On Dynamic Bin Packing for Resource Allocation in the Cloud

Yusen Li
Multi-plAtform Game Innovation Centre (MAGIC)
Nanyang Technological University
Singapore
s080007@e.ntu.edu.sg

Xueyan Tang
Parallel and Distributed Computing Centre
Nanyang Technological University
Singapore
asxytang@ntu.edu.sg

Wentong Cai
Parallel and Distributed Computing Centre
Nanyang Technological University
Singapore
astwcai@ntu.edu.sg

ABSTRACT

Dynamic Bin Packing (DBP) is a variant of classical bin packing, which assumes that items may arrive and depart at arbitrary times. Existing works on DBP generally aim to minimize the maximum number of bins ever used in the packing. In this paper, we consider a new version of the DBP problem, namely, the MinTotal DBP problem which targets at minimizing the total cost of the bins used over time. It is motivated by the request dispatching problem arising in cloud gaming systems. We analyze the competitive ratios of the commonly used First Fit, Best Fit, and Any Fit packing (the family of packing algorithms that open a new bin only when no currently opened bin can accommodate the item to be packed) algorithms for the MinTotal DBP problem. We show that the competitive ratio of Any Fit packing cannot be better than the max/min item interval length ratio μ. The competitive ratio of Best Fit packing is not bounded for any given μ. For First Fit packing, if all the item sizes are smaller than $\frac{W}{k}$ (W is the bin capacity and $k > 1$ is a constant), it has a competitive ratio of $\frac{k}{k-1} \cdot \mu + \frac{6k}{k-1} + 1$. For the general case, First Fit packing has a competitive ratio of $2\mu + 13$. We also propose a Modified First Fit packing algorithm that can achieve a competitive ratio of $\frac{8}{7}\mu + \frac{55}{7}$ when μ is not known and can achieve a competitive ratio of $\mu + 8$ when μ is known.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Combinatorial algorithms*

Keywords

Dynamic bin packing; cloud gaming; request dispatching; approximation algorithms; worst case bounds

SPAA'14, June 23–25, 2014, Prague, Czech Republic.

http://dx.doi.org/10.1145/2612669.2612675.

1. INTRODUCTION

Bin packing is a classical combinatorial optimization problem which has been studied extensively [13, 11]. In the classical bin packing problem, given a set of items, the objective is to pack the items into a minimum number of bins such that the total size of the items in each bin does not exceed the bin capacity. Dynamic bin packing (DBP) is a generalization of the classical bin packing problem [12]. In the DBP problem, each item has a size, an arrival time and a departure time. The item stays in the system from its arrival to its departure. The objective is to pack the items into bins to minimize the maximum number of bins ever used over time. Dynamic bin packing has been widely used to model the resource consolidation problems in cloud computing [19, 25].

In this paper, we consider a new version of the DBP problem, which is called the MinTotal DBP problem. In this problem, we assume that each bin used has a cost that is proportional to the duration of its usage, i.e., the period from its opening (when the first item is put into the bin) to its close (when all the items in the bin depart). The objective is to pack the items into bins to minimize the total cost of packing over time. We focus on the online version of the problem, where the items must be assigned to bins as they arrive without any knowledge of their departure times and future item arrivals. The arrival time and the size of an item are only known when the item arrives and the departure time is only known when the item departs. The items are not allowed to move from one bin to another once they have been assigned upon arrivals.

The MinTotal DBP problem considered in this paper is primarily motivated by the request dispatching problem arising in cloud gaming systems. In a cloud gaming system, computer games run on powerful cloud servers, while players interact with the games via networked thin clients [17]. The cloud servers run the game instances, render the 3D graphics, encode them into 2D videos, and stream them to the clients. The clients then decode and display the video streams. This approach frees players from the overhead of setting up games, the hardware/software incompatibility problems, and the need for upgrading their computers regularly. Cloud gaming is a promising application of the rapidly expanding cloud computing infrastructure, and it has attracted a great deal of interests among entrepreneurs and researchers [22]. Several companies have offered cloud gaming services, such as OnLive [3], StreamMyGame [4],

and GaiKai [2]. The cloud gaming market has been forecasted to reach 8 billion US dollars in 2017 [1].

Running each game instance demands a certain amount of GPU resources and the resource requirement can be different for running different games. In a cloud gaming system, when a playing request is received by the service provider, it needs to be dispatched to a game server that has enough GPU resources to run the game instance of this request. Several game instances can share the same game server as long as the server's GPU resources are not saturated. Each game instance keeps running in the system until the user stops playing the game. In general, the migration of game instances from one game server to another is not preferable due to large migration overheads and interruption to game play. In order to provide a good user experience, the gaming service provider needs to maintain a set of game servers with powerful GPUs for rendering the game instances. Constant workload fluctuation in cloud gaming makes the provisioning of game servers a challenging issue. The on-demand resource provisioning services in public clouds like Amazon EC2 provide an attractive solution. With these services, game service providers can rent virtual machines on demand to serve as game servers and pay for the resources according to their running hours. This frees game service providers from the complex process of planning, purchasing, and maintaining hardware. This approach has been adopted by many cloud gaming service providers like Gaikai and OnLive [26]. In the cloud gaming systems that use public clouds, one natural and important issue is how to dispatch the playing requests to game servers (i.e., virtual machines) so that the total cost of renting the game servers is minimized. The online MinTotal DBP problem we have defined exactly models this issue, where the game servers and playing requests correspond to the bins and items respectively.

For online bin packing, Any Fit packing algorithms have been extensively studied since they are simple and make decisions based on the current system state only. Any Fit packing refers to the family of packing algorithms that open a new bin only when no currently opened bin can accommodate the item to be packed. First Fit and Best Fit are two commonly used Any Fit packing algorithms. First Fit packing attempts to put a new item into the earliest opened bin that can accommodate the item. Best Fit attempts to assign a new item to the bin with smallest residual capacity that can accommodate the item. In this paper, we analyze the performance of the First Fit, Best Fit and Any Fit packing algorithms for the MinTotal DBP problem. We assume that an infinite number of bins are available for packing and all the bins have the same capacity and the same cost.

The contributions of this paper are as follows. We prove that the competitive ratio of Any Fit packing cannot be better than the max/min item interval length ratio μ. The competitive ratio of Best Fit packing is not bounded for any given μ. We show that for the case where all the item sizes are smaller than $\frac{W}{k}$ (W is the bin capacity and $k > 1$ is a constant), First Fit packing has a competitive ratio of $\frac{k}{k-1} \cdot \mu + \frac{6k}{k-1} + 1$. For the general case, First Fit packing has a competitive ratio of $2\mu + 13$. In addition, we propose a Modified First Fit packing algorithm which classifies and assigns items according to their sizes. Modified First Fit packing can achieve a competitive ratio of $\frac{8}{7}\mu + \frac{55}{7}$ when μ is not known and can achieve a competitive ratio of $\mu + 8$ when μ is known.

The rest of this paper is structured as follows. The related work is summarized in Section 2. Section 3 introduces the system model, notations and packing algorithms. In Sections 4.1 to 4.3, the competitive ratios of First Fit, Best Fit, and Any Fit packing for the MinTotal DBP problem are analyzed. Then, the Modified First Fit packing algorithm is proposed and its competitive ratio for the MinTotal DBP problem is analyzed in Section 4.4. Finally, conclusions are made and future work is discussed in Section 5.

2. RELATED WORK

Cloud gaming systems have been implemented for both commercial use and research studies [17, 3, 4]. However, most of the existing work has focused on measuring the performance of cloud gaming systems [10, 24]. To the best of our knowledge, the resource management issues of cloud gaming have never been studied. The MinTotal DBP problem studied in this paper is related to a variety of research topics including the classical bin packing problem and its variations, as well as the interval scheduling problem.

The classical bin packing problem aims to put a set of items into the least number of bins. The problem and its variations have been studied extensively in both the offline and online versions [11, 15]. It is well known that the offline version of the classical bin packing problem is NP-hard already [16]. For the online version, each item must be assigned to a bin without the knowledge of subsequent items. The items are not allowed to move from one bin to another. So far, the best upper bound on the competitive ratio for classical online bin packing is 1.58889, which is achieved by the HARMONIC++ algorithm proposed in [23]. The best known lower bound for any online packing algorithm is 1.54037 [5].

Dynamic bin packing is a variant of the classical bin packing problem [12]. It generalizes the problem by assuming that items may arrive and depart at arbitrary times. The objective is to minimize the maximum number of bins ever used in the packing. Coffman et al. [12] showed that the First Fit packing algorithm has a competitive ratio between 2.75 to 2.897 and no online algorithm can achieve a competitive ratio smaller than 2.5. Joseph et al. [9] proved that the lower bound 2.5 on the competitive ratio also holds when the offline algorithm does not repack. Ivkovic et al. [18] studied an even more general problem called the fully dynamic bin packing problem, where the migration of items is allowed. They proposed an online algorithm that achieves a competitive ratio of 1.25. Chan et al. [8] studied dynamic bin packing of unit fractions items (i.e., each item has a size $\frac{1}{w}$ for some integer $w \geq 1$). They showed that Any Fit packing algorithms have a tight competitive ratio of 3. They also proved that no online algorithm can achieve a competitive ratio better than 2.428. Classical dynamic bin packing does not consider bin usage costs and focuses simply on minimizing the maximum number of bins ever used. In contrast, the MinTotal DBP problem considered in this paper aims to minimize the total cost of the bins used in the packing.

The interval scheduling problem is also related to our problem [20]. The classical interval scheduling problem considers a set of jobs, each associated with a weight and an interval over which the job should be executed. Each machine can process only a single job at any time. Given a fixed number of machines, the objective is to schedule a feasible subset of jobs whose total weight is maximized [6].

Michele et al. [14] have extended the classical model to a more general version, which is called interval scheduling with bounded parallelism. In this model, each machine can process $g > 1$ jobs simultaneously. If there is a job running on a machine, the machine is called busy. The objective is to assign the jobs to the machines such that the total busy time of the machines is minimized. It was proved that the problem to minimize the total busy time is NP-hard for $g \geq 2$ and a 4-competitive offline algorithm was proposed. George et al. [21] considered two special instances: clique instances (the intervals of all jobs share a common time point) and proper instances (the intervals of all jobs are not contained in one another), and provided constant factor approximation algorithms. However, the interval scheduling problem differs from our problem because the ending time of a job is known at the time of its assignment in interval scheduling, whereas in our MinTotal DBP model, the departure time is not known at the time of item assignment. Furthermore, our MinTotal DBP problem does not assume that all the items have the same size, so the number of items that can be packed into a bin is not fixed.

3. PRELIMINARIES

3.1 Notations and Definitions

We first define the notations used in this paper. Each item r to pack is associated with a 3-tuple $(a(r), d(r), s(r))$, where $a(r)$, $d(r)$ and $s(r)$ denote the arrival time, the departure time and the size of r respectively. Let $I(r)$ denote the time interval $[a(r), d(r)]$ in which item r stays in the system (assume that $d(r) > a(r)$ is always true). We say that item r is *active* during this interval. The interval length of item r is represented by $len(I(r)) = d(r) - a(r)$. For any list of items $\mathcal{R}$, we define the span of $\mathcal{R}$ as $span(\mathcal{R}) = len(\bigcup_{r\in\mathcal{R}} I(r))$, i.e., the length of time in which at least one item in $\mathcal{R}$ is active. Figure 1 shows an example of the span. Let $u(r) = s(r) \cdot len(I(r))$ denote the *resource demand* of item r. For any list of items $\mathcal{R}$, we define the total resource demand of $\mathcal{R}$ as $u(\mathcal{R}) = \sum_{r\in\mathcal{R}} u(r)$.

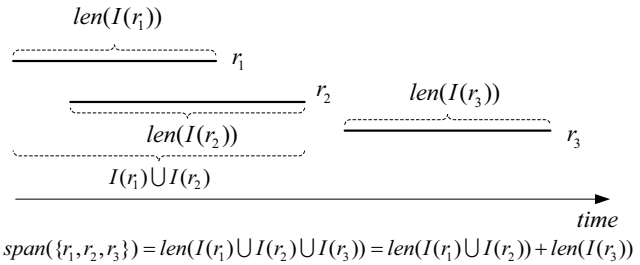

Figure 1: Span of an item list

At any time, the set of all the items in an opened bin is called the *bin configuration*. The total size of the items in an opened bin is called the *level* of the bin. To represent bin configurations, we denote by $x|_y$ a total size of x that is composed of items with size y each. For example, $1|_{\frac{1}{3}}$ means a total size 1 constituted by 3 items of size $\frac{1}{3}$ each. Then, a bin configuration can be represented in the form of $\langle x_1|_{y_1}, x_2|_{y_2}, ..., x_k|_{y_k}\rangle$. For example, $\langle \frac{1}{2}|_{\frac{1}{2}}, \frac{2}{5}|_{\frac{1}{10}}\rangle$ represents a bin with a level of $\frac{9}{10}$, which packs one item of size $\frac{1}{2}$ and 4 items of size $\frac{1}{10}$ each.

Table 1: Summary of Key Notations

Notation	Definition
$a(r)$	the arrival time of an item r
$d(r)$	the departure time of an item r
$s(r)$	the size of an item r
$I(r)$	the time interval of $[a(r), d(r)]$
$len(I(r))$	the length of $I(r)$, $len(I(r)) = d(r) - a(r)$
$u(r)$	the resource demand of an item r, $u(r) = s(r) \cdot len(I(r))$
$span(\mathcal{R})$	the span of an item list $\mathcal{R}$, $span(\mathcal{R}) = len(\bigcup_{r\in\mathcal{R}} I(r))$
$u(\mathcal{R})$	the total resource demand of an item list $\mathcal{R}$, $u(\mathcal{R}) = \sum_{r\in\mathcal{R}} u(r)$
W	the bin capacity
C	the bin cost rate
$x\|_y$	a total size of x that is composed of items with size y each
$\langle x_1\|_{y_1}, ..., x_k\|_{y_k}\rangle$	a bin configuration
$A(\mathcal{R}, t)$	the number of opened bins at time t by algorithm A in packing an item list $\mathcal{R}$
$A_{total}(\mathcal{R})$	the total cost incurred by algorithm A in packing $\mathcal{R}$ over the packing period, $A_{total}(\mathcal{R}) = \int_{\min_{r\in\mathcal{R}} a(r)}^{\max_{r\in\mathcal{R}} d(r)} A(\mathcal{R}, t) \cdot C \, \mathrm{d}t$
$OPT(\mathcal{R}, t)$	the number of bins needed by optimally repacking the active items in an item list $\mathcal{R}$ at time t
$OPT_{total}(\mathcal{R})$	$OPT_{total}(\mathcal{R}) = \int_{\min_{r\in\mathcal{R}} a(r)}^{\max_{r\in\mathcal{R}} d(r)} OPT(\mathcal{R}, t) \cdot C \, \mathrm{d}t$

Let W and C denote the bin capacity and cost rate respectively. Let $n(t)$ denote the number of opened bins at time t. Given a list of items $\mathcal{R}$ to pack, we refer to the period from the first item arrival to the last item departure, i.e., $[\min_{r\in\mathcal{R}} a(r), \max_{r\in\mathcal{R}} d(r)]$ as the *packing period*. The MinTotal DBP problem targets at minimizing the total cost of the bins used over the packing period, i.e.,

$$\text{minimize} \int_{\min_{r\in\mathcal{R}} a(r)}^{\max_{r\in\mathcal{R}} d(r)} n(t) \cdot C \, \mathrm{d}t$$

The above notations are summarized in Table 1.

3.2 Packing Algorithms

As aforementioned, we consider the commonly used First Fit, Best Fit, and Any Fit packing algorithms in this paper. Their formal definitions are given below.

- **First Fit (FF)**: Each time when a new item arrives, First Fit packing tries to put it into the earliest opened bin that can accommodate it. If none of the opened bins has enough residual capacity to accommodate the new item, then a new bin is opened. When all the items in a bin depart, the bin is closed.
- **Best Fit (BF)**: Each time when a new item arrives, Best Fit packing tries to put it into the best opened bin, i.e., the one with the smallest residual capacity after adding the item. If no opened bin has sufficient residual capacity to accommodate the new item, then a new bin is opened. When all the items in a bin depart, the bin is closed.

- **Any Fit (AF)**: Each time when a new item arrives, Any Fit packing can put the item into any opened bin that can accommodate it if one exists. If no opened bin has adequate residual capacity to accommodate the new item, then a new bin is opened. When all the items in a bin depart, the bin is closed. It is easy to see that First Fit and Best Fit are special cases of Any Fit packing.

The performance of an online algorithm is normally measured by its competitive ratio, i.e., the worst-case ratio between the cost of the solution constructed by the algorithm and the cost of an optimal solution [7]. In our MinTotal DBP problem, for an online packing algorithm A, let $A(\mathcal{R},t)$ denote the number of opened bins at time t by A in packing a list of items $\mathcal{R}$. The total cost incurred by A in packing $\mathcal{R}$ over the packing period is given by

$$A_{total}(\mathcal{R}) = \int_{\min_{r\in\mathcal{R}} a(r)}^{\max_{r\in\mathcal{R}} d(r)} A(\mathcal{R},t)\cdot C \ \mathrm{d}t$$

Let $OPT(\mathcal{R},t)$ denote the minimum achievable number of bins into which all the active items at time t can be repacked. Define

$$OPT_{total}(\mathcal{R}) = \int_{\min_{r\in\mathcal{R}} a(r)}^{\max_{r\in\mathcal{R}} d(r)} OPT(\mathcal{R},t)\cdot C \ \mathrm{d}t$$

In the MinTotal DBP problem, we shall analyze the worst-case ratio of $A_{total}(\mathcal{R})$ to $OPT_{total}(\mathcal{R})$. The worst-case ratio is defined as the positive value α such that the following relation holds for any instance of the problem [12]:

$$A_{total}(\mathcal{R}) \le \alpha \cdot OPT_{total}(\mathcal{R})$$

4. THE COMPETITIVE RATIOS

In this section, we analyze the competitive ratios of the packing algorithms for the MinTotal DBP problem. First, for a given item list $\mathcal{R}$, we have the following obvious bounds for the total cost of any packing algorithm A.

Bound (b.1) $A_{total}(\mathcal{R}) \ge \frac{u(\mathcal{R})\cdot C}{W}$

Bound (b.2) $A_{total}(\mathcal{R}) \ge span(\mathcal{R})\cdot C$

Bound (b.3) $A_{total}(\mathcal{R}) \le \sum_{r\in\mathcal{R}} len(I(r))\cdot C$

Bound (b.1) is derived by assuming that no bin capacity is wasted at any time. Bound (b.2) is derived from the fact that at least one bin must be in use at any time when there is at least one active item. Bound (b.3) is derived by assuming that each item is assigned to a new bin.

For any item list $\mathcal{R}$, let $\mu = \frac{\max_{r\in\mathcal{R}} len(I(r))}{\min_{r\in\mathcal{R}} len(I(r))}$ denote the max/min item interval length ratio, where $\max_{r\in\mathcal{R}} len(I(r))$ is the maximum interval length among all the items $r\in\mathcal{R}$ and $\min_{r\in\mathcal{R}} len(I(r))$ is the minimum interval length among all the items $r\in\mathcal{R}$.

4.1 A Lower Bound for Any Fit Packing

First, we have the following result for Any Fit packing.

THEOREM 1. *For the MinTotal DBP problem, the competitive ratio of Any Fit packing has a lower bound given by the max/min item interval length ratio μ.*

PROOF. Without loss of generality, suppose $W=1$. Let Δ be the minimum item interval length and $\mu\Delta$ be the maximum item interval length. Let k be an integer. At time 0, let k^2 items of size $\frac{1}{k}$ arrive. Any Fit packing needs to open k bins to pack these items. At time Δ, let some items depart such that each opened bin has only one item left. At time $\mu\Delta$, all the remaining items leave the system.

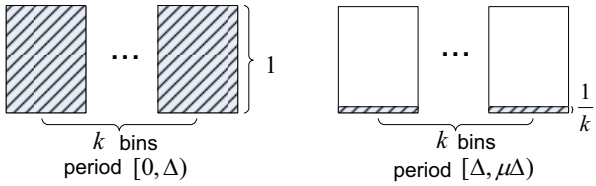

Figure 2: Bin configurations in Any Fit packing

As shown in Figure 2, there are always k opened bins in Any Fit packing from time 0 to $\mu\Delta$. Therefore, the total cost of Any Fit packing is $AF_{total}(\mathcal{R})=k\mu\Delta\cdot C$. From time Δ to $\mu\Delta$, there are only k active items in the system which can in fact be packed into one bin. Therefore, $OPT_{total}(\mathcal{R})=k\Delta\cdot C + (\mu-1)\Delta\cdot C$. It follows that

$$\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})} = \frac{k\mu\Delta\cdot C}{k\Delta\cdot C+(\mu-1)\Delta\cdot C} = \frac{k\mu}{k+\mu-1} \quad (1)$$

According to (1), given any small value ϵ, we can always find an integer k such that $\frac{AF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})} > \mu-\epsilon$. Therefore, the competitive ratio of Any Fit packing is at least μ and the theorem is proven.[1] □

4.2 Best Fit Packing

Next, we have the following result for Best Fit packing.

THEOREM 2. *For the MinTotal DBP problem, Best Fit packing has no bounded competitive ratio for any given max/min item interval length ratio μ.*

PROOF. Without loss of generality, suppose $W=1$. Let k be an integer. Let Δ be the minimum item interval length and $\mu\Delta$ be the maximum item interval length. Suppose that all the items have the same size ε, which is a sufficiently small value and $\frac{1}{\varepsilon}$ is an integer.

At time 0, let $\frac{k}{\varepsilon}$ items arrive. Best Fit packing needs to open k bins to pack all these items since their total size is k. Denote these k bins by $b_1, b_2, ..., b_k$. At time Δ, for each bin b_i, let some items depart to form the bin configuration $\langle(\frac{1}{k}-i\cdot\varepsilon)|_\varepsilon\rangle$ at b_i.

Then, let items arrive and depart according to the following iterative process. In the jth ($j\ge 1$) iteration, k groups of items arrive sequentially in the period $[j\mu\Delta-\delta, j\mu\Delta]$, where δ is a very small variable. The items in each group arrive at the same time and the mth group has $\frac{\frac{1}{k}-(j\cdot k+m)\cdot\varepsilon}{\varepsilon}$ items. By using Best Fit packing, the items in the first group (i.e., $m=1$) will be assigned to b_1 since b_1 is the bin with the highest level in the system. After the items in the first group are packed, before the second group of items arrive, let all the "old" items in b_1 (the items arrived before time $j\mu\Delta-\delta$) depart. The new configuration of b_1

[1] In fact, this example and the lower bound μ are applicable to any online packing algorithm.

will become $\langle(\frac{1}{k}-(jk+1)\cdot\varepsilon)|_\varepsilon\rangle$. Then, the second group will be assigned to bin b_2, and so on so forth. In general, the items in the mth group will be packed in b_m since b_m is the bin with the highest level in the system when the mth group of items arrive. Before the $(m+1)$th group of items arrive, let the "old" items in b_m depart to form the configuration $\langle(\frac{1}{k}-(jk+m)\cdot\varepsilon)|_\varepsilon\rangle$ at b_m. Figure 3 shows the bin configurations in the first few iterations.

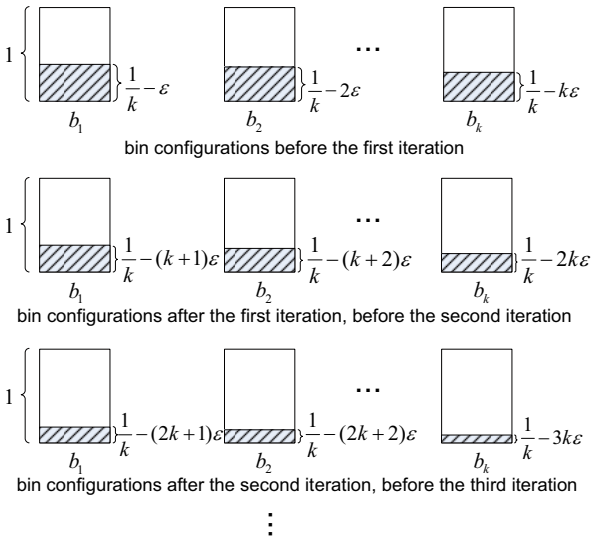

Figure 3: Bin configurations in Best Fit packing

Consider the time interval $[0, n\mu\Delta]$ in the above packing process, where n is an integer. Since there are always k bins in the system, the total cost incurred by Best Fit packing is $BF_{total}(\mathcal{R}) = kn\mu\Delta\cdot C$. On the other hand, the total resource demand in the period $[0,\Delta]$ is $k\Delta$. After time Δ, except the periods $[j\mu\Delta-\delta, j\mu\Delta]$ (for each $1\le j\le n$), all the active items in the system can be packed into one bin at any time. For the periods $[j\mu\Delta-\delta, j\mu\Delta]$, at most two bins are required to pack all the active items. Therefore,

$$\begin{aligned} OPT_{total}(\mathcal{R}) &\le k\Delta\cdot C + (n\mu\Delta-\Delta-n\delta)\cdot C + n\delta\cdot 2C \\ &= k\Delta\cdot C + (n\mu\Delta-\Delta)\cdot C + n\delta\cdot C \end{aligned}$$

It follows that

$$\begin{aligned} \frac{BF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})} &\ge \frac{kn\mu\Delta\cdot C}{k\Delta\cdot C + (n\mu\Delta-\Delta)\cdot C + n\delta\cdot C} \\ &= \frac{kn\mu\Delta}{k\Delta + (n\mu\Delta-\Delta) + n\delta} \end{aligned}$$

It can be proved that when $n \ge \frac{(k-1)\Delta}{\mu\Delta-\delta}$, we have

$$\frac{BF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})} \ge \frac{k}{2} \tag{2}$$

Inequality (2) implies that the ratio $\frac{BF_{total}(\mathcal{R})}{OPT_{total}(\mathcal{R})}$ can be made arbitrarily large as k goes towards infinity. Therefore, Best Fit has no bounded competitive ratio for any given max/min item interval length ratio μ and the theorem is proven. □

4.3 First Fit Packing

Now, we study the competitive ratio of First Fit packing. We start by analyzing two particular cases in which all the item sizes are "large" and "small" respectively. First, we have the following result for the case of "large" items.

THEOREM 3. *For the MinTotal DBP problem, for any item list $\mathcal{R}$, if the item size $s(r)\ge\frac{W}{k}$ ($k>1$ is a constant) for all the items $r\in\mathcal{R}$, the total cost of First Fit packing is at most $k\cdot OPT_{total}(\mathcal{R})$.*

PROOF. According to Bound (b.3), the total cost of any packing algorithm is at most $C\cdot\sum_{r\in\mathcal{R}} len(I(r))$. If all the item sizes are larger than or equal to $\frac{W}{k}$, we have

$$\begin{aligned} C\cdot\sum_{r\in\mathcal{R}} len(I(r)) &= C\cdot\sum_{r\in\mathcal{R}}\frac{u(r)}{s(r)} \\ &\le C\cdot\frac{\sum_{r\in\mathcal{R}}u(r)}{\frac{W}{k}} \\ &\le k\cdot C\cdot\frac{u(\mathcal{R})}{W} \\ &\le k\cdot OPT_{total}(\mathcal{R}) \end{aligned} \tag{3}$$

The last step is based on Bound (b.1). Hence, the theorem is proven. □

Next, we analyze the case of "small" items. It serves as the basis for analyzing the competitive ratio of First Fit packing in the general case, as well as for designing a Modified First Fit packing algorithm in Section 4.4 with improved competitive ratios.

Consider an item list $\mathcal{R}$ in which each item $r\in\mathcal{R}$ has a size $s(r)<\frac{W}{k}$ ($k>1$ is a constant). Let Δ be the minimum item interval length and $\mu\Delta$ be the maximum item interval length. Suppose a total of m bins $b_1, b_2, ..., b_m$ are used in First Fit packing to pack $\mathcal{R}$. For each bin b_i, let I_i denote the usage period of b_i, i.e., the period from the time when b_i is opened to the time when b_i is closed. Let I_i^- and I_i^+ denote the left and right endpoints of I_i respectively, then I_i can be represented by $[I_i^-, I_i^+]$. Denote the length of I_i by $len(I_i)$, then $len(I_i) = I_i^+ - I_i^-$. Without loss of generality, assume that the bins are indexed in the temporal order of their openings, i.e., $I_1^-\le I_2^-\le\cdots\le I_m^-$.

Let $\mathcal{R}_i$ denote the set of items that are assigned to b_i in First Fit packing, then we have $I_i = \bigcup_{r\in\mathcal{R}_i} I(r)$. The total cost of First Fit packing is given by

$$FF_{total}(\mathcal{R}) = C\cdot\sum_{i=1}^{m} len(I_i)$$

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. Let I_i^L denote the period $[I_i^-, \min\{I_i^+, E_i\}]$. If $E_i\le I_i^-$, define $I_i^L=\emptyset$. Let $I_i^R = I_i - I_i^L$ be the remaining period. For the first opened bin b_1, we define E_1 to be the start of the packing period so that $I_1^L=\emptyset$ and $I_1^R = I_1$. Figure 4 shows an example of these definitions. According to the definitions, we have $len(I_i) = len(I_i^L) + len(I_i^R)$ and it follows that

$$FF_{total}(\mathcal{R}) = C\cdot\sum_{i=1}^{m}\Big(len(I_i^L)+len(I_i^R)\Big) \tag{4}$$

Obviously, for any two different bins b_i and b_j, $I_i^R\cap I_j^R=\emptyset$. It is also easy to see that $span(\mathcal{R}) = len(\bigcup_{i=1}^m I_i^R)$. There-

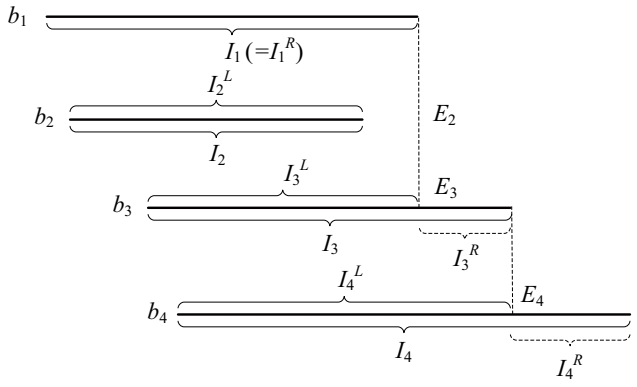

Figure 4: An example of usage periods

fore,

$$span(\mathcal{R}) = \sum_{i=1}^{m} len(I_i^R) \tag{5}$$

According to (4) and (5), we have

$$FF_{total}(\mathcal{R}) = C \cdot \left(\sum_{i=1}^{m} len(I_i^L) \right) + C \cdot span(\mathcal{R}) \tag{6}$$

For each period I_i^L, if its length $len(I_i^L) > (\mu + 2)\Delta$, we split I_i^L into $\lceil \frac{len(I_i^L)}{(\mu+2)\Delta} \rceil$ sub-periods by inserting splitter points that are multiples of $(\mu + 2)\Delta$ before the end of I_i^L, i.e., at times $\min\{I_i^+, E_i\} - k \cdot (\mu + 2)\Delta$, for $k = 1, 2, \ldots, \lceil \frac{len(I_i^L)}{(\mu+2)\Delta} \rceil - 1$. After splitting, if the length of the first sub-period is shorter than 2Δ, we merge the first two sub-periods into one. Then, we label all the sub-periods in the temporal order by $I_{i,1}$, $I_{i,2}$, $I_{i,3}$, ..., i.e., their left endpoints satisfy $I_{i,1}^- < I_{i,2}^- < I_{i,3}^- < \cdots$. Figure 5 shows an example of the period split and mergence.

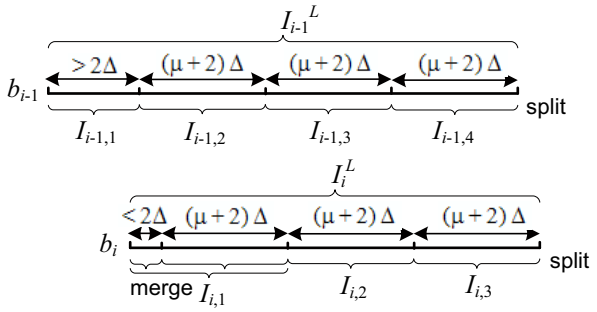

Figure 5: An example of period split and mergence

Note that if the length of I_i^L does not exceed $(\mu + 2)\Delta$, I_i^L is not split. In this case, we define $I_{i,1} = I_i^L$. The above splitting and merging process implies the following features:

Feature (f.1) $len(I_{i,j}) \leq (\mu + 4)\Delta$ for any i and j.

Feature (f.2) $len(I_{i,j}) = (\mu + 2)\Delta$ for any $j \geq 2$ and any i.

Feature (f.3) For any i, if there exists a sub-period $I_{i,j}$ where $j \geq 2$, then the length of the first sub-period $len(I_{i,1}) \geq 2\Delta$.

Let $\mathcal{I}^L$ denote the set of all the sub-periods produced by the above splitting and merging process from all I_i^L's. Define $len(\mathcal{I}^L) = \sum_{I_{i,j} \in \mathcal{I}^L} len(I_{i,j})$. It is obvious that

$$\sum_{i=1}^{m} len(I_i^L) = len(\mathcal{I}^L) \tag{7}$$

For each period $I_{i,j}$, it can be shown that at least one new item must be packed into bin b_i during $I_{i,j} = [I_{i,j}^-, I_{i,j}^+]$. In fact, if $len(I_{i,j}) \geq (\mu + 2)\Delta$, there must be at least one new item packed into b_i during $[I_{i,j}^-, I_{i,j}^- + \mu\Delta]$. This is because all the items packed into b_i before $I_{i,j}^-$ would have departed by time $I_{i,j}^- + \mu\Delta$ since $\mu\Delta$ is the maximum item interval length. Thus, if no new item is packed into b_i during $[I_{i,j}^-, I_{i,j}^- + \mu\Delta]$, b_i would become empty and be closed by time $I_{i,j}^- + \mu\Delta$. On the other hand, if $len(I_{i,j}) < (\mu + 2)\Delta$, according to Feature (f.2), $I_{i,j}$ must be the first sub-period in I_i^L, i.e., $j = 1$. Since b_i is opened at time $I_i^- = I_{i,1}^-$, at least one new item is packed into b_i at time $I_{i,1}^-$.

Let $t_{i,j}$ denote the time point when the "earliest" item (among all the items that are newly packed into b_i in period $I_{i,j}$) is packed into b_i in $I_{i,j}$. We refer to $t_{i,j}$ as the *reference point* of $I_{i,j}$. The above analysis implies that:

Feature (f.4) For each period $I_{i,1}$, it holds that $t_{i,1} = I_{i,1}^-$.

Feature (f.5) For each period $I_{i,j}$, it holds that $I_{i,j}^- \leq t_{i,j} \leq I_{i,j}^- + \mu\Delta$.

It is also easy to infer that there must exist at least one bin b_k satisfying $k < i$ and $t_{i,j} < I_k^+$. Otherwise, $t_{i,j}$ would not be contained in any period I_k where $k < i$. That is, b_i would be the opened bin with the lowest index at time $t_{i,j}$. According to the previous definitions, $t_{i,j}$ would then belong to I_i^R, which contradicts that $t_{i,j}$ is in $I_{i,j}$ (a sub-period of I_i^L). Among all the bins b_k satisfying $k < i$ and $t_{i,j} < I_k^+$, we define the last opened bin (the bin with the highest index) as the *reference bin* of $I_{i,j}$, and denote it by $b^\dagger(I_{i,j})$. We define the period $[t_{i,j} - \Delta, t_{i,j} + \Delta]$ associated with bin $b^\dagger(I_{i,j})$ as the *reference period* of $I_{i,j}$, and denote it by $p^\dagger(I_{i,j})$. Figure 6 shows an example of reference bins and reference periods.

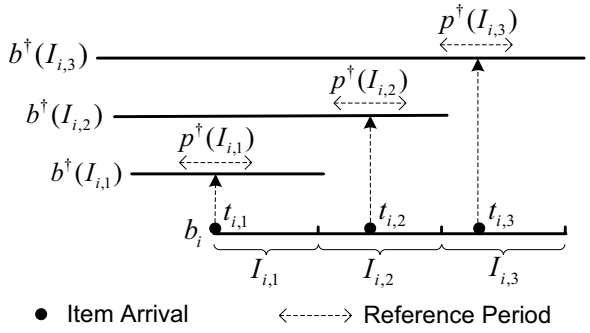

Figure 6: An example of reference bins and periods

Since there is a new item packed into b_i at time $t_{i,j}$ and the item size is smaller than $\frac{W}{k}$, the reference bin $b^\dagger(I_{i,j})$ must

have a level higher than $W - \frac{W}{k}$ at time $t_{i,j}$ according to the First Fit packing algorithm. That is, the total size of the items in $b^\dagger(I_{i,j})$ at time $t_{i,j}$ is larger than $W - \frac{W}{k}$. Recall that each of these items resides in the system for at least Δ time (the minimum item interval length). Thus, each of them must stay in bin $b^\dagger(I_{i,j})$ for at least Δ time during the reference period $p^\dagger(I_{i,j}) = [t_{i,j} - \Delta, t_{i,j} + \Delta]$. Denote by $u(p^\dagger(I_{i,j}))$ the total resource demand of the items in bin $b^\dagger(I_{i,j})$ over period $p^\dagger(I_{i,j})$. It follows that

$$u(p^\dagger(I_{i,j})) \geq \left(W - \frac{W}{k}\right) \cdot \Delta \tag{8}$$

We define that two reference periods intersect if and only if they are associated with the same bin and their time intervals overlap. Then, for any two different periods I_{i_1,j_1} and I_{i_2,j_2}, their reference periods intersect if and only if $b^\dagger(I_{i_1,j_1}) = b^\dagger(I_{i_2,j_2})$ and $|t_{i_1,j_1} - t_{i_2,j_2}| < 2\Delta$. Next, we analyze whether two reference periods will intersect according to the five cases classified by Table 2.

Table 2: Case Classification

	$j_1 \geq 2$, $j_2 \geq 2$	$j_1 = 1$, $j_2 \geq 2$ **or** $j_1 \geq 2$, $j_2 = 1$	$j_1 = 1$, $j_2 = 1$
$i_1 = i_2$	Case I	Case II	—
$i_1 \neq i_2$	Case III	Case IV	Case V

For Cases I, II, III and IV, we have the following lemma.

LEMMA 1. *The reference periods of I_{i_1,j_1} and I_{i_2,j_2} do not intersect in Cases I, II, III and IV.*

PROOF. For convenience, denote $I_{i_1,j_1} = [I^-_{i_1,j_1}, I^+_{i_1,j_1}]$ and $I_{i_2,j_2} = [I^-_{i_2,j_2}, I^+_{i_2,j_2}]$.

For Case I, since $j_1 \geq 2$ and $j_2 \geq 2$, it follows from Feature (f.2) that $len(I_{i_1,j_1}) = len(I_{i_2,j_2}) = (\mu + 2)\Delta$. Without loss of generality, suppose $j_1 < j_2$. Since $i_1 = i_2$, it follows that $I^+_{i_1,j_1} \leq I^-_{i_2,j_2}$. According to Features (f.2) and (f.5), we have $t_{i_1,j_1} \leq I^-_{i_1,j_1} + \mu\Delta = I^+_{i_1,j_1} - 2\Delta$ and $t_{i_2,j_2} \geq I^-_{i_2,j_2} \geq I^+_{i_1,j_1}$. As a result, $t_{i_2,j_2} - t_{i_1,j_1} \geq 2\Delta$. Therefore, the two reference periods cannot intersect.

For Case II, without loss of generality, suppose $j_1 = 1$ and $j_2 \geq 2$. Since $i_1 = i_2$, we have $I^+_{i_1,j_1} \leq I^-_{i_2,j_2}$. According to Feature (f.4), $t_{i_1,j_1} = I^-_{i_1,j_1}$. It also follows from Feature (f.3) that $len(I_{i_1,j_1}) = I^+_{i_1,j_1} - I^-_{i_1,j_1} \geq 2\Delta$. Moreover, based on Feature (f.5), we have $t_{i_2,j_2} \geq I^-_{i_2,j_2} \geq I^+_{i_1,j_1}$. Thus, $t_{i_2,j_2} - t_{i_1,j_1} = t_{i_2,j_2} - I^-_{i_1,j_1} \geq I^+_{i_1,j_1} - I^-_{i_1,j_1} \geq 2\Delta$. Therefore, the two reference periods cannot intersect.

For Case III, without loss of generality, suppose $i_1 < i_2$. If $b^\dagger(I_{i_1,j_1}) \neq b^\dagger(I_{i_2,j_2})$, the two reference periods must not intersect according to the definition. If $b^\dagger(I_{i_1,j_1}) = b^\dagger(I_{i_2,j_2})$, it must hold that $t_{i_2,j_2} \geq I^+_{i_1} \geq I^+_{i_1,j_1}$. This is because otherwise, the reference bin of I_{i_2,j_2} would be b_{i_1}, which cannot be the same as $b^\dagger(I_{i_1,j_1})$. Moreover, since $j_1 \geq 2$, it follows from Features (f.2) and (f.5) that $t_{i_1,j_1} \leq I^-_{i_1,j_1} + \mu\Delta = I^+_{i_1,j_1} - 2\Delta$. Thus, we have $t_{i_2,j_2} - t_{i_1,j_1} \geq 2\Delta$. Therefore, the two reference periods cannot intersect.

For Case IV, without loss of generality, suppose $j_1 = 1$ and $j_2 \geq 2$. According to Feature (f.4), $t_{i_1,j_1} = I^-_{i_1,j_1}$. If $i_1 < i_2$, it follows that $I^-_{i_1,j_1} = I^-_{i_1} \leq I^-_{i_2}$. Since $j_2 \geq 2$, Feature (f.3) implies that $t_{i_2,j_2} \geq I^-_{i_2,j_2} \geq I^-_{i_2} + 2\Delta$. Therefore, $t_{i_2,j_2} - t_{i_1,j_1} = t_{i_2,j_2} - I^-_{i_1,j_1} \geq t_{i_2,j_2} - I^-_{i_2} \geq 2\Delta$. If $i_1 > i_2$, for the case where $b^\dagger(I_{i_1,j_1}) \neq b^\dagger(I_{i_2,j_2})$, the two reference periods must not intersect according to the definition. For the case where $b^\dagger(I_{i_1,j_1}) = b^\dagger(I_{i_2,j_2})$, it must hold that $t_{i_1,j_1} \geq I^+_{i_2} \geq I^+_{i_2,j_2}$. This is because otherwise, the reference bin of I_{i_1,j_1} would be b_{i_2}, which cannot be the same as $b^\dagger(I_{i_2,j_2})$. Since $j_2 \geq 2$, based on Features (f.2) and (f.5), we have $t_{i_2,j_2} \leq I^-_{i_2,j_2} + \mu\Delta = I^+_{i_2,j_2} - 2\Delta$. As a result, $t_{i_1,j_1} - t_{i_2,j_2} \geq 2\Delta$. Thus, the two references periods cannot intersect. □

Now, we examine Case V. First, we have the following lemma.

LEMMA 2. *In Case V, suppose $i_1 < i_2$, if the reference periods of I_{i_1,j_1} and I_{i_2,j_2} intersect, the length of I_{i_1,j_1} must be shorter than 2Δ.*

PROOF. In Case V, according to Feature (f.4), we have $t_{i_1,j_1} = I^-_{i_1,j_1}$ and $t_{i_2,j_2} = I^-_{i_2,j_2}$. Assume on the contrary that the length of I_{i_1,j_1} is equal to or longer than 2Δ, i.e., $I^+_{i_1,j_1} - I^-_{i_1,j_1} \geq 2\Delta$. If the two reference periods of I_{i_1,j_1} and I_{i_2,j_2} intersect, we have $b^\dagger(I_{i_1,j_1}) = b^\dagger(I_{i_2,j_2})$. This implies $t_{i_2,j_2} \geq I^+_{i_1} \geq I^+_{i_1,j_1}$ because otherwise, the reference bin of I_{i_2,j_2} would be b_{i_1}, which cannot be the same as $b^\dagger(I_{i_1,j_1})$. Since $I^+_{i_1,j_1} - t_{i_1,j_1} = I^+_{i_1,j_1} - I^-_{i_1,j_1} \geq 2\Delta$, it follows that $t_{i_2,j_2} - t_{i_1,j_1} \geq 2\Delta$, which contradicts that the two reference periods intersect. Thus, the length of I_{i_1,j_1} must be shorter than 2Δ. □

Based on the above analysis, the intersection between reference periods can only happen in Case V, i.e., among the reference periods of $I_{i,1}$'s. We divide $\mathcal{I}^L$ into two subsets: $\mathcal{I}^L_I$ and $\mathcal{I}^L_U$. $\mathcal{I}^L_I$ contains the periods in $\mathcal{I}^L$ whose reference periods intersect with at least one other reference period. $\mathcal{I}^L_U$ contains the periods in $\mathcal{I}^L$ whose reference periods do not intersect with any other reference period. It is apparent that $\mathcal{I}^L_I \cap \mathcal{I}^L_U = \emptyset$ and

$$len(\mathcal{I}^L) = len(\mathcal{I}^L_I) + len(\mathcal{I}^L_U) \tag{9}$$

For any period $I_{i,j} \in \mathcal{I}^L_I$, we must have $j = 1$. Consider two different periods $I_{i,1} \in \mathcal{I}^L_I$ and $I_{k,1} \in \mathcal{I}^L_I$. Suppose $i < k$. If their reference periods intersect, we call $I_{i,1}$ the "front-intersect" period of $I_{k,1}$ and call $I_{k,1}$ the "back-intersect" period of $I_{i,1}$. We then have the following lemma.

LEMMA 3. *For each period $I_{i,1} \in \mathcal{I}^L_I$, there is at most one "front-intersect" period and at most one "back-intersect" period of $I_{i,1}$.*

PROOF. Assume on the contrary that there are two "back-intersect" periods of $I_{i,1}$, which are denoted by $I_{k,1}$ and $I_{h,1}$. It follows that $i < k$ and $i < h$. Without loss of generality, suppose $k < h$. Then, we have $t_{h,1} \geq I^+_k$ (otherwise, the reference bin of $I_{h,1}$ would be b_k and cannot be the same as $b^\dagger(I_{i,1})$) and $t_{k,1} \geq I^+_i$ (otherwise, the reference bin of $I_{k,1}$ would be b_i and cannot be the same as $b^\dagger(I_{i,1})$). According to Feature (f.4), $t_{i,1} = I^-_{i,1}$ and $t_{k,1} = I^-_{k,1}$. Since the minimum item interval length is Δ, we have $I^+_k \geq I^-_k + \Delta = I^-_{k,1} + \Delta$ and $I^+_i \geq I^-_i + \Delta = I^-_{i,1} + \Delta$. Therefore, $t_{h,1} - t_{i,1} \geq I^+_k - t_{i,1} \geq \Delta + I^-_{k,1} - t_{i,1} = \Delta + t_{k,1} - t_{i,1} \geq \Delta + I^+_i - t_{i,1} = \Delta + I^+_i - I^-_{i,1} \geq 2\Delta$. It contradicts the assumption that $I_{h,1}$ is a "back-intersect" period of $I_{i,1}$. Thus, $I_{i,1}$ can have at most one "back-intersect" period. The proof for the "front-intersect" case is similar. □

Next, we construct pairs for the periods in $\mathcal{I}^L_I$ according to the following rule. Consider each period $I_{i,1} \in \mathcal{I}^L_I$ in the

ascending order of i. If $I_{i,1}$ has not been added into any pair and $I_{i,1}$ has a "back-intersect" period (denoted by $I_{i',1}$), we construct a pair $(I_{i,1}, I_{i',1})$. We name the pair as a *joint-period* composed of $(I_{i,1}, I_{i',1})$ (where $i < i'$), and define the reference period of the joint-period as the reference period of $I_{i,1}$, i.e., $p^\dagger(I_{i,1})$. Note that a period in $\mathcal{I}_I^L$ that has no "back-intersect" period might not be added into any pair. We name such period as a *single period*. An example of the pairing process is shown in Figure 7.

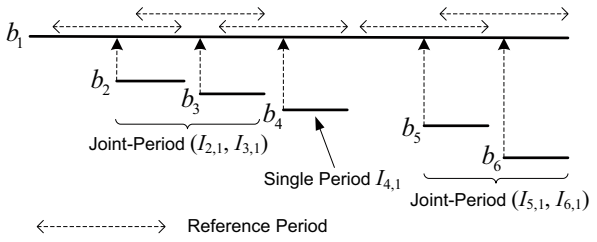

Figure 7: An example of pairing

LEMMA 4. *The reference periods of all the joint-periods and single-periods do not intersect.*

PROOF. We first show that the reference periods of any two joint-periods do not intersect. Assume on the contrary that the reference period of a joint-period composed of $(I_{i,1}, I_{i',1})$ (where $i < i'$) intersects with the reference period of another joint-period composed of $(I_{k,1}, I_{k',1})$ (where $k < k'$). Without loss of generality, suppose $i < k$. Then, $I_{i,1}$ would have two "back-intersect" periods $I_{k,1}$ and $I_{i',1}$, which contradicts Lemma 3.

We then show that the reference period of any single period does not intersect with the reference period of any joint-period. Assume on the contrary that the reference period of a single period $I_{k,1}$ intersects with the reference period of a joint-period composed of $(I_{i,1}, I_{i',1})$ (where $i < i'$). If $k < i$, it implies that $I_{k,1}$ has a "back-intersect" period $I_{i,1}$, which contradicts our pairing rule. If $k > i$, it implies that $I_{i,1}$ has two "back-intersect" periods $I_{k,1}$ and $I_{i',1}$, which contradicts Lemma 3.

Furthermore, it is obvious that the reference periods of any two single periods do not intersect. Therefore, in summary, the reference periods of all the joint-periods and single periods do not intersect. □

Note that according to Feature (f.1) and Lemma 2, the total length of the two periods constituting a joint-period should be shorter than $(\mu + 4)\Delta + 2\Delta = (\mu + 6)\Delta$. Let $\mathcal{I}_I^L(J)$ denote the set of all the joint-periods, and $|\mathcal{I}_I^L(J)|$ be the number of joint-periods in $\mathcal{I}_I^L(J)$. Let $\mathcal{I}_I^L(S)$ denote the set of all the single periods, and $|\mathcal{I}_I^L(S)|$ be the number of single periods in $\mathcal{I}_I^L(S)$. It follows that $len(\mathcal{I}_I^L) \le |\mathcal{I}_I^L(J)| \cdot (\mu + 6)\Delta + |\mathcal{I}_I^L(S)| \cdot (\mu + 4)\Delta$.

Let $|\mathcal{I}_U^L|$ be the number of periods in $\mathcal{I}_U^L$. Since each period in $\mathcal{I}_U^L$ is at most $(\mu + 4)\Delta$ long (Feature (f.1)), we have $len(\mathcal{I}_U^L) \le |\mathcal{I}_U^L| \cdot (\mu + 4)\Delta$. Therefore, it follows from (9) that

$$\begin{aligned} len(\mathcal{I}^L) &\le |\mathcal{I}_I^L(J)| \cdot (\mu + 6)\Delta + |\mathcal{I}_I^L(S)| \cdot (\mu + 4)\Delta \\ &\quad + |\mathcal{I}_U^L| \cdot (\mu + 4)\Delta \\ &\le \big(|\mathcal{I}_I^L(J)| + |\mathcal{I}_I^L(S)| + |\mathcal{I}_U^L|\big) \cdot (\mu + 6)\Delta \end{aligned}$$

According to (6) and (7), we have

$$\begin{aligned} FF_{total}(\mathcal{R}) &\le C \cdot \big(|\mathcal{I}_I^L(J)| + |\mathcal{I}_I^L(S)| + |\mathcal{I}_U^L|\big) \cdot (\mu + 6)\Delta \\ &\quad + C \cdot span(\mathcal{R}) \end{aligned} \tag{10}$$

On the other hand, according to the previous analysis, the reference periods of all the periods in $\mathcal{I}_U^L$, the joint-periods in $\mathcal{I}_I^L(J)$ and the single periods in $\mathcal{I}_I^L(S)$ do not intersect. Thus, based on (8), the total resource demand $u(\mathcal{R})$ satisfies

$$u(\mathcal{R}) \ge \big(|\mathcal{I}_I^L(J)| + |\mathcal{I}_I^L(S)| + |\mathcal{I}_U^L|\big) \cdot \Big(W - \frac{W}{k}\Big) \cdot \Delta \tag{11}$$

According to (10) and (11), we have

$$FF_{total}(\mathcal{R}) \le \frac{C \cdot (\mu + 6)}{W \cdot (1 - \frac{1}{k})} \cdot u(\mathcal{R}) + C \cdot span(\mathcal{R}) \tag{12}$$

Moreover, we have the following bounds for $OPT_{total}(\mathcal{R})$: $OPT_{total}(\mathcal{R}) \ge \frac{C \cdot u(\mathcal{R})}{W}$ (this is derived by assuming that no bin capacity is wasted at any time) and $OPT_{total}(\mathcal{R}) \ge C \cdot span(\mathcal{R})$ (this is derived from the fact that at least one bin must be in use at any time when there is at least one active item). It follows that

$$FF_{total}(\mathcal{R}) \le \left(\frac{k}{k-1} \cdot \mu + \frac{6k}{k-1} + 1\right) \cdot OPT_{total}(\mathcal{R})$$

Therefore, we have the following result.

THEOREM 4. *For the MinTotal DBP problem, for any item list $\mathcal{R}$, if the item size $s(r) < \frac{W}{k}$ ($k > 1$ is a constant) for all the items $r \in \mathcal{R}$, the total cost of First Fit packing is at most $(\frac{k}{k-1} \cdot \mu + \frac{6k}{k-1} + 1) \cdot OPT_{total}(\mathcal{R})$.* □

Next, we consider the general case for First Fit packing. We follow the above analysis for Theorem 4. Let $b_1, b_2, ..., b_m$ denote the bins used in First Fit packing to pack an item list $\mathcal{R}$. For each bin b_i, its usage period I_i is divided into two parts I_i^L and I_i^R. Then, we perform split and mergence on all I_i^L's to produce the set of sub-periods $\mathcal{I}^L$. $\mathcal{I}^L$ is further divided into two subsets $\mathcal{I}_I^L$ and $\mathcal{I}_U^L$ based on the intersections between periods. From $\mathcal{I}_I^L$, we construct the set of joint-periods $\mathcal{I}_I^L(J)$ and the set of single periods $\mathcal{I}_I^L(S)$. According to (10), the total cost of First Fit packing is at most

$$C \cdot \big(|\mathcal{I}_I^L(J)| + |\mathcal{I}_I^L(S)| + |\mathcal{I}_U^L|\big) \cdot (\mu + 6)\Delta + C \cdot span(\mathcal{R}) \tag{13}$$

where $|\mathcal{I}_I^L(J)|$, $|\mathcal{I}_I^L(S)|$ and $|\mathcal{I}_U^L|$ are the sizes of sets $\mathcal{I}_I^L(J)$, $\mathcal{I}_I^L(S)$ and $\mathcal{I}_U^L$ respectively.

Consider a period $I_{i,j}$ in $\mathcal{I}_U^L \cup \mathcal{I}_I^L(S)$ or a joint-period composed of $(I_{i,j}, I_{i',j})$ in $\mathcal{I}_I^L(J)$. Recall that its reference period $p^\dagger(I_{i,j})$ is the period $[t_{i,j} - \Delta, t_{i,j} + \Delta]$ associated with the reference bin $b^\dagger(I_{i,j})$. Let $p^\ddagger(I_{i,j})$ denote the same period $[t_{i,j} - \Delta, t_{i,j} + \Delta]$ associated with bin b_i. We refer to $p^\ddagger(I_{i,j})$ as the *auxiliary period* of $I_{i,j}$. Figure 8 shows an example of auxiliary periods.

In the above analysis for Theorem 4, we have shown that a new item must be packed into b_i at time $t_{i,j}$. According to the First Fit packing algorithm, after this item is packed, the total level of bins b_i and $b^\dagger(I_{i,j})$ should be higher than W (the bin capacity). Otherwise, the new item would have been packed into $b^\dagger(I_{i,j})$ instead. Moreover, since Δ is the minimum item interval length, all the items in bin b_i at time $t_{i,j}$ must reside in the system for at least Δ time during the auxiliary period $p^\ddagger(I_{i,j})$, and all the items in bin $b^\dagger(I_{i,j})$

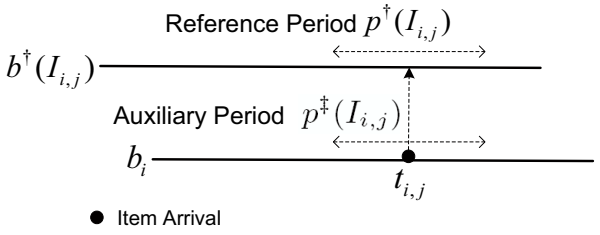

Figure 8: An example of auxiliary periods

at time $t_{i,j}$ must reside in the system for at least Δ time during the reference period $p^\dagger(I_{i,j})$. It follows that the total resource demand of the items in bin b_i over $p^\ddagger(I_{i,j})$ and the items in bin $b^\dagger(I_{i,j})$ over $p^\dagger(I_{i,j})$ satisfies

$$u(p^\dagger(I_{i,j})) + u(p^\ddagger(I_{i,j})) \geq W \cdot \Delta \tag{14}$$

According to the previous analysis, all the reference periods of those in $\mathcal{I}_U^L \cup \mathcal{I}_I^L(J) \cup \mathcal{I}_I^L(S)$ do not intersect with each other. Next, we examine the intersections among the auxiliary periods.

LEMMA 5. *The auxiliary periods of two different periods I_{i_1,j_1} and I_{i_2,j_2} do not intersect.*

PROOF. If $i_1 \neq i_2$, the auxiliary periods $p^\ddagger(I_{i_1,j_1})$ and $p^\ddagger(I_{i_2,j_2})$ do not intersect since they are associated with different bins. If $i_1 = i_2$, without loss of generality, suppose $j_1 < j_2$. According to the analysis on Cases I and II in Lemma 1, it must hold that $t_{i_2,j_2} - t_{i_1,j_1} \geq 2\Delta$. Therefore, $p^\ddagger(I_{i_1,j_1})$ and $p^\ddagger(I_{i_2,j_2})$ cannot intersect. □

Since all the reference periods of those in $\mathcal{I}_U^L \cup \mathcal{I}_I^L(J) \cup \mathcal{I}_I^L(S)$ do not intersect and all the auxiliary periods of those in $\mathcal{I}_U^L \cup \mathcal{I}_I^L(J) \cup \mathcal{I}_I^L(S)$ do not intersect either, any time point associated with each bin can be shared by at most one reference period and one auxiliary period. Therefore, it follows from (14) that

$$u(\mathcal{R}) \geq \frac{1}{2}\big(|\mathcal{I}_I^L(J)| + |\mathcal{I}_I^L(S)| + |\mathcal{I}_U^L|\big) \cdot W \cdot \Delta \tag{15}$$

According to (13) and (15), we have

$$\begin{aligned} FF_{total}(\mathcal{R}) &\leq \frac{2C \cdot (\mu+6)}{W} \cdot u(\mathcal{R}) + C \cdot span(\mathcal{R}) \\ &\leq 2 \cdot (\mu+6) \cdot OPT_{total}(\mathcal{R}) + OPT_{total}(\mathcal{R}) \\ &\leq (2\mu+13) \cdot OPT_{total}(\mathcal{R}) \end{aligned}$$

Therefore, we have the following result.

THEOREM 5. *For the MinTotal DBP problem, First Fit packing has a competitive ratio of $2\mu + 13$.* □

4.4 A Modified First Fit Packing Algorithm

The analysis in the previous section shows that the competitive ratio of First Fit packing for the MinTotal DBP problem is much related to the item sizes. Inspired by Theorem 3 and Theorem 4, we propose a new Modified First Fit packing algorithm that can achieve improved competitive ratios.

- **Modified First Fit (MFF)**: Define a variable $k > 1$. The items with sizes equal to or larger than $\frac{W}{k}$ are classified as large items. The items with sizes smaller than $\frac{W}{k}$ are classified as small items. Modified First Fit packing uses the classical First Fit algorithm to pack the large items and the small items separately.

Given an item list $\mathcal{R}$, let $\mathcal{R}^L$ denote the set of all the large items and $\mathcal{R}^S$ denote the set of all the small items. Then, $s(r) \geq \frac{W}{k}$ for all $r \in \mathcal{R}^L$, and $s(r) < \frac{W}{k}$ for all $r \in \mathcal{R}^S$. According to (3) in the proof of Theorem 3, we have

$$MFF_{total}(\mathcal{R}^L) \leq k \cdot \frac{C \cdot u(\mathcal{R}^L)}{W}$$

According to (12) in the analysis of Theorem 4, we have

$$MFF_{total}(\mathcal{R}^S) \leq \frac{C \cdot (\mu+6)}{W \cdot (1-\frac{1}{k})} \cdot u(\mathcal{R}^S) + C \cdot span(\mathcal{R}^S)$$

Note that $u(\mathcal{R}^L) \leq u(\mathcal{R})$, $u(\mathcal{R}^S) \leq u(\mathcal{R})$, and $span(\mathcal{R}^S) \leq span(\mathcal{R})$. Thus, it follows that

$$\begin{aligned} &MFF_{total}(\mathcal{R}) = MFF_{total}(\mathcal{R}^L) + MFF_{total}(\mathcal{R}^S) \\ &\leq k \cdot \frac{C \cdot u(\mathcal{R}^L)}{W} + \frac{C \cdot (\mu+6)}{W \cdot (1-\frac{1}{k})} \cdot u(\mathcal{R}^S) + C \cdot span(\mathcal{R}^S) \\ &\leq \max\left\{k, \frac{\mu+6}{1-\frac{1}{k}}\right\} \cdot \frac{C \cdot u(\mathcal{R})}{W} + C \cdot span(\mathcal{R}) \end{aligned}$$

If the max/min item interval length ratio μ is not known, we can set $k = 8$ in Modified First Fit packing. In this case,

$$\max\left\{k, \frac{\mu+6}{1-\frac{1}{k}}\right\} = \max\left\{8, \frac{8}{7}\mu + \frac{48}{7}\right\}$$

Since $\mu \geq 1$, we have $\frac{8}{7}\mu + \frac{48}{7} \geq 8$. Therefore,

$$\begin{aligned} MFF_{total}(\mathcal{R}) &\leq \left(\frac{8}{7}\mu + \frac{48}{7}\right) \cdot \frac{C \cdot u(\mathcal{R})}{W} + C \cdot span(\mathcal{R}) \\ &\leq \left(\frac{8}{7}\mu + \frac{55}{7}\right) \cdot OPT_{total}(\mathcal{R}) \end{aligned}$$

Therefore, when μ is not known, Modified First Fit packing can achieve a competitive ratio of $\frac{8}{7}\mu + \frac{55}{7}$ for the MinTotal DBP problem.

In certain applications such as cloud gaming, it is possible to estimate the max/min item interval length ratio μ according to the statistics of historical playing data. If μ is known, it can be derived that when $k = \mu+7$, $\max\left\{k, \frac{\mu+6}{1-\frac{1}{k}}\right\}$ achieves the smallest value which is given by $\mu + 7$. Therefore, we have

$$\begin{aligned} MFF_{total}(\mathcal{R}) &\leq (\mu+7) \cdot \frac{C \cdot u(\mathcal{R})}{W} + C \cdot span(\mathcal{R}) \\ &\leq (\mu+8) \cdot OPT_{total}(\mathcal{R}) \end{aligned}$$

Therefore, when μ is known, Modified First Fit packing can achieve a competitive ratio of $\mu + 8$ for the MinTotal DBP problem.[2]

5. CONCLUSIONS

In this paper, we have studied the MinTotal Dynamic Bin Packing problem that aims to minimize the total cost of the bins used over time. We have analyzed the competitive ratios of the commonly used First Fit, Best Fit and Any Fit packing algorithms for this problem. It is shown that the competitive ratio of Any Fit packing cannot be better than

[2]Since μ is known, Modified First Fit packing in this case is a semi-online algorithm.

the max/min item interval length ratio μ. Best Fit packing has no bounded competitive ratio even for any given μ. In a special case where all the item sizes are smaller than $\frac{W}{k}$ (W is the bin capacity and $k > 1$ is a constant), First Fit packing has a competitive ratio of $\frac{k}{k-1}\mu + \frac{6k}{k-1} + 1$. For the general case, First Fit packing has a competitive ratio of $2\mu+13$. We have also proposed a Modified First Fit packing algorithm which classifies and assigns items according to their sizes. We have shown that Modified First Fit packing can achieve a competitive ratio of $\frac{8}{7}\mu + \frac{55}{7}$ when μ is not known and can achieve a competitive ratio of $\mu + 8$ when μ is known. In the future work, we would like to further investigate the constrained Dynamic Bin Packing problem in which each item is allowed to be assigned to only a subset of bins to cater for the interactivity constraints of dispatching playing requests among distributed clouds in cloud gaming.

6. ACKNOWLEDGMENTS

This research is supported by Multi-plAtform Game Innovation Centre (MAGIC), funded by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority.

7. REFERENCES

[1] Distribution and monetization strategies to increase revenues from cloud gaming. `http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf`.

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

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

[4] Streammygame. `http://www.streammygame.com/smg/index.php`.

[5] 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, Volume 6534)*, pages 25–36, 2011.

[6] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):735–744, Sept. 2001.

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

[8] 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.

[9] 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.

[10] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 1269–1272. ACM, 2011.

[11] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Survey and classification. *Handbook of Combinatorial Optimization (second ed.)*, pages 455–531, 2013.

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

[13] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. *Approximation Algorithms for NP-hard Problems*, pages 46–93, 1997.

[14] 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 23th IEEE International Symposium on Parallel and Distributed Processing*, pages 1–12. IEEE, 2009.

[15] G. Galambos and G. J. Woeginger. On-line bin packingąła restricted survey. *Zeitschrift für Operations Research*, 42(1):25–45, 1995.

[16] M. R. Gary and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[17] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen. Gaminganywhere: an open cloud gaming system. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 36–47. ACM, 2013.

[18] Z. Ivkovic and E. L. Lloyd. Fully dynamic algorithms for bin packing: Being (mostly) myopic helps. *SIAM Journal on Computing*, 28(2):574–611, 1998.

[19] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. Joint vm placement and routing for data center traffic engineering. In *Proceedings of the 31th IEEE International Conference on Computer Communications*, pages 2876–2880. IEEE, 2012.

[20] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in Operations Research and Management Science*, 4:445–522, 1993.

[21] 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*, pages 238–248. IEEE, 2012.

[22] P. E. Ross. Cloud computing's killer app: Gaming. *IEEE Spectrum*, 46(3):14–14, 2009.

[23] S. S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

[24] C. Sharon, W. Bernard, G. Simon, C. Rosenberg, et al. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proceedings of the 11th ACM Annual Workshop on Network and Systems Support for Games*, 2012.

[25] A. Stolyar. An infinite server system with general packing constraints. *arXiv preprint arXiv:1205.4271*, 2012.

[26] C. Zhang, Z. Qi, J. Yao, M. Yu, and H. Guan. vgasa: Adaptive scheduling algorithm of virtualized gpu resource in cloud gaming. *IEEE Transactions on Parallel and Distributed Systems*, accepted to appear, 2014.