It was shown 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. The competitive ratio of First Fit packing has an upper bound of $2\mu + 7$. A Hybrid First Fit algorithm that classifies and packs items based on their sizes can achieve a competitive ratio within $\frac{8}{7}\mu + \frac{55}{7}$ when μ is not known and within $\mu + 5$ when μ is known. Kamali *et al.* [13] later showed that Next Fit packing has a competitive ratio bounded above by $2\mu + 1$. Recently, Tang *et al.* [24] conducted an improved competitive analysis for First Fit packing and established a new upper bound of $\mu + 4$ on its competitive ratio. All the above results are for the MinUsageTime DBP problem in the non-clairvoyant setting. To the best of our knowledge, there has been no study on the MinUsageTime DBP problem in the clairvoyant setting.

Contributions of this paper. In this paper, we investigate the Clairvoyant MinUsageTime DBP problem. We first study approximation algorithms for the offline Clairvoyant MinUsageTime DBP problem. We show that a Duration Descending First Fit algorithm can achieve an approximation ratio within 5. We develop a Dual Coloring algorithm and show that it has an approximation ratio bounded above by 4. We then study the online Clairvoyant MinUsageTime DBP problem. We establish a lower bound of $\frac{1+\sqrt{5}}{2}$ on the competitive ratio of any online packing algorithm. We propose two strategies of item classification for online Clairvoyant MinUsageTime DBP, including a classify-by-departure-time strategy and a classify-by-duration strategy. We apply these strategies to First Fit packing (which is the best known online algorithm in the non-clairvoyant setting) and analyze their competitiveness. If the minimum and maximum item durations are not known, classify-by-departure-time First Fit can achieve a competitive ratio within $\frac{\rho}{\Delta} + \frac{\mu\Delta}{\rho} + 3$, where ρ is an algorithm parameter of classification and Δ is the minimum item duration, and classify-by-duration First Fit can achieve a competitive ratio within $\alpha + \lceil log_{\alpha}\mu \rceil + 4$, where α is the max/min item duration ratio of each item category. If the minimum and maximum item durations are known, classify-by-departure-time First Fit can achieve a competitive ratio within $2\sqrt{\mu} + 3$, and classify-by-duration First Fit can achieve a competitive ratio within $\min_{n \geq 1} \mu^{\frac{1}{n}} + n + 3$. Numerical results show that both classification strategies can substantially reduce the competitive ratio for Clairvoyant MinUsageTime DBP compared to the original First Fit packing algorithm.

2. RELATED WORK

The classical DBP problem, which was first defined by Coffman *et al.* [9], aims at minimizing the maximum number of bins concurrently used in the packing process. A large amount of research work has been carried out to analyze the competitiveness of various algorithms for classical DBP [9, 6, 7]. However, classical DBP does not consider the duration of bin usage. In contrast, the MinUsageTime DBP problem defined by Li *et al.* [19, 17] aims to minimize the accumulated bin usage time.

As mentioned, interval scheduling with bounded parallelism [10, 20, 23, 14, 8] is also related to our MinUsageTime DBP problem. Flammini *et al.* [10] presented a 4-approximation algorithm for interval scheduling with bounded parallelism. They also developed a greedy algorithm with an approximation ratio of 2 for the special case where no interval is properly contained in another. Mertzios *et al.* [20] studied two versions of interval scheduling with bounded parallelism: one scheduling minimization version for minimizing the total busy time of machines to process all jobs and one resource allocation maximization version for maximizing the number of jobs processed under a busy time budget. For the scheduling minimization problem, they proposed an improved $(2 - \frac{1}{g})$-approximation algorithm for the special case where no interval is properly contained in another. They also proposed a $\frac{g \cdot H_g}{H_g + g - 1}$-approximation algorithm (H_g is the g-th harmonic number) for the special case where any two job intervals intersect. Chang *et al.* [8] outlined two 2-approximation algorithms for the problem of interval scheduling with bounded parallelism based on the work of Alicherry *et al.* [4] and Kumar *et al.* [15]. They also proposed a GREEDYTRACKING algorithm which has an approximation ratio of 3. The aforementioned work all focused on the offline version of interval scheduling. Shalom *et al.* [23] investigated the online version of interval scheduling with bounded parallelism, in which interval jobs are scheduled to run on machines when they arrive without any knowledge of future job arrivals. They proposed a BucketFirstFit algorithm and showed that it has a competitive ratio bounded by $(2\alpha + 2) \cdot \lceil \log_{\alpha} \mu \rceil$, where μ is the max/min length ratio of job intervals and α is an algorithm parameter. All the above work on interval scheduling assumes that the jobs all have the same resource demands which are equal to $\frac{1}{g}$ of the machine capacity for a given integer g. In contrast, our MinUsageTime DBP problem considers items (jobs) with arbitrary sizes (demands). As shall be discussed later, some of our analysis implies significant improvement to the existing result of online interval scheduling. Khandekar *et al.* [14] considered scheduling flexible jobs whose intervals may not be fixed. Each job is associated with a release time, a deadline, a processing time (length), and a demand for machine capacity. A First_Fit_with_Demands algorithm was proposed to divide jobs into narrow and wide ones based on their demands and the algorithm was shown to achieve an approximation ratio of 5. Our Clairvoyant MinUsageTime DBP problem is similar to this one, but we focus on interval jobs whose starting and ending times are fixed. We propose a 5-approximation algorithm different from [14] (without dividing jobs according to their demands) and a new 4-approximation algorithm for Clairvoyant MinUsageTime DBP. Moreover, besides packing items in an offline manner, we also study online Clairvoyant MinUsageTime DBP.

3. PRELIMINARIES

3.1 Notations and Definitions

We first define some key notations for 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 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 interval $I(r)$, and refer to $I(r)$ as the *active interval* of r. The length of $I(r)$ is known as the *item duration*. Let $s(r)$ denote the size of item r. The product of the size and duration of item r, i.e., $s(r) \cdot l(I(r))$ represents its *time-space demand*. For a list of items $\mathcal{R}$, we use $d(\mathcal{R})$ to denote the total time-space demand of all the items in $\mathcal{R}$, i.e., $d(\mathcal{R}) = \sum_{r \in \mathcal{R}} s(r) \cdot l(I(r))$. 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). 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 $\mathcal{R}$.

We assume that the items are not allowed to move from one bin to another during their active intervals. For example, in highly interactive applications such as cloud gaming [11, 3, 2, 16], it is not

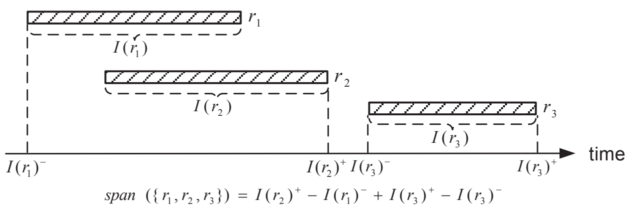

Figure 1: Span of an item list

preferable to migrate game instances from one server to another during execution as migration causes interruption to game play. At any time in the packing process, the total size of all the active items placed in a bin, known as the *bin level*, cannot exceed the bin capacity. The usage time of a bin is defined as the span of all the items placed in it. The objective of MinUsageTime DBP is to minimize the total usage time of all the bins.

3.2 Approximation and Competitive Ratios

The performance of an approximation algorithm and an online algorithm is often characterized by their *approximation ratio* and *competitive ratio* respectively, i.e., the worst-case ratio between the objective function value of a solution constructed by the algorithm and the objective function value of an optimal solution [5, 25].

We give three lower bounds on the total bin usage time of an optimal packing solution to the MinUsageTime DBP problem. These bounds shall be used in deriving the approximation ratios or competitive ratios of our algorithms. 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 active items 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

$$OPT_{total}(\mathcal{R}) = \int_{\bigcup_{r\in\mathcal{R}} I(r)} OPT(\mathcal{R}, t)\, \mathrm{d}t,$$

where $\bigcup_{r\in\mathcal{R}} I(r)$ is the time period corresponding to the span of $\mathcal{R}$. It is easy to obtain the following lower bounds on $OPT_{total}(\mathcal{R})$:

PROPOSITION 1. *$OPT_{total}(\mathcal{R}) \geq d(\mathcal{R})$.*

PROPOSITION 2. *$OPT_{total}(\mathcal{R}) \geq span(\mathcal{R})$.*

PROPOSITION 3. *$OPT_{total}(\mathcal{R}) \geq \int_{\bigcup_{r\in\mathcal{R}} I(r)} \lceil S(t) \rceil\, \mathrm{d}t$, where $S(t)$ is the total size of all the active items at time t.*

The first bound is derived by assuming that no capacity of any bin is wasted at any time, where $d(\mathcal{R})$ is the total time-space demand of all the items in $\mathcal{R}$. 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. The above two bounds have been given in the earlier work [19, 17]. The third bound is a new one motivated by [8]. It exploits the fact that the number of open bins must be at least $\lceil S(t) \rceil$ at any time t. Obviously, the third bound is tighter than the first two bounds.

4. APPROXIMATION ALGORITHMS FOR CLAIRVOYANT MINUSAGETIME DBP

We start by studying the offline problem of Clairvoyant MinUsageTime DBP. Note that when all the items have the same arrival times and the same departure times, Clairvoyant MinUsageTime DBP reduces to the classical bin packing problem since minimizing the total bin usage time is equivalent to minimizing the number of bins opened. Thus, it is obvious that Clairvoyant MinUsageTime DBP is NP-hard. In the following, we present and analyze two approximation algorithms for Clairvoyant MinUsageTime DBP. Without loss of generality, we shall assume that the minimum item duration is 1, and the maximum item duration is μ ($\mu \geq 1$).

4.1 Duration Descending First Fit Algorithm

Our first algorithm is called *Duration Descending First Fit* and it works as follows. First, all the items are sorted in the descending order of their durations. Then, the items are placed one at a time according to the first fit rule. That is, each item is placed in the bin of the smallest index that can accommodate the item throughout its duration, where the bins are indexed according to their opening order by the execution of the packing algorithm. If no opened bin can accommodate the item throughout its duration, a new bin is opened for it. This algorithm was also used by [10] for interval scheduling with bounded parallelism and shown to have an approximation ratio of 4 when all jobs (items) have the same resource demands (sizes). In such a special case, the maximum number of jobs that can be concurrently processed by each machine (i.e., the machine capacity) is fixed. If the machines have capacity g, each machine can essentially be viewed as a collection of g mini-machines each with capacity 1. These g mini-machines are then considered separately in the analysis of [10]. In contrast, in our MinUsageTime DBP problem, items can have arbitrary sizes so that the maximum number of items that can be placed in each bin is not predictable and it may also vary at different times. Thus, the analysis of [10] is not directly applicable to MinUsageTime DBP since it is impossible to divide each bin into mini-bins. Inspired by [10], in what follows, we present a new analysis to show that Duration Descending First Fit has an approximation ratio of 5 for MinUsageTime DBP. Our key observation is that whenever a new bin is opened to pack an item r, there must exist at least one moment in r's duration such that the level of an earlier opened bin plus the size of r is larger than the bin capacity.

Suppose that a total of m bins $b_1, b_2, \ldots, b_m$ are used by the Duration Descending First Fit algorithm to pack a list of items $\mathcal{R}$. The bins are indexed in their opening order, i.e., b_1 is the first bin opened by the Duration Descending First Fit algorithm and b_m is the last bin opened by the algorithm. For each bin b_k, let $\mathcal{R}_k$ denote the set of the items packed into b_k. Then, by definition, the usage time of b_k is $span(\mathcal{R}_k)$. Thus, the total bin usage time of Duration Descending First Fit is given by $\sum_{k=1}^{m} span(\mathcal{R}_k)$.

Consider any item set $\mathcal{R}_k$ ($k \geq 2$). We reduce $\mathcal{R}_k$ to a subset $\mathcal{R}'_k$ by removing any item whose duration is completely contained in the duration of another item. In this way, the items left in $\mathcal{R}'_k$ satisfy the following property: let n denote the number of items in $\mathcal{R}'_k$ and let $r_1, r_2, \ldots, r_n$ be the items sorted according to their arrival times, i.e., $I(r_1)^- < I(r_2)^- < \cdots < I(r_n)^-$, then it holds that $I(r_1)^+ < I(r_2)^+ < \cdots < I(r_n)^+$. Otherwise, if $I(r_x)^+ \geq I(r_y)^+$ for some $x < y$, the duration of item r_y is fully contained in the duration of item r_x, which contradicts the definition of $\mathcal{R}'_k$. It is easy to see that reducing $\mathcal{R}_k$ to $\mathcal{R}'_k$ does not shorten the usage time of bin b_k, i.e., $span(\mathcal{R}_k) = span(\mathcal{R}'_k)$.

As shown in Figure 2, we can split $\bigcup_{r\in\mathcal{R}'_k} I(r)$ into n disjoint periods at the arrival times of the items:

$$\begin{aligned} X(r_1) &= [I(r_1)^-, \min\{I(r_2)^-, I(r_1)^+\}), \\ X(r_2) &= [I(r_2)^-, \min\{I(r_3)^-, I(r_2)^+\}), \\ &\ldots\ldots \end{aligned}$$

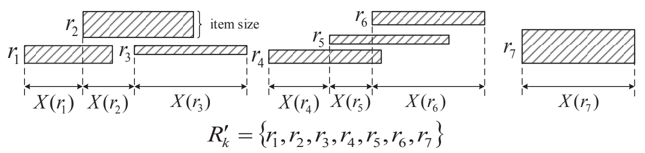

Figure 2: Splitting $\bigcup_{r\in\mathcal{R}'_k} I(r)$ for bin b_k

$$\begin{aligned} X(r_{n-1}) &= [I(r_{n-1})^-, \min\{I(r_n)^-, I(r_{n-1})^+\}), \\ X(r_n) &= [I(r_n)^-, I(r_n)^+). \end{aligned}$$

We refer to the above periods as the X-periods of items r_1, r_2, $\cdots$, r_n respectively. Obviously, the total length of the X-periods is $span(\mathcal{R}'_k)$, i.e., $\sum_{i=1}^n l\big(X(r_i)\big) = span(\mathcal{R}'_k)$. We define

$$d_k = \sum_{i=1}^{n} s(r_i)\cdot l\big(X(r_i)\big).$$

Apparently, 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 the active interval of the item:

$$\begin{aligned} d_k &\le \sum_{i=1}^{n} s(r_i)\cdot l\big(I(r_i)\big) \\ &= d(\mathcal{R}'_k) \\ &\le d(\mathcal{R}_k). \end{aligned} \tag{1}$$

By the definition of Duration Descending First Fit, when each item r_i in $\mathcal{R}'_k$ ($k \ge 2$) is placed in bin b_k, there must exist at least one moment t_i, such that the level of bin b_{k-1} at time t_i plus the size of item r_i is larger than 1 (the bin capacity). Let $W(r_i)$ be the set of all the active items in bin b_{k-1} at time t_i when item r_i is placed in bin b_k by the Duration Descending First Fit algorithm. By the algorithm definition, each item in $W(r_i)$ must have a duration no shorter than that of r_i. We define

$$d_k^* = \sum_{i=1}^{n}\Big(\sum_{r\in W(r_i)} s(r)\cdot l\big(X(r_i)\big)\Big).$$

It then follows that

$$\begin{aligned} d_k + d_k^* &= \sum_{i=1}^{n}\Big(s(r_i) + \sum_{r\in W(r_i)} s(r)\Big)\cdot l\big(X(r_i)\big) \\ &> \sum_{i=1}^{n} 1\cdot l\big(X(r_i)\big) \\ &= span(\mathcal{R}'_k) \\ &= span(\mathcal{R}_k). \end{aligned} \tag{2}$$

We next show that d_k^* is bounded by $3\cdot d(\mathcal{R}_{k-1})$, where $d(\mathcal{R}_{k-1})$ is the total time-space demand of all the items placed in bin b_{k-1}.

LEMMA 1. $d_k^* \le 3\cdot d(\mathcal{R}_{k-1})$.

Proof: Please refer to the extended version of this paper [22]. □

It follows from (1), (2) and Lemma 1 that

$$span(\mathcal{R}_k) < d(\mathcal{R}_k) + 3\cdot d(\mathcal{R}_{k-1}).$$

Note that the above inequality holds for any item set $\mathcal{R}_k$ where $k \ge 2$. As a result, we have

$$\begin{aligned} \sum_{k=2}^{m} span(\mathcal{R}_k) &< \sum_{k=2}^{m}\big(d(\mathcal{R}_k) + 3\cdot d(\mathcal{R}_{k-1})\big) \\ &< \sum_{k=1}^{m} d(\mathcal{R}_k) + 3\cdot\sum_{k=1}^{m} d(\mathcal{R}_k) \\ &= d(\mathcal{R}) + 3\cdot d(\mathcal{R}) \\ &= 4\cdot d(\mathcal{R}), \end{aligned}$$

where $d(\mathcal{R})$ is the total time-space demand of all the items.

Therefore, the total bin usage time of Duration Descending First Fit satisfies

$$\begin{aligned} \sum_{k=1}^{m} span(\mathcal{R}_k) &= \Big(\sum_{k=2}^{m} span(\mathcal{R}_k)\Big) + span(\mathcal{R}_1) \\ &< 4\cdot d(\mathcal{R}) + span(\mathcal{R}) \\ &\le 4\cdot OPT_{total}(\mathcal{R}) + OPT_{total}(\mathcal{R}) \\ &= 5\cdot OPT_{total}(\mathcal{R}), \end{aligned}$$

where the last inequality follows from the bounds given in Propositions 1 and 2. Thus, we have the following conclusion.

THEOREM 1. *The Duration Descending First Fit algorithm is a 5-approximation algorithm for Clairvoyant MinUsageTime DBP.*

4.2 Dual Coloring Algorithm

In this section, we develop a *Dual Coloring* algorithm for Clairvoyant MinUsageTime DBP and show that it has an approximation ratio of 4. Our algorithm is motivated by the Simple algorithm described in [15] for the Fiber Minimization problem which aims to construct a cost-effective optical fiber network to meet demands. A recent study [8] indicated that the Simple algorithm can also be used for interval scheduling with bounded parallelism and has an approximation ratio of 2 for it. In the context of interval scheduling, this algorithm works by considering a machine that can process g jobs concurrently as a collection of g mini-machines each with capacity 1. It allocates jobs and schedules them on one mini-machine at a time. However, this approach is not guaranteed to produce a feasible solution for our MinUsageTime DBP problem as the number of items that can be simultaneously placed in each bin is not fixed and should be dynamically determined as packing progresses. The arbitrary sizes of items make it challenging to distribute items among bins to produce a quality solution.

Our Dual Coloring algorithm divides the items in a list $\mathcal{R}$ into two groups: a small group $\mathcal{R}_S$ and a large group $\mathcal{R}_L$. $\mathcal{R}_S$ contains all the items of size no larger than $\frac{1}{2}$, and $\mathcal{R}_L$ contains all the items of size larger than $\frac{1}{2}$. These two groups of items are packed separately. The large items are packed arbitrarily and the bins used to pack large items do not further pack any small item. The small items are packed by first constructing a demand chart. As shown in Figure 3, the horizontal dimension of the demand chart represents the time, which ranges over the span of the item list. The vertical dimension of the demand chart represents the space demand. The height of the demand chart at any time t is given by the total size of all the small items that are active at time t. With the demand chart, the small items are then packed in two phases. In Phase 1, all the small items are placed in the demand chart such that no three items overlap together. In Phase 2, the small items are packed into bins according to their positions in the demand chart.

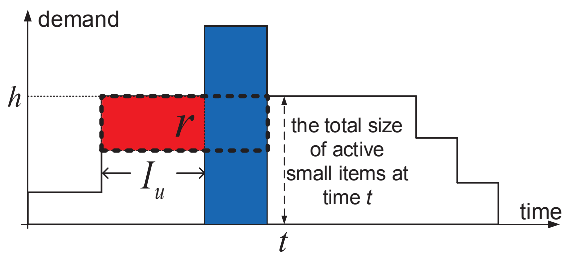

Figure 3: Item placement in demand chart

Phase 1 Item placement in demand chart

```
1: $M \leftarrow \{$the total size of all active small items at time $t \mid t \in \cup_{r \in \mathcal{R}_S} I(r)\}$ and $\mathcal{J} \leftarrow \mathcal{R}_S$
2: while $M \neq \emptyset$ do
3:    Find the maximum altitude $h \in M$ to examine
4:    Let $R$, $B$ and $U$ be the respective sets of all maximal red, blue and uncolored intervals at altitude $h$
5:    while $U \neq \emptyset$ do
6:       Pick an uncolored interval $I_u \in U$
7:       if $\exists\ r \in \mathcal{J}$ such that $I(r) \cap I_u \neq \emptyset$ and $\forall\ I \in U \cup R \backslash \{I_u\}, I(r) \cap I = \emptyset$ then
8:          Place $r$ at altitude $h$ and delete $r$ from $\mathcal{J}$
9:          Color the rectangle $\{I(r) \cap I_u\} \times (h - s(r), h]$ red
10:         Delete $I_u$ from $U$
11:         if $I_u^- < I(r)^-$ then
12:            Add $[I_u^-, I(r)^-)$ into $U$
13:         if $I_u^+ > I(r)^+$ then
14:            Add $[I(r)^+, I_u^+)$ into $U$
15:         if $h > s(r)$ and $\big(h - s(r)\big) \notin M$ then
16:            Add $\big(h - s(r)\big)$ into $M$
17:      else
18:         Color the rectangle $I_u \times (0, h]$ blue
19:         Delete $I_u$ from $U$
20:   Delete $h$ from $M$
```

In Phase 1, each item r placed in the demand chart occupies a rectangle covering its active interval $I(r)$ in the time dimension and of height $s(r)$ in the demand dimension. Thus, the area occupied by each item r is equal to its time-space demand $s(r) \cdot l(I(r))$. To simplify presentation, we say that an item r is placed at altitude h in the demand chart if it occupies the range $(h - s(r), h]$ in the demand dimension. The area of the demand chart is gradually colored either red or blue as items are placed. To place items, the Dual Coloring algorithm examines a collection of altitudes from high to low in the demand chart. Let M denote the collection of altitudes to examine. Initially, M contains all possible total sizes of active small items at different times (step 1), i.e., all the ceiling altitudes in the demand chart. In each iteration, the algorithm examines the next (maximum) altitude h in M (step 3). Based on its coloring status, the horizontal line at altitude h in the demand chart is divided into three sets of intervals: red, blue and uncolored (step 4). For each uncolored interval I_u, the algorithm looks for an item r (not yet placed) whose active interval $I(r)$ intersects with I_u but does not intersect with any other uncolored interval and any red interval (step 7). If there is no such item, all the area below I_u in the demand chart is colored blue (step 18). If such an item r exists, r is placed at altitude h (step 8) (see Figure 3 for an example). The uncolored area covered by r is then colored red (step 9). For convenience of algorithm presentation, we intentionally leave r's lower boundary uncolored for identifying uncolored intervals in the subsequent examination. The altitude of r's lower boundary, i.e., $h - s(r)$, is added into M (step 16) as other items may be able to be placed immediately below r at altitude $h - s(r)$. The examination of an altitude h proceeds until there is no more uncolored interval (step 5). Phase 1 completes when all altitudes in M have been examined (step 2).

We now prove that Phase 1 runs in polynomial time. Note that the total size of active small items can change only upon item arrivals and departures. Thus, the initial set M (the set of all possible total sizes of active small items over time) contains at most $2 \cdot |\mathcal{R}_S|$ different altitudes, where $|\mathcal{R}_S|$ is the number of small items (step 1). Subsequently, the placement of each item introduces at most one additional altitude to examine (step 16). Therefore, the total number of altitudes examined in Phase 1 is bounded by $3 \cdot |\mathcal{R}_S|$. In the examination of each altitude, the total number of red, blue and uncolored intervals is bounded by $O(|\mathcal{R}_S|)$ because the endpoints of each interval must correspond to item arrivals and/or departures. There are at most $|\mathcal{R}_S|$ items to check for each uncolored interval, and checking the eligibility of each item for placement takes $O(|\mathcal{R}_S|)$ time. So, the time complexity for examining each altitude is bounded by $O(|\mathcal{R}_S|^3)$. As a result, the total time complexity of Phase 1 is bounded by $O(|\mathcal{R}_S|^4)$.

Phase 1 gives rise to the following properties of item placement in the demand chart. The complete proofs of these properties are given in the extended version of this paper [22].

LEMMA 2. *After Phase 1, the entire area of the demand chart is colored.*

Proof Sketch: This claim is proved by showing that when the examination of an altitude h completes, all the area between altitude h and the next altitude h^- to examine $(h^- < h)$ is colored. □

LEMMA 3. *Whenever an item r is placed at an altitude h in Phase 1, the rectangle occupied by r is within the demand chart.*

Proof Sketch: It is obvious that r's left boundary (at time $I(r)^-$), right boundary (at time $I(r)^+$) and upper boundary (at altitude h) are within the demand chart. Thus, we only need to check r's lower boundary (at altitude $h - s(r)$). It can be shown that r's lower boundary must also be within the demand chart. □

LEMMA 4. *All the small items are placed in the demand chart after Phase 1.*

LEMMA 5. *No three items overlap together in their placement of Phase 1.*

In Phase 2, to pack items into bins, the demand chart is partitioned into stripes of height $\frac{1}{2}$ each as shown in Figure 4. Let $S_S(t)$ denote the total size of all active small items at time t. Then, the total number of stripes is given by $m = \lceil 2 \cdot \max_{t \in \cup_{r \in \mathcal{R}_S} I(r)} S_S(t) \rceil$ (step 1), where $\max_{t \in \cup_{r \in \mathcal{R}_S} I(r)} S_S(t)$ is the maximum height of the demand chart. The algorithm creates $(2m - 1)$ bins to pack items, among which m bins are used for items placed completely within individual stripes in Phase 1 and the other $(m - 1)$ bins are used for items placed across stripes. Specifically, for each stripe, since the height of the stripe is $\frac{1}{2}$ and no three items overlap together in their placement (Lemma 5), the total size of all the items placed completely within the stripe is bounded by 1 at any time. Thus, they can be accommodated by one bin (step 6). On the other hand, note that the size of each small item is bounded by $\frac{1}{2}$. This implies that an item can cross at most two stripes in the placement by Phase 1. For each pair of neighboring stripes, by Lemma 5, at

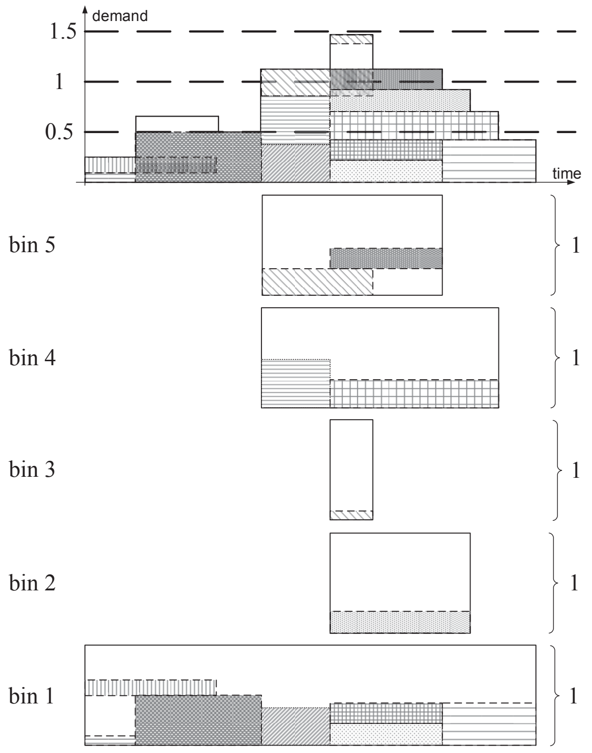

Figure 4: Item packing

```
Phase 2 Item packing
1: $m \leftarrow \lceil 2 \cdot \max_{t \in \cup_{r \in \mathcal{R}_S} I(r)} S_S(t) \rceil$
2: Create $2m - 1$ bins
3: for each item $r$ do
4:     Let $h$ be the altitude where $r$ is placed in the demand chart
5:     if $\frac{k-1}{2} \leq h - s(r) < h \leq \frac{k}{2}$ for some $k$ $(1 \leq k \leq m)$ then
6:         Pack item $r$ into the $k$-th bin
7:     if $h - s(r) < \frac{k}{2} < h$ for some $k$ $(1 \leq k < m)$ then
8:         Pack item $r$ into the $(m + k)$-th bin
```

most two items can cross them concurrently at any time, so their total size is bounded by 1. Therefore, all the items placed across a pair of neighboring stripes can be accommodated by one bin (step 8). Since there are $(m-1)$ pairs of neighboring stripes, we need at most $(m-1)$ bins to pack all the items placed across stripes. It is easy to see that the time complexity of Phase 2 is $O(|\mathcal{R}_S|)$, where $|\mathcal{R}_S|$ is the number of small items.

Now, let's compute the bin usage time of the Dual Coloring algorithm. In Phase 2 of packing small items, the number of stripes produced at each time point t is $\lceil 2S_S(t) \rceil$. Thus, the number of open bins for packing small items at time t is at most $2\lceil 2S_S(t) \rceil - 1$, among which $\lceil 2S_S(t) \rceil$ bins are used for items placed within individual stripes and $\lceil 2S_S(t) \rceil - 1$ bins are used for items placed across stripes. So, the total bin usage time for packing small items is bounded by $\int_{\cup_{r \in \mathcal{R}_S} I(r)} (2\lceil 2S_S(t) \rceil - 1)\, \mathrm{d}t$.

Recall that large items are packed arbitrarily and the size of each large item is greater than $\frac{1}{2}$. Let $S_L(t)$ denote the total size of all active large items at time t. Then, the number of active large items at time t is smaller than $2 \cdot S_L(t)$. As a result, the number of open bins for packing large items at time t is at most $\lfloor 2S_L(t) \rfloor$. So, the total bin usage time for packing large items is bounded by $\int_{\cup_{r \in \mathcal{R}_L} I(r)} \lfloor 2S_L(t) \rfloor\, \mathrm{d}t$.

Therefore, the total bin usage time of the Dual Coloring algorithm is bounded by

$$\int_{\cup_{r \in \mathcal{R}_L} I(r)} \lfloor 2S_L(t) \rfloor\, \mathrm{d}t + \int_{\cup_{r \in \mathcal{R}_S} I(r)} (2\lceil 2S_S(t) \rceil - 1)\, \mathrm{d}t.$$

THEOREM 2. *The Dual Coloring algorithm is a 4-approximation algorithm for Clairvoyant MinUsageTime DBP.*

Proof Sketch: Let $S(t)$ denote the total size of all active items at time t. Following algebra, it can be shown that at any time t, the total number of open bins is bounded by $4\lceil S(t) \rceil$. Therefore,

$$\begin{aligned}&\int_{\cup_{r \in \mathcal{R}_L} I(r)} \lfloor 2S_L(t) \rfloor\, \mathrm{d}t + \int_{\cup_{r \in \mathcal{R}_S} I(r)} (2\lceil 2S_S(t) \rceil - 1)\, \mathrm{d}t \\ &\leq \int_{\cup_{r \in \mathcal{R}} I(r)} 4\lceil S(t) \rceil\, \mathrm{d}t \\ &\leq 4 \cdot OPT_{total}(\mathcal{R}),\end{aligned}$$

where the last inequality follows from the bound given in Proposition 3. Please refer to the extended version of this paper [22] for the complete proof. □

5. ONLINE ALGORITHMS FOR CLAIRVOYANT MINUSAGETIME DBP

Earlier work has established a lower bound μ on the competitive ratio of any online packing algorithm for Non-Clairvoyant MinUsageTime DBP where the departure times of items are not known when they are placed in bins [13, 17]. In Clairvoyant MinUsageTime DBP, with additional information of item departure times, it is possible to design more competitive online packing algorithms. In this section, we first present a lower bound on the competitive ratio of any online packing algorithm for Clairvoyant MinUsageTime DBP. Then, we propose two strategies of item classification to improve the competitiveness of online packing for Clairvoyant MinUsageTime DBP. The first strategy is called *classify-by-departure-time* and the second strategy is called *classify-by-duration*. We apply these two strategies to First Fit packing (the best known online algorithm in the non-clairvoyant setting) and analyze their competitiveness. To facilitate presentation, we say that a bin is *opened* in an online packing process when it receives the first item. When all the items in a bin depart, the bin is *closed*.

5.1 A lower bound of online packing algorithms

THEOREM 3. *For Clairvoyant MinUsageTime DBP, no deterministic online packing algorithm can achieve a competitive ratio lower than* $\frac{1+\sqrt{5}}{2}$.

Proof: At time 0, let two items of size $\frac{1}{2} - \varepsilon$ arrive, where $0 < \varepsilon < \frac{1}{2}$ is a small value. The duration of the first item is x and the duration of the second item is 1 $(x > 1)$. We consider two cases as shown in Figure 5. In case A, the item list to pack contains the above two items only. In case B, another two items of size $\frac{1}{2} + \varepsilon$ arrive at time τ ($\tau > 0$ is a small value), where the duration of the third item is x and the duration of the fourth item is 1. Obviously, the optimal solution for case A is to place the first two items in the same bin, which gives a total bin usage time of x. The optimal solution for case B is to place the first and third items in one bin

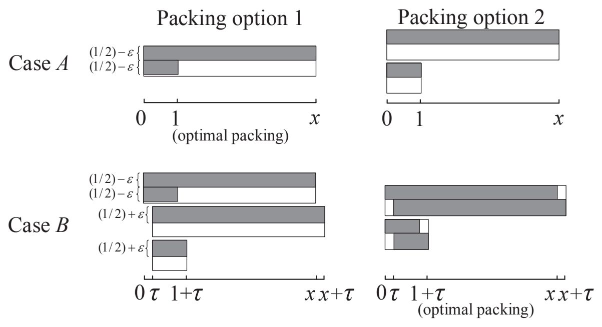

Figure 5: Two cases for online packing

and the second and fourth items in another bin, which gives a total bin usage time of $x + \tau + 1 + \tau = x + 1 + 2\tau$.

Note that any deterministic online packing algorithm always packs the first two items in the same way for both cases A and B, since the knowledge of the third and fourth items is not available when packing the first two items in case B. As shown in Figure 5, if an algorithm places the first two items in the same bin, this bin will not be able to accommodate any later item. Since the next two items in case B have size $\frac{1}{2} + \varepsilon$ each, two additional bins are needed to pack them separately. Thus, the total bin usage time for case B is $x + x + 1 = 2x + 1$, resulting in a ratio of $\frac{2x+1}{x+1+2\tau}$ to the optimal solution, which approaches $\frac{2x+1}{x+1}$ as τ approaches 0. On the other hand, if an algorithm places the first two items in two bins separately, then the resultant ratio to the optimal solution for case A is $\frac{x+1}{x}$. Therefore, the competitive ratio of any deterministic online packing algorithm must be at least $\min\{\frac{x+1}{x}, \frac{2x+1}{x+1}\}$. When $x = \frac{1+\sqrt{5}}{2}$, the value of $\min\{\frac{x+1}{x}, \frac{2x+1}{x+1}\}$ reaches the maximum at $\frac{x+1}{x} = \frac{2x+1}{x+1} = \frac{1+\sqrt{5}}{2}$. Hence, the theorem is proven. □

5.2 Classify-by-Departure-Time Strategy

When the departure times of the items are not known at the times of their packing, it is likely that items with very different departure times are packed into the same bin. As a result, after the items with early departure times leave, the remaining items may cause the bin to keep open at some low bin level for a long time, preventing the bin from being closed timely. This can introduce significant but unnecessary bin usage time. To reduce the bin usage time, we propose a classify-by-departure-time strategy to pack the items based on their departure times: the items with similar departure times are considered for placement in the same bin. In this way, the items in a bin would depart at around the same time, after which the bin can be closed timely. As such, the total bin usage time can be saved.

In our classify-by-departure-time strategy, the time is split into intervals of length ρ each, where $\rho > 0$ is a constant. We classify all the items into categories according to their departure times. Each category contains all the items that depart in a time interval of length ρ. Without loss of generality, in our analysis, we assume that the times when there is at least one active item constitute a continuous interval. If they form multiple disjoint intervals, then the entire item list can be broken into multiple sublists with disjoint spans and our analysis can be applied to each sublist separately.

Given an item list $\mathcal{R}$, the above classification produces $\lceil \frac{span(\mathcal{R})}{\rho} \rceil$ categories of items. Without loss of generality, suppose that the first item of $\mathcal{R}$ arrives at time 0, and the last item of $\mathcal{R}$ departs at time $span(\mathcal{R})$. Then, the first category consists of the items departing

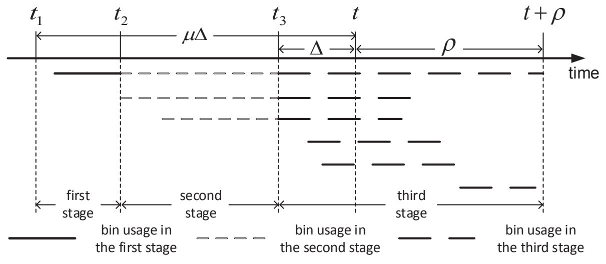

Figure 6: Time-space demands in three stages

in the time interval $(0, \rho]$, the second category consists of the items departing in the time interval $(\rho, 2\rho]$, and so on. The category of each item can be determined at the time of its arrival since the departure time of the item is known. We apply the First Fit algorithm to pack the items in each category separately. In First Fit, when a new item arrives, if there are one or more open bins that can accommodate it, the new item is placed in the bin which was opened earliest among these bins. Otherwise, if no open bin can accommodate the new item, a new bin is opened to receive the item.

Next, we study the competitive ratio of the classify-by-departure-time First Fit algorithm. Let Δ denote the minimum item duration and $\mu\Delta$ ($\mu \geq 1$) denote the maximum item duration. Consider a category in which the items depart in a time interval $(t, t + \rho]$. To analyze the bin usage time of packing all the items in the category, we divide the time-space demand of the items into three parts. As shown in Figure 6, We first define t_1 as the time point that is $\mu\Delta$ time units before the beginning of the departure time interval, i.e., $t_1 = t - \mu\Delta$. Since the maximum item duration is $\mu\Delta$, all the items in the category must arrive from t_1 onwards. We then define t_3 as the time point that is Δ time units before the beginning of the departure time interval, i.e., $t_3 = t - \Delta$. We further define t_2 as the time point between t_1 and t_3 when the second bin is opened to pack the items in the category. If no second bin is opened by time t_3, i.e., there is at most one open bin up to time t_3, we define $t_2 = t_3$. The time-space demand of the items in the category is then broken into those over the following three stages: the first stage is from time t_1 to t_2; the second stage is from time t_2 to t_3; and the third stage is from time t_3 to $t + \rho$.

Now, we analyze the bin usage times and the time-space demands in these three stages separately. In the first stage, by definition, there is at most one open bin. Since the length of the first stage is $t_2 - t_1 \leq t_3 - t_1 = (t - \Delta) - (t - \mu\Delta) = (\mu - 1)\Delta$, the bin usage time in the first stage of an item category is bounded by $(\mu - 1)\Delta$. Recall that there are $\lceil \frac{span(\mathcal{R})}{\rho} \rceil$ item categories. For the first item category, i.e., the items departing in the interval $(0, \rho]$, there is no bin usage in its first stage since it is beyond the span of the item list $\mathcal{R}$. Thus, the total bin usage time in the first stages of all the item categories (denoted by $usage_A$) is bounded by

$$\begin{aligned} usage_A &\leq (\mu - 1)\Delta \cdot \left(\left\lceil \frac{span(\mathcal{R})}{\rho} \right\rceil - 1 \right) \\ &< (\mu - 1)\Delta \cdot \frac{span(\mathcal{R})}{\rho} \\ &= \frac{(\mu - 1)\Delta}{\rho} \cdot span(\mathcal{R}). \end{aligned} \quad (3)$$

In the second stage, at least two bins are opened. The following lemma establishes a lower bound on the average bin level in the second stage.

LEMMA 6. *At any moment in the second stage, the average level of all the open bins must exceed $\frac{1}{2}$.*

Proof: Please refer to the extended version of this paper [22]. $\Box$

Lemma 6 implies that at any moment in the second stage, the total size of all the active items is at least half the total capacity of all open bins. Thus, the bin usage time in the second stage is bounded by twice the time-space demand of the items over the second stage. This holds for all the item categories. Let $usage_B$ denote the total bin usage time in the second stages of all the item categories, and let d_B denote the total time-space demand of the items over these stages. Then, we have

$$usage_B < 2 \cdot d_B. \tag{4}$$

Finally, we analyze the third stage. Suppose that a total of m bins $b_1, b_2, \ldots, b_m$ are used by the First Fit algorithm to pack the items of a category in the third stage. Assume that the bins are indexed in the temporal order of their openings, i.e., b_1 is the first opened bin and b_m is the last opened bin. For each bin b_i, if b_i is opened before time t_3, let I_i denote the period of b_i's usage from t_3 to its closing. If b_i is opened after time t_3, let I_i denote the entire usage period of b_i from its opening to closing. Then, the bin usage time in the third stage is given by $\sum_{i=1}^{m} l(I_i)$.

For each bin b_i, let E_i be the latest closing time of all the bins that are opened before b_i, i.e., $E_i = \max\{I_j^+ | 1 \le j < i\}$. We divide period I_i into two parts: I_i^L and I_i^R, where I_i^L is the period $[I_i^-, \min\{I_i^+, E_i\})$ (if $E_i \le I_i^-$, then $I_i^L = \emptyset$) and $I_i^R = I_i - I_i^L = [\min\{I_i^+, E_i\}, I_i^+)$ is the remaining period. For the first opened bin b_1, we define $E_1 = I_1^-$ so that $I_1^L = \emptyset$ and $I_1^R = I_1$. Apparently, we have $l(I_i) = l(I_i^L) + l(I_i^R)$ for each I_i. Therefore, the bin usage time in the third stage can be rewritten as $\sum_{i=1}^{m} l(I_i) = \sum_{i=1}^{m} l(I_i^L) + \sum_{i=1}^{m} l(I_i^R)$.

We shall refer to $\sum_{i=1}^{m} l(I_i^L)$ as the *left bin usage time* and $\sum_{i=1}^{m} l(I_i^R)$ as the *right bin usage time*. Obviously, for any two different bins b_i and b_j ($1 \le i \ne j \le m$), it holds that $I_i^R \cap I_j^R = \emptyset$. Thus, it is easy to see that the right bin usage time is bounded by the duration of the third stage, i.e., $t+\rho-t_3 = t+\rho-(t-\Delta) = \rho+\Delta$. This applies to all the item categories.

Moreover, for the first item category (i.e., the items departing in the interval $(0, \rho]$), there is no bin usage in the first Δ time units of the third stage (i.e., from time $-\Delta$ to 0) as it is beyond the span of the item list $\mathcal{R}$. So, its right bin usage time is bounded by ρ.

Similarly, for the last item category, the departure time interval is $\left((\lceil \frac{span(\mathcal{R})}{\rho} \rceil - 1) \cdot \rho, \lceil \frac{span(\mathcal{R})}{\rho} \rceil \cdot \rho\right]$, and the third stage is from time $(\lceil \frac{span(\mathcal{R})}{\rho} \rceil - 1) \cdot \rho - \Delta$ to $\lceil \frac{span(\mathcal{R})}{\rho} \rceil \cdot \rho$. There is bin usage only within the first $\Delta + span(\mathcal{R}) - (\lceil \frac{span(\mathcal{R})}{\rho} \rceil - 1) \cdot \rho$ time units of the third stage as the rest is beyond the span of the item list $\mathcal{R}$. Hence, the right bin usage time for the last item category is bounded by $\Delta + span(\mathcal{R}) - (\lceil \frac{span(\mathcal{R})}{\rho} \rceil - 1) \cdot \rho$.

Thus, the total right bin usage time in the third stages of all the item categories is bounded by

$$\begin{aligned}
&\rho + \left(\left\lceil \frac{span(\mathcal{R})}{\rho} \right\rceil - 2\right) \cdot (\rho + \Delta) \\
&\quad + \left(\Delta + span(\mathcal{R}) - \left(\left\lceil \frac{span(\mathcal{R})}{\rho} \right\rceil - 1\right) \cdot \rho\right) \\
&= span(\mathcal{R}) + \left(\left\lceil \frac{span(\mathcal{R})}{\rho} \right\rceil - 1\right) \cdot \Delta \\
&< span(\mathcal{R}) + \frac{span(\mathcal{R})}{\rho} \cdot \Delta \\
&= \frac{\rho + \Delta}{\rho} \cdot span(\mathcal{R}).
\end{aligned}$$

Next, we examine the left bin usage time in the third stage. We shall derive a lower bound on the time-space demand over the third stage as a function of the left bin usage time in order to bound the competitiveness of the latter. This derivation is an extension of our recent analysis of the original First Fit algorithm [24].

For each open bin b_i where $I_i^L \ne \emptyset$, we find the first item r_i with size less than $\frac{\Delta}{\rho+2\Delta}$ (if any) among all the items placed in bin b_i by the end of I_i^L and let a_i denote the arrival time of item r_i. We consider the following three subperiods $I_i^{L_1}$, $I_i^{L_2}$ and $I_i^{L_3}$, whose union covers the whole I_i^L, i.e., $I_i^L \subseteq I_i^{L_1} \cup I_i^{L_2} \cup I_i^{L_3}$. $I_i^{L_1}$ denotes the period from the beginning of I_i to r_i's arrival, $I_i^{L_2}$ denotes a period of length Δ immediately after r_i's arrival (such a period must exist since item r_i stays in bin b_i for at least Δ time units) and $I_i^{L_3}$ denotes the remaining period in I_i^L if any. Note that r_i may arrive before the third stage starts (i.e., before time t_3) or during the third stage (i.e., after time t_3). If r_i arrives before time t_3, then $I_i^{L_1} = \emptyset$, $I_i^{L_2} = [t_3, t)$ and $I_i^{L_3} = [t, I_i^{L+})$ (for example, bin b_2 in Figure 7). If r_i arrives after time t_3, then $I_i^{L_1} = [I_i^-, a_i)$, $I_i^{L_2} = [a_i, a_i + \Delta)$ and $I_i^{L_3} = [a_i + \Delta, \max\{a_i + \Delta, I_i^{L+}\})$ (for example, bins b_3 and b_5 in Figure 7). Obviously, before the arrival of r_i, the level of bin b_i must be at least $\frac{\Delta}{\rho+2\Delta}$ high if it is open. Thus, b_i's bin level is at least $\frac{\Delta}{\rho+2\Delta}$ high in $I_i^{L_1}$. If no item with size less than $\frac{\Delta}{\rho+2\Delta}$ is placed in bin b_i by the end of I_i^L, we define $I_i^{L_1} = I_i^L$, $I_i^{L_2} = \emptyset$ and $I_i^{L_3} = \emptyset$ (for example, bin b_4 in Figure 7). In this case, again, b_i's bin level is at least $\frac{\Delta}{\rho+2\Delta}$ high in $I_i^{L_1}$. For notational convenience, if $I_i^L = \emptyset$, we also define $I_i^{L_1} = I_i^L = \emptyset$, $I_i^{L_2} = \emptyset$ and $I_i^{L_3} = \emptyset$. Based on the above definitions, we have the following properties:

PROPOSITION 4. *For each bin b_i, if $I_i^{L_1} \ne \emptyset$, b_i's bin level is at least $\frac{\Delta}{\rho+2\Delta}$ high in $I_i^{L_1}$.*

PROPOSITION 5. *For each bin b_i, if $I_i^L \ne \emptyset$ and an item of size less than $\frac{\Delta}{\rho+2\Delta}$ is placed in b_i by the end of I_i^L, then $I_i^{L_2} \ne \emptyset$. Otherwise, $I_i^{L_2} = \emptyset$ and $I_i^{L_3} = \emptyset$.*

To bound the time-space demand, we define the concepts of *supplier bin* and *supplier period*. For each bin b_i where $I_i^L \ne \emptyset$ and an item of size less than $\frac{\Delta}{\rho+2\Delta}$ is placed in b_i by the end of I_i^L, there must exist at least one bin b_h opened before b_i and closed after time a_i, i.e., the arrival time of item r_i. Otherwise, by definition, we have $E_i = \max\{I_j^+ \mid 1 \le j < i\} \le a_i$ and hence a_i should belong to I_i^R rather than I_i^L. Among all such bins opened before bin b_i and closed after time a_i, we define the last opened bin (the bin with the highest index) as the supplier bin of b_i. Consider the following three cases of r_i's arrival time.

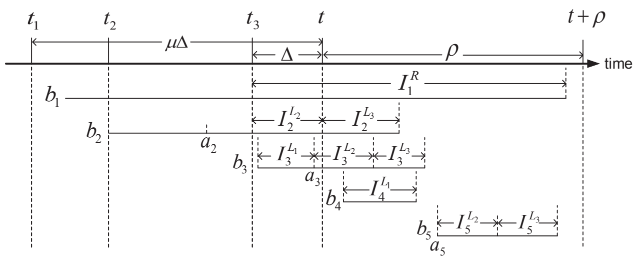

Figure 7: Definitions of $I_i^{L_1}$, $I_i^{L_2}$ and $I_i^{L_3}$

Case 1: Item r_i arrives on or before time t_3, i.e., $a_i \leq t_3$. In this case, b_i's supplier bin must be b_{i-1} since the open bins of the item category can only be closed in the interval $(t, t+\rho]$ in which the items depart. For example, in Figure 7, b_2's supplier bin is b_1. According to the First Fit packing algorithm, at the moment of a_i, the sum of the levels of b_{i-1} and b_i is higher than 1 (the bin capacity). This implies b_{i-1}'s level at time a_i is higher than $1 - s(r_i)$. We define the time interval $[t_3, t)$ associated with bin b_{i-1} as the supplier period of bin b_i and denote it by P_i. Note that the level of b_{i-1} never decreases before t. Also note that the length of period $I_i^{L_2}$ is Δ, and item r_i stays in bin b_i throughout $I_i^{L_2}$. Let $d(P_i)$ denote the total time-space demand of the items in b_i's supplier bin over the supplier period P_i, and let $d(I_i^{L_2})$ denote the total time-space demand of the items in bin b_i over the period $I_i^{L_2}$. It follows that

$$\begin{aligned} d(P_i) + d(I_i^{L_2}) &> \big(1 - s(r_i)\big) \cdot l(P_i) + s(r_i) \cdot l(I_i^{L_2}) \\ &= \big(1 - s(r_i)\big) \cdot \Delta + s(r_i) \cdot \Delta \\ &= \Delta. \end{aligned}$$

Case 2: Item r_i arrives between time t_3 and t, i.e., $t_3 < a_i < t$. In this case, b_i's supplier bin must also be b_{i-1} since no open bin is closed before time t. For example, in Figure 7, b_3's supplier bin is b_2. We define the time interval $[t_3, a_i + \Delta)$ associated with bin b_{i-1} as the supplier period of bin b_i and denote it by P_i. Following similar arguments to Case 1, at the moment of a_i, the level of b_{i-1} is higher than $1 - s(r_i)$. All the items in b_{i-1} at that time remain in b_{i-1} till at least time t. Recall that each item resides in the system for at least Δ time units. Thus, each item in b_{i-1} at time a_i must stay for a duration at least Δ time units long in the time interval $[a_i - (\Delta - (t - a_i)), t + (\Delta - (t - a_i))) = [t_3, a_i + \Delta)$. Also note that item r_i stays in bin b_i throughout $I_i^{L_2}$. Therefore,

$$d(P_i) + d(I_i^{L_2}) > \big(1 - s(r_i)\big) \cdot \Delta + s(r_i) \cdot \Delta = \Delta.$$

Case 3: Item r_i arrives on or after time t, i.e., $a_i \geq t$. For example, $a_5 \geq t$ in Figure 7 and b_5's supplier bin is b_1. Again, the supplier bin of b_i must have a level higher than $1 - s(r_i)$ at time a_i. We define the time interval $[a_i - \Delta, a_i + \Delta)$ associated with the supplier bin as the supplier period of bin b_i and denote it by P_i. Since each item stays in the system for at least a duration of Δ time units, we have $d(P_i) > \big(1 - s(r_i)\big) \cdot \Delta$ and $d(I_i^{L_2}) \geq s(r_i) \cdot \Delta$. Thus,

$$d(P_i) + d(I_i^{L_2}) > \big(1 - s(r_i)\big) \cdot \Delta + s(r_i) \cdot \Delta = \Delta.$$

So far, we have created a supplier period P_i for each bin b_i where $I_i^L \neq \emptyset$ and an item with size less than $\frac{\Delta}{\rho + 2\Delta}$ is placed in b_i by the end of I_i^L. The total time-space demand over periods P_i and $I_i^{L_2}$ is at least Δ, i.e.,

$$d(P_i) + d(I_i^{L_2}) \geq d(P_i) + s(r_i) \cdot l(I_i^{L_2}) > \Delta. \tag{5}$$

If multiple bins share the same supplier bin, their supplier periods may overlap. We group the bins $b_2, b_3, \ldots, b_m$ according to their supplier bins. Let $\mathcal{G}$ be the set of all the bin groups. Consider a bin group $H \in \mathcal{G}$. Suppose there are s bins $b_{j_1}, b_{j_2}, \ldots, b_{j_s}$ in H that share the same supplier bin. Without loss of generality, suppose that $j_1 < j_2 < \cdots < j_s$. We first analyze the total length of the periods $I_{j_1}^{L_2} \cup I_{j_1}^{L_3}$, $I_{j_2}^{L_2} \cup I_{j_2}^{L_3}$, $\ldots$, $I_{j_s}^{L_2} \cup I_{j_s}^{L_3}$ together with the supplier period P_{j_1} of the first bin b_{j_1} in the group.

LEMMA 7. *The total length of the supplier period P_{j_1} together with periods $I_{j_1}^{L_2} \cup I_{j_1}^{L_3}$, $I_{j_2}^{L_2} \cup I_{j_2}^{L_3}$, $\ldots$, $I_{j_s}^{L_2} \cup I_{j_s}^{L_3}$ is bounded by $\rho + 2\Delta$.*

Proof: Please refer to the extended version of this paper [22]. □

In what follows, we shall refer to the lowest-indexed bin in each group $H \in \mathcal{G}$ as its *flag bin* and denote its index by $f(H)$ (i.e., $f(H) = j_1$ in the above discussion).

Lemma 7 together with (5) imply that for each bin group $H \in \mathcal{G}$, we have

$$\begin{aligned} &\frac{\Delta}{\rho + 2\Delta} \cdot \Big(l(P_{f(H)}) + \sum_{b_j \in H} l(I_j^{L_2} \cup I_j^{L_3})\Big) \\ &\leq \frac{\Delta}{\rho + 2\Delta} \cdot (\rho + 2\Delta) \\ &< d(P_{f(H)}) + s(r_{f(H)}) \cdot l(I_{f(H)}^{L_2}) \\ &\leq d(P_{f(H)}) + d(I_{f(H)}^{L_2}). \end{aligned} \tag{6}$$

To compute the aggregate time-space demand of the periods $P_{f(H)}$ and $I_{f(H)}^{L_2}$ for all the bin groups, we need to study the possible overlaps between the periods of different bin groups. Note that the time-space demands repeatedly counted in the overlapping parts are all due to items $r_{f(H)}$'s that have sizes less than $\frac{\Delta}{\rho+2\Delta}$. Also note that the bin levels are at least $\frac{\Delta}{\rho+2\Delta}$ high in the L_1-periods (Proposition 4). It can finally be shown that (see the extended version of this paper [22] for the complete derivation)

$$\frac{\Delta}{\rho + 2\Delta} \cdot \sum_{i=1}^{m} l(I_i^L) \leq \sum_{i=1}^{m} d(I_i). \tag{7}$$

Since the above inequality holds for all the item categories, the total left bin usage time over the third stages of all the item categories is bounded by $(\frac{\rho}{\Delta} + 2) \cdot d_C$, where d_C denotes the total time-space demand of the items over these periods. Putting together the left and right bin usage times, the total bin usage time in the third stages of all the item categories (denoted by $usage_C$) is bounded by

$$usage_C \leq \Big(\frac{\rho}{\Delta} + 2\Big) \cdot d_C + \frac{\rho + \Delta}{\rho} \cdot span(\mathcal{R}). \tag{8}$$

By definition, $d_B + d_C$ is capped by the total time-space demand $d(\mathcal{R})$ of the entire item list $\mathcal{R}$. Therefore, adding up the bin usage times in the three stages of all the item categories (i.e., (3), (4) and (8)) and according to Propositions 1 and 2, the total bin usage time of the classify-by-departure-time First Fit algorithm satisfies

$$\begin{aligned} &usage_A + usage_B + usage_C \\ &< 2 \cdot d_B + \Big(\frac{\rho}{\Delta} + 2\Big) \cdot d_C + \frac{\mu\Delta + \rho}{\rho} \cdot span(\mathcal{R}) \\ &\leq \Big(\frac{\rho}{\Delta} + 2\Big) \cdot (d_B + d_C) + \frac{\mu\Delta + \rho}{\rho} \cdot span(\mathcal{R}) \\ &\leq \Big(\frac{\rho}{\Delta} + 2\Big) \cdot d(\mathcal{R}) + \frac{\mu\Delta + \rho}{\rho} \cdot span(\mathcal{R}) \\ &\leq \Big(\frac{\rho}{\Delta} + 2\Big) \cdot OPT_{total}(\mathcal{R}) + \frac{\mu\Delta + \rho}{\rho} \cdot OPT_{total}(\mathcal{R}) \\ &= \Big(\frac{\rho}{\Delta} + \frac{\mu\Delta}{\rho} + 3\Big) \cdot OPT_{total}(\mathcal{R}). \end{aligned} \tag{9}$$

There exists an optimal value of the algorithm parameter ρ that minimizes the factor $\frac{\rho}{\Delta} + \frac{\mu\Delta}{\rho} + 3$. If the minimum and maximum item durations are known prior to packing, ρ can be set to $\sqrt{\mu}\Delta$ so that $\frac{\rho}{\Delta} + \frac{\mu\Delta}{\rho} + 3$ reaches the minimum value of $2\sqrt{\mu} + 3$. Therefore, we have the following conclusion.

THEOREM 4. *For Clairvoyant MinUsageTime DBP, if the minimum and maximum item durations are not known, the classify-*

by-departure-time First Fit algorithm has a competitive ratio of $\frac{\rho}{\Delta} + \frac{\mu\Delta}{\rho} + 3$*, where* ρ *is the length of the departure time interval for each item category and* Δ *is the minimum item duration. If the minimum and maximum item durations are known, the classify-by-departure-time First Fit algorithm can achieve a competitive ratio of* $2\sqrt{\mu} + 3$*.*

5.3 Classify-by-Duration Strategy

In our recent work, we have established an upper bound of $\mu + 4$ on the competitive ratio of First Fit packing for Non-Clairvoyant MinUsageTime DBP [24]. Observing that the competitive ratio of First Fit packing is a linear function of the max/min item duration ratio μ, we propose another classify-by-duration strategy to make the packing algorithm more competitive by reducing the max/min ratio of item durations. Specifically, we classify the items into categories such that the max/min item duration ratio for each category is a given constant α. Given a base item duration b, each category includes all the items with durations between $b \cdot \alpha^{i-1}$ and $b \cdot \alpha^{i}$ for an integer i. Suppose that the max/min item duration for the entire item list $\mathcal{R}$ is μ. Then, there can be up to $\lceil log_{\alpha}\mu \rceil + 1$ non-empty item categories produced by the classification.[2] Let $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3, \ldots, \mathcal{R}_{\lceil log_{\alpha}\mu \rceil + 1}$ denote these item categories. Since the departure time (and thus the duration) of each item is known at the time of its arrival, the category of the item can be determined at the time of its assignment. We use First Fit packing to pack the items in each category $\mathcal{R}_i$ separately.

In our proof that First Fit packing is $(\mu+4)$-competitive for Non-Clairvoyant MinUsageTime DBP [24], we have established that the total bin usage time of First Fit packing is bounded by

$$(\mu + 3) \cdot d(\mathcal{R}) + span(\mathcal{R}),$$

where $\mathcal{R}$ is the item list packed by First Fit, and $d(\mathcal{R})$ and $span(\mathcal{R})$ are the total time-space demand and the span of $\mathcal{R}$ respectively.

Thus, by applying the classify-by-duration First Fit algorithm, the bin usage time of packing all the items in each category $\mathcal{R}_i$ ($i = 1, 2, 3, \ldots, \lceil log_{\alpha}\mu \rceil + 1$) is bounded by

$$(\alpha + 3) \cdot d(\mathcal{R}_i) + span(\mathcal{R}_i).$$

Therefore, the total bin usage time of classify-by-duration First Fit packing for the entire item list $\mathcal{R}$ is bounded by

$$(\alpha + 3) \cdot \sum_{i=1}^{\lceil log_{\alpha}\mu \rceil + 1} d(\mathcal{R}_i) + \sum_{i=1}^{\lceil log_{\alpha}\mu \rceil + 1} span(\mathcal{R}_i). \tag{10}$$

Note that $d(\mathcal{R}) = \sum_{i=1}^{\lceil log_{\alpha}\mu \rceil + 1} d(\mathcal{R}_i)$ and for any $1 \leq i \leq \lceil log_{\alpha}\mu \rceil + 1$, by definition, $span(\mathcal{R}_i) \leq span(\mathcal{R})$. As a result, the total bin usage time is bounded by

$$(\alpha + 3) \cdot d(\mathcal{R}) + (\lceil log_{\alpha}\mu \rceil + 1) \cdot span(\mathcal{R}).$$

By Propositions 1 and 2, it follows that the total bin usage time of classify-by-duration First Fit packing is bounded by

$$(\alpha + \lceil log_{\alpha}\mu \rceil + 4) \cdot OPT_{total}(\mathcal{R}).$$

If the minimum and maximum item durations are known prior to packing, the base item duration b can be set to match the minimum item duration and the ratio α can be set to $\mu^{\frac{1}{n}}$ such that there are exactly n item categories produced by the classification. In this case, the total bin usage time of classify-by-duration First Fit packing is bounded by $(\mu^{\frac{1}{n}} + n + 3) \cdot OPT_{total}(\mathcal{R})$. Given a value of μ, there exists an optimal number of item categories that minimizes the factor $\mu^{\frac{1}{n}} + n + 3$, which can be found numerically. Therefore, we have the following result.

[2]For example, if $\alpha = 2$ and the minimum and maximum item durations are 1.5 and 4.5 respectively, there are $\lceil log_2 3 \rceil + 1 = 3$ non-empty item categories: $[1, 2)$, $[2, 4)$ and $[4, 8)$.

THEOREM 5. *For Clairvoyant MinUsageTime DBP, if the minimum and maximum item durations are not known, the classify-by-duration First Fit algorithm has a competitive ratio of* $\alpha + \lceil log_{\alpha}\mu \rceil + 4$*, where* α *is the max/min item duration ratio for each item category. If the minimum and maximum item durations are known, the classify-by-duration First Fit algorithm can achieve a competitive ratio of* $\min_{n \geq 1} \mu^{\frac{1}{n}} + n + 3$*.*

We remark that our result significantly improves the existing one of online interval scheduling with bounded parallelism. Shalom *et al.* [23] proposed a BucketFirstFit algorithm that classifies interval jobs according to their lengths and applies First Fit to each category of jobs. They showed that BucketFirstFit has a competitive ratio of $(2\alpha + 2) \cdot \lceil log_{\alpha}\mu \rceil$, where μ is the max/min length ratio of all job intervals and α is the max/min length ratio of job intervals in each category. Our result implies that BucketFirstFit can achieve a competitive ratio of $\alpha + \lceil log_{\alpha}\mu \rceil + 4$ which is asymptotically lower than $(2\alpha + 2) \cdot \lceil log_{\alpha}\mu \rceil$. Our result is also much more general in that we allow items to have arbitrary sizes while the analysis of [23] is restricted to jobs with the same resource demands.

5.4 Numerical results

In this section, we compare the competitive ratios of the classify-by-duration and classify-by-departure-time strategies based on the results of Theorems 4 and 5. Suppose that the minimum and maximum item durations are known prior to packing. Figure 8 shows the best achievable competitive ratios of the classify-by-departure-time and classify-by-duration First Fit algorithms at their respective optimal parameter settings of ρ and n for different max/min item duration ratios μ. For comparison purposes, we also include in Figure 8 the competitive ratio of the original First Fit packing algorithm in the non-clairvoyant setting which is given by $\mu + 4$.

It can be seen that the First Fit algorithms with item classification can achieve asymptotically much lower competitive ratios compared to the original First Fit algorithm. When $\mu < 4$, the classify-by-departure-time strategy has a lower competitive ratio than the classify-by-duration strategy. When $\mu > 4$, the classify-by-duration strategy has a lower competitive ratio than the classify-by-departure-time strategy. This implies that it may be possible to design an even more competitive packing algorithm by combining these two strategies of item classification. Items can first be classified by their durations to reduce the max/min ratio of item durations

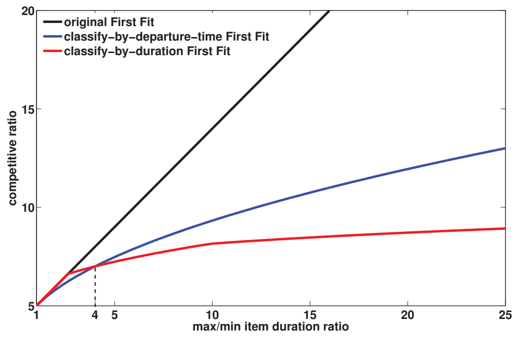

Figure 8: Comparison between classify-by-departure-time and classify-by-duration First Fit

and then be further classified by their departure times for packing. We leave the design of such an algorithm as our future work.

6. CONCLUDING REMARKS

In this paper, we have studied both the offline and online versions of the Clairvoyant MinUsageTime DBP problem. One direction for future work is to design a more competitive online packing algorithm by combining the two proposed item classification strategies for Clairvoyant MinUsageTime DBP. We shall also analyze how inaccurate estimates of item durations would impact the competitiveness. Another direction is to extend the Clairvoyant MinUsageTime DBP problem to model flexible jobs that have release times and deadlines and do not have to be processed immediately upon arrival. Finally, we are also interested in extending MinUsageTime DBP to multiple resource dimensions.

7. ACKNOWLEDGMENTS

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 2 under Grant MOE2013-T2-2-067, and Academic Research Fund Tier 1 under Grant 2013-T1-002-123.

8. REFERENCES

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

[2] Gaikai. http://www.gaikai.com/.

[3] Onlive. http://www.onlive.com/.

[4] M. Alicherry and R. Bhatia. Line system design and a generalized coloring problem. In *ESA 2003: European Symposium on Algorithms*, pages 19–30. Springer, 2003.

[5] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*, volume 53. Cambridge University Press Cambridge, 1998.

[6] 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.

[7] J. W.-T. Chan, P. W. Wong, and F. C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. *Computing and Combinatorics*, pages 309–319, 2006.

[8] J. Chang, S. Khuller, and K. Mukherjee. LP rounding and combinatorial algorithms for minimizing active and busy time. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 118–127, 2014.

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

[10] M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks. Minimizing total busy time in parallel scheduling with application to optical networks. In *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–12, 2009.

[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 Transactions on Multimedia Computing, Communications, and Applications*, 10(1):10, 2014.

[12] V. Jalaparti, P. Bodik, I. Menache, S. Rao, K. Makarychev, and M. Caesar. Network-aware scheduling for data-parallel jobs: Plan when you can. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, pages 407–420, 2015.

[13] S. Kamali and A. López-Ortiz. Efficient online strategies for renting servers in the cloud. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 277–288. Springer, 2015.

[14] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir. Minimizing busy time in multiple machine real-time scheduling. In *FSTTCS 2010: Foundations of Software Technology and Theoretical Computer Science*, pages 169–180. Springer, 2010.

[15] V. Kumar and A. Rudra. Approximation algorithms for wavelength assignment. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science*, pages 152–163. Springer, 2005.

[16] Y. Li, Y. Deng, R. Seet, X. Tang, and W. Cai. MASTER: Multi-platform application streaming toolkits for elastic resources. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM)*, pages 805–806, 2015.

[17] 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.

[18] Y. Li, X. Tang, and W. Cai. Play request dispatching for efficient virtual machine usage in cloud gaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2052–2063, 2015.

[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, 2016.

[20] G. B. Mertzios, M. Shalom, A. Voloshin, P. W. Wong, and S. Zaks. Optimizing busy time on parallel machines. In *Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 238–248, 2012.

[21] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica. Low latency geo-distributed data analytics. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, pages 421–434, 2015.

[22] R. Ren and X. Tang. Clairvoyant dynamic bin packing for job scheduling with minimum server usage time (extended version).
Available at http://www.ntu.edu.sg/home/asxytang/.

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

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

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