Bounds for scheduling jobs on grid processors

Boyar, Joan; Ellen, Faith

Published in:
Space-Efficient Data Structures, Streams, and Algorithms

DOI:
10.1007/978-3-642-40273-9_2

Publication date:
2013

Document version:
Accepted manuscript

Document license:
Unspecified

Citation for pulished version (APA):
Boyar, J., & Ellen, F. (2013). Bounds for scheduling jobs on grid processors. In A. Brodnik, A. Lopez-Ortiz, V. Raman, & A. Viola (Eds.), *Space-Efficient Data Structures, Streams, and Algorithms: Papers in Honor of J. Ian Munro, on the Occasion of His 66th Birthday* (pp. 12-26). Springer. https://doi.org/10.1007/978-3-642-40273-9_2

Bounds for Scheduling Jobs on Grid Processors *

Joan Boyar[1] and Faith Ellen[2]

[1] University of Southern Denmark, joan@imada.sdu.dk
[2] University of Toronto, faith@cs.toronto.edu

Abstract. In the Grid Scheduling problem, there is a set of jobs each with a positive integral memory requirement. Processors arrive in an online manner and each is assigned a maximal subset of the remaining jobs such that the sum of the memory requirements of those jobs does not exceed the processor's memory capacity. The goal is to assign all the jobs to processors so as to minimize the sum of the memory capacities of the processors that are assigned at least one job. Previously, a lower bound of $\frac{5}{4}$ on the competitive ratio of this problem was achieved using jobs of size S and $2S-1$. For this case, we obtain matching upper and lower bounds, which vary depending on the ratio of the number of small jobs to the number of large jobs.

1 Introduction

The Grid is a computing environment comprised of various processors which arrive at various times and to which jobs can be assigned. We consider the problem of scheduling a set of jobs, each with a specific memory requirement. When a processor arrives, it announces its memory capacity. Jobs are assigned to the processor so that the sum of the requirements of its assigned jobs does not exceed its capacity. In this way, the processor can avoid the expensive costs of paging when it executes the jobs. The charge for a set of jobs is (proportional to) the memory capacities of the processors to which they are assigned. There is no charge for processors whose capacities are too small for any remaining jobs. The goal is to assign all the jobs to processors in a manner that minimizes the total charge.

The Grid Scheduling problem was motivated by a problem in bioinformatics in which genomes are compared to a very large database of DNA sequences to identify regions of interest [2]. In this application, an extremely large problem is divided into a set of independent jobs with varying memory requirements. The

* This research was supported in part by the Danish Council for Independent Research, Natural Sciences (FNU), the VELUX Foundation, and the Natural Science and Engineering Research Council of Canada (NSERC). Parts of this work were carried out while Joan Boyar was visiting University of Waterloo and University of Toronto and while Faith Ellen was visiting University of Southern Denmark.

Grid Scheduling problem can also be rephrased as a bin packing problem for a given set of items, using variable-sized bins, which arrive one by one, in an online manner. In contrast, usual bin packing problems assume the bins are given and the items arrive online.

The Grid Scheduling problem was introduced by Boyar and Favrholdt in [1]. They gave an algorithm to solve this problem with competitive ratio $\frac{13}{7}$. This solved an open question in [8], which considered a similar problem. They also proved that the competitive ratio of any algorithm for this problem is at least $\frac{5}{4}$. The lower bound proof uses $s = 2\ell$ items of size S and ℓ items of size $L = 2S - 1$, where $S > 1$ is an integer and $M = 2L - 1$ is the maximum bin size. If $S = 1$, then $L = 2S - 1 = 1$, so all items have the same size, making the problem uninteresting. Likewise, if there is no bound on the maximum bin size, the problem is uninteresting, because the competitive ratio is unbounded: Once enough bins for an optimal packing have arrived, an adversary can send bins of arbitrarily large size, which the algorithm would be forced to use.

In many applications of bin packing, there are only a small number of different item sizes. A number of papers have considered the problem of packing a sequence of items of two different sizes in bins of size 1 in an online matter. In particular, there is a lower bound of 4/3 on the competitive ratio [6, 4] and a matching upper bound [4]. When both item sizes are bounded above by $1/k$, the competitive ratio can be improved to $\frac{(k+1)^2}{k^2+k+1}$ [3].

In this paper, we consider the Restricted Grid Scheduling problem, a version of the Grid Scheduling problem where the input contains exactly s items of size S and ℓ items of size $L = 2S - 1$ and the maximum bin size is $M = 2L - 1$, which includes the scenarios used in the $\frac{5}{4}$ lower bound. Two natural questions arise for this problem:

1. Is there an algorithm matching the $\frac{5}{4}$ lower bound on the competitive ratio or can this lower bound be improved?
2. What is the competitive ratio for the problem when there are initially s items of size S and ℓ items of size L, when the ratio of s to ℓ is arbitrary, rather than fixed to 2?

We obtain matching upper and lower bounds on the competitive ratio of the Restricted Grid Scheduling problem.

Theorem 1. *For unbounded S, the competitive ratio of the Restricted Grid Scheduling problem is*

$$\begin{cases} 1 + \frac{1}{2+s/\ell} & \text{if } \ell \le s/2, \\ 1 + \frac{1}{4} & \text{if } s/2 < \ell \le 5s/6, \text{ and} \\ 1 + \frac{2}{3+6\ell/s} & \text{if } 5s/6 < \ell. \end{cases}$$

We begin with some preliminaries, including a formal definition of the Grid Scheduling problem and some properties of optimal packings. In Section 3, we prove the lower bound on the competitive ratio of the Restricted Grid Scheduling problem. Then, in Section 4, we present properties of optimal packings for the Restricted Grid Scheduling problem. Some of these provide motivation for our design, while others are important for the analysis. We show why a few simple algorithms are not optimal in Section 5. Our algorithm, 2-Phase-Packer, appears in Section 6, together with part of the analysis. We conclude with some open questions. The omitted proofs and the rest of the analysis are in the full paper.

2 Preliminaries

The competitive ratio [7, 5] of an on-line algorithm is the worst-case ratio of the on-line performance to the optimal off-line performance, up to an additive constant. More precisely, for a set I of items (or, equivalently, a multi-set of item sizes), a sequence σ of bins, and an algorithm $\mathbb{A}$ for the Grid Scheduling problem, let $\mathbb{A}(I, \sigma)$ denote the total size of all the bins used by $\mathbb{A}$ when packing I in the sequence σ of bins. Then, the *competitive ratio* $\mathrm{CR}_{\mathbb{A}}$ of $\mathbb{A}$ is

$$\mathrm{CR}_{\mathbb{A}} = \inf \{c \mid \exists b, \forall I, \forall \sigma, \mathbb{A}(I, \sigma) \leq c \cdot \mathrm{OPT}(I, \sigma) + b\},$$

where OPT denotes an optimal off-line algorithm. For specific choices of families of sets I_n and sequences σ_n, the performance ratios, $\frac{\mathbb{A}(I_n, \sigma_n)}{\mathrm{OPT}(I_n, \sigma_n)}$, can be used to prove a lower bound on the competitive ratio of $\mathbb{A}$.

Given a set of items and a sequence of bins, each with a size in $\{1, \ldots, M\}$, the goal of the Grid Scheduling problem is to pack the items in the bins so that the sum of the sizes of the items packed in each bin is at most the size of the bin and the sum of the sizes of bins used is minimized. The bins in the sequence arrive one at a time and each must be packed before the next bin arrives, without knowledge of the sizes of any future bins. If a bin is at least as large as the smallest unpacked item, it must be packed with at least one item. There is no charge for a bin that is smaller than this. It is assumed that enough sufficiently large bins arrive so that any algorithm eventually packs all items. For example, it suffices that every sequence ends with enough bins of size M to pack every item one per bin.

Given a set of items and a sequence of bins, a *partial packing* is an assignment of some of the items to bins such that the sum of the sizes of the items assigned to each bin is at most the size of the bin. A *packing* is a partial packing that assigns every item to a bin. If p is a packing of a set of items I into a sequence of bins σ and p' is a packing of a disjoint set of items I' into a sequence of bins σ', then we use pp' to denote the packing of $I \cup I'$ into the sequence of bins $\sigma\sigma'$, where each item in I is assigned to the same bin as in p and each item in I' is assigned to the same bin as in p'.

If every item has size at least S, there is no loss of generality in assuming that every bin has size at least S. This is because no packing can use a bin of size less than S.

A packing is *valid* if every bin that can be used is used. In other words, in a valid packing, if a bin is empty, then all remaining items are larger than the bin.

A bin in a packing p for σ is *wasteful* if its empty space is at least the size of the smallest remaining item. A packing is *thrifty* if it contains no wasteful bins. Since a packing is valid if and only if it has no wasteful empty bin, every thrifty packing is valid.

A packing is *optimal* if it is valid and the sum of the sizes of the bins it uses is at least as small as any other valid packing.

Lemma 1. *For any sequence of bins and any optimal packing pp' of these bins, there is an optimal packing pq that uses the same set of bins, in which q has no wasteful bins.*

Taking p to be empty in Lemma 1 yields the following result.

Corollary 1. *For any optimal packing of a sequence of bins, there is an optimal thrifty packing of that sequence using the same set of bins.*

3 Lower Bounds

Theorem 2. *For unbounded S, any algorithm for the Restricted Grid Scheduling problem has competitive ratio at least*

$$\begin{cases} 1 + \frac{1}{2+s/\ell} & \text{if } \ell \le s/2, \\ 1 + \frac{1}{4} & \text{if } s/2 < \ell \le 5s/6, \text{ and} \\ 1 + \frac{2}{3+6\ell/s} & \text{if } 5s/6 < \ell. \end{cases}$$

Proof. Consider an instance of the Grid Scheduling problem in which there are s items of size S, ℓ items of size $L = 2S - 1$, and maximum bin size $M = 2L - 1$. We start with the case when $\ell \le s/2$ and then handle the case when $\ell > s/2$. In both cases, we consider two subcases, depending on how the algorithm packs the first batch of bins.

Case I: $\ell \le s/2$.

The adversary begins by giving ℓ bins of size $2S$. In each of these bins, the algorithm must pack either two items of size S or one item of size L. Let $0 \le k \le \ell$ be the number of these bins in which the algorithm packs two items of size S. Then the algorithm has $s - 2k$ items of size S and k items of size L left to pack.

Case I.1: $k \leq \ell/2$.
Next, the adversary gives $s - 2\ell$ bins of size S, followed by 2ℓ bins of size L. The algorithm must pack one item of size S in each bin of size S and must use one bin of size L for each of the remaining $s - 2k - (s - 2\ell) = 2(\ell - k)$ items of size S and k items of size L. The total cost incurred by the algorithm is $\ell \cdot 2S + (s - 2\ell) \cdot S + (2\ell - k) \cdot L = s \cdot S + 2\ell \cdot L - k \cdot L \geq s \cdot S + 3\ell \cdot L/2$.

For this sequence, OPT packs two items of size S in each of the ℓ bins of size $2S$, one item of size S in each of the $s - 2\ell$ bins of size S, and one item of size L in each of the next ℓ bins of size L, for total cost $s \cdot S + \ell \cdot L$. Thus, the performance ratio of the algorithm is at least

$$\begin{aligned}\frac{s \cdot S + 3\ell \cdot L/2}{s \cdot S + \ell \cdot L} &= 1 + \frac{\ell \cdot L}{2\ell \cdot L + 2s \cdot S} = 1 + \frac{\ell}{2\ell + s + s/L} \\ &= 1 + \frac{1}{2 + \frac{s}{\ell} + \frac{s/\ell}{2S-1}} \to 1 + \frac{1}{2 + s/\ell} \text{ as } S \to \infty.\end{aligned}$$

Case I.2: $k > \ell/2$.
Next, the adversary gives s bins of size S, followed by ℓ bins of size M. The algorithm packs one item of size S in the first $s - 2k \geq 2\ell - 2k \geq 0$ of these bins, using up all its items of size S. It discards the remaining $2k$ bins of size S, because it has no remaining elements that are small enough to fit in them. Then the algorithm packs its remaining k items of size L into k bins of size $M = 4S - 3$. The total cost incurred by the algorithm is

$$\begin{aligned}&\ell \cdot 2S + (s - 2k) \cdot S + k \cdot (4S - 3) \\ &= (2\ell + s) \cdot S + k \cdot (2S - 3) \\ &> (2\ell + s) \cdot S + \ell \cdot (S - 3/2).\end{aligned}$$

For this sequence, OPT packs one item of size L in each of the ℓ bins of size $2S$ and one item of size S in each of the next s bins. The total cost used by OPT is $\ell \cdot 2S + s \cdot S$. Thus, the performance ratio of the algorithm is at least

$$\frac{(3\ell + s) \cdot S - 3\ell/2}{(2\ell + s) \cdot S} = 1 + \frac{1 - 3/(2S)}{2 + s/\ell} \to 1 + \frac{1}{2 + s/\ell} \text{ as } S \to \infty.$$

Case II: $\ell > s/2$.

The adversary begins by giving $\lfloor s/2 \rfloor$ bins of size $2S$. In each of these bins, the algorithm must pack either two items of size S or one item of size L. Let $0 \leq k \leq \lfloor s/2 \rfloor$ be the number of these bins in which the algorithm packs two items of size S. Then the algorithm has $s - 2k$ items of size S and $\ell - \lfloor s/2 \rfloor + k$ items of size L left to pack.

Case II.1: $k \leq \lfloor s/2 \rfloor - s/8 - \ell/4 + 1$ or $k \leq \lfloor s/2 \rfloor - s/3 + 1$.
Next, the adversary next gives $\lceil s/2 \rceil - \lfloor s/2 \rfloor$ bins of size S (i.e. one bin of size S if s is odd and no bins of size S if s is even), $\lfloor s/2 \rfloor - k + \ell - 1$ bins of size L, and 1 bin

of size M. Since $(s-2k)+(\ell-\lfloor s/2\rfloor+k) = (\lceil s/2\rceil-\lfloor s/2\rfloor)+(\lfloor s/2\rfloor-k+\ell-1)+1$, the algorithm packs one of its remaining items in each of these bins, so the total cost it incurs is $\lfloor s/2\rfloor \cdot 2S + (\lceil s/2\rceil - \lfloor s/2\rfloor)\cdot S + (\lfloor s/2\rfloor - k + \ell - 1)\cdot L + M = s\cdot S + \ell\cdot L + (\lfloor s/2\rfloor - k + 1)\cdot L - 1$.

If $s \geq 4$, then $\lfloor s/2\rfloor - k \geq \min\{s/8+\ell/4, s/3\} - 1 > s/4 - 1 \geq 0$, so there are at least ℓ bins of size L. Because of the additive constant in the definition of the competitive ratio, the case $s \leq 3$ can be ignored. For this sequence, OPT packs two items of size S in each of the $\lfloor s/2\rfloor$ bins of size $2S$, one item of size S in the bin of size S, if s is odd, and one item of size L in each of the next ℓ bins of size L. Its total cost is $s\cdot S + \ell\cdot L$.

If $k \leq \lfloor s/2\rfloor - s/8 - \ell/4 + 1$, then $(\lfloor s/2\rfloor - k + 1)\cdot L \geq (s/8+\ell/4)\cdot L = S\cdot s/4 - s/8 + L\cdot\ell/4$, so the performance ratio of the algorithm is at least

$$\frac{(s\cdot S + \ell\cdot L)\cdot 5/4 - s/8 - 1}{s\cdot S + \ell\cdot L} = \frac{5}{4} - \frac{s/8+1}{s\cdot S + \ell\cdot(2S-1)} \to \frac{5}{4} \text{ as } S\to\infty.$$

Similarly, if $k \leq \lfloor s/2\rfloor - s/3 + 1$, then $(\lfloor s/2\rfloor - k + 1)\cdot L \geq (s/3)\cdot L$, so the performance ratio of the algorithm is at least

$$1 + \frac{(s/3)\cdot L}{s\cdot S + \ell\cdot L} = 1 + \frac{2 - 1/(s\cdot S)}{3 + 6\ell/s - 3\ell/(s\cdot S)} \to 1 + \frac{2}{3+6\ell/s} \text{ as } S\to\infty.$$

Case II.2: $k > \lfloor s/2\rfloor - s/8 - \ell/4 + 1$ and $k > \lfloor s/2\rfloor - s/3 + 1 > s/6$.
Next, the adversary gives $\max\{s-2k, s-\ell+\lfloor s/2\rfloor\}$ bins of size S, followed by $\min\{2k, \ell-\lfloor s/2\rfloor\}$ bins of size $L+S$, $\max\{\ell-\lfloor s/2\rfloor-2k, 0\}$ bins of size L, and finally k bins of size $M = 4S-3$. The algorithm packs its $s-2k$ items of size S into bins of size S. Since $\min\{2k, \ell-\lfloor s/2\rfloor\}+\max\{\ell-\lfloor s/2\rfloor-2k, 0\}+k = \ell-\lfloor s/2\rfloor+k$, the algorithm packs one item of size L in each bin of size $L+S$, L, and M. The total cost incurred by the algorithm is

$$\begin{aligned}&\lfloor s/2\rfloor\cdot 2S + (s-2k)\cdot S + \min\{2k, \ell-\lfloor s/2\rfloor\}\cdot(L+S)\\ &\quad + \max\{\ell-\lfloor s/2\rfloor-2k, 0\}\cdot L + k\cdot M\\ &= (2\lfloor s/2\rfloor + s)\cdot S + (\ell-\lfloor s/2\rfloor)\cdot L + k\cdot(2S-3) + \min\{2k, \ell-\lfloor s/2\rfloor\}\cdot S.\end{aligned}$$

For this sequence, OPT fills every bin it uses except for the $\lfloor s/2\rfloor$ bins of size $2S$, in which it puts items of size $L = 2S-1$. Therefore, the total cost used by OPT is $s\cdot S + \ell\cdot L + \lfloor s/2\rfloor$.

If $2k \geq \ell - \lfloor s/2\rfloor$, the performance ratio of the algorithm is at least

$$\begin{aligned}&\frac{(2\lfloor s/2\rfloor + s)\cdot S + (\ell-\lfloor s/2\rfloor)\cdot L + k\cdot(2S-3) + (\ell-\lfloor s/2\rfloor)\cdot S}{s\cdot S + \ell\cdot L + \lfloor s/2\rfloor}\\ &> \frac{(\ell + s + \lfloor s/2\rfloor)\cdot S + (\ell-\lfloor s/2\rfloor)\cdot L + (\lfloor s/2\rfloor - s/8 - \ell/4 + 1)\cdot(2S-3)}{s\cdot S + \ell\cdot L + \lfloor s/2\rfloor}\\ &= \frac{5}{4} + \frac{(\lfloor s/2\rfloor - s/2 + 2)\cdot S + \ell + 3s/8 - 13\lfloor s/2\rfloor/4 - 3}{(s+2\ell)\cdot S - \ell + \lfloor s/2\rfloor}\\ &> \frac{5}{4} + \frac{\ell + 3s/8 - 13\lfloor s/2\rfloor/4 - 3}{(s+2\ell)\cdot S - \ell + \lfloor s/2\rfloor} \to \frac{5}{4} \text{ as } S\to\infty.\end{aligned}$$

If $2k \leq \ell - \lfloor s/2 \rfloor$, the performance ratio of the algorithm is at least

$$\begin{aligned}
&\frac{(2\lfloor s/2 \rfloor + s) \cdot S + (\ell - \lfloor s/2 \rfloor) \cdot L + k \cdot (4S - 3)}{s \cdot S + \ell \cdot L + \lfloor s/2 \rfloor} \\
&= 1 + \frac{k \cdot (4S - 3)}{(s + 2\ell) \cdot S - \ell + \lfloor s/2 \rfloor} \\
&> 1 + \frac{(s/6) \cdot (4S - 3)}{(s + 2\ell) \cdot S + \lfloor s/2 \rfloor - \ell} \\
&= 1 + \frac{2 - 3/2S}{3 + 6\ell/s + 3(\lfloor s/2 \rfloor - \ell)/sS} \to 1 + \frac{2}{3 + 6\ell/s} \text{ as } S \to \infty.
\end{aligned}$$

Note that $\frac{5}{4} \leq 1 + \frac{2}{3+6\ell/s}$ if and only if $\ell \leq 5s/6$. Thus, $\frac{5}{4}$ is a lower bound on the competitive ratio when $s/2 < \ell \leq 5s/6$ and $1 + \frac{2}{3+6\ell/s}$ is a lower bound on the competitive ratio when $\ell > 5s/6$.

4 Properties of Optimal Packings for Restricted Grid Scheduling

We say that a bin used in a packing is *bad* if it contains an item of size S, it has empty space at least $L - S$, and it occurs while items of size L remain. Note that a bin containing an item of size L and an item of size S has empty space at most $M - L - S = L - S - 1$, so it is not bad.

Lemma 2. *For any set of items and any sequence of bins, there exists an optimal thrifty packing that contains no bad bin.*

Proof. Fix a set of items and a sequence of bins and suppose every optimal thrifty packing contains a bad bin. Consider an optimal thrifty packing p which has the longest prefix without bad bins. Let b' be the last bin to which p assigns an item of size L. Then the first bad bin b occurs before b'. Since p is thrifty, the empty space in b is less than L.

Note that p has no empty bins (of size at least S) between b and b'. Otherwise, let b'' be the first empty bin following b. Since p is valid, $size(b'') < L$ and, when bin b'' arrives, only items of size L remain. Then each nonempty bin after b'', including b, contains exactly one item, which is of size L. Consider the packing p' obtained from p by moving one item of size S in b to bin b'' and moving the item of size L in b' to bin b. Then b' is empty in the packing p'. Since p is valid, p and p' are the same prior to bin b, there are no empty bins between b and b'' in p', and no item which occurred before b'' in p has been moved after b'' in p', it follows that p' is valid. But the cost of p' is equal to the cost of p − the size of b' + size of b'', which is lower than the cost of p, since $size(b'') < L \leq size(b')$. By

Corollary 1, there is an optimal thrifty packing with the same cost as p', which contradicts the optimality of p.

Let p' be the packing obtained from p by switching an item of size S in b with the item of size L in b'. Note that p' is optimal, since p is. Since b contains an item of size L, it is not bad in p'. Furthermore, bin b is not wasteful, since its empty space, which was less than L in p, is less than $L-(S-L) = S$ in p'. None of the bins in p are wasteful, since p is thrifty, so none of the bins in p' prior to b are wasteful. By Lemma 1, there is an optimal packing p'' that is identical to p' up to and including bin b and which has no wasteful bins after b. This implies that p'' is thrifty. But p'' has a longer prefix without bad bins than p does. Thus, it contradicts the choice of p.

This motivates the following definition.

Definition 1. *A partial packing is* reasonable *if every bin b contains*

- *one item of size S, if $size(b) \in [S, L-1]$,*
- *one item of size L, if $size(b) = L$,*
- *two items of size S or one item of size L, if $size(b) \in [L+1, L+S-1]$,*
- *one item of size S and one item of size L, if $size(b) = L+S$, and*
- *three items of size S or one item of size S and one item of size L, if $size(b) \in [L+S+1, 2L-1]$.*

Note that a reasonable partial packing contains no wasteful, bad, or empty bins.

For any packing, consider the first bin after which there are no items of size L remaining or at most 2 items of size S remaining. This bin is called the *key bin* of the packing. The partial packing that assigns the same items into each bin up to and including its key bin and assigns no items to any subsequent bin is called the *front* of the packing. We say that a packing is *reasonable* if it is thrifty and its front is reasonable.

Corollary 2. *For any set of items and any sequence of bins, there exists an optimal packing that is reasonable.*

From now on, we will restrict attention to reasonable packings.

Given a set of items and a sequence of bins, two reasonable partial packings can differ as to whether they use one item of size L or two items of size S in certain bins. Therefore, the numbers of items of size S and items of size L they do not assign may differ. However, if both have at least one item of size S and at least one item of size L unassigned, the set of bins they have used is the same and there is a simple invariant relating the numbers of unassigned items of size S and items of size L they have.

Lemma 3. *Consider two reasonable partial packings of a set of items into a sequence of bins. Suppose that before bin b, each packing has at least one unassigned item of size S and at least one unassigned item of size L or at least three unassigned items of size S. Then immediately after bin b has been packed, the number of items of size S available plus twice the number of items of size L available is the same for both packings.*

Once a packing runs out of items of size S, it may be impossible for it to completely use some bins, so this relationship does not necessarily hold.

For any sequence of bins σ and any nonnegative integers s and ℓ, let $OPT(\sigma, s, \ell)$ denote the cost of an optimal packing of s items of size S and ℓ items of size $L = 2S - 1$ using σ. This must be at least the sum of the sizes of all the items.

Proposition 1. $OPT(\sigma, s, \ell) \geq sS + \ell L$.

Given any optimal packing for a set of items, a packing for a subset of these items can be obtained by removing the additional items from bins, starting from the end.

Proposition 2. *For any integers* $s' \leq s$, $\ell' \leq \ell$, $OPT(\sigma, s', \ell') \leq OPT(\sigma, s, \ell)$.

For any sequence of bins σ and any nonnegative integers s and ℓ, let $R(\sigma, s, \ell)$ denote the maximum cost of any reasonable packing of s items of size S and ℓ items of size $L = 2S - 1$ using σ.

When all items have the same size, all thrifty algorithms, including OPT, behave exactly the same.

Proposition 3. *For all integers* $s, \ell > 0$, $R(\sigma, s, 0) = OPT(\sigma, s, 0)$ *and* $R(\sigma, 0, \ell) = OPT(\sigma, 0, \ell)$.

Thus, if R and OPT both run out of items of size L at the same time or they both run out of items of size S at the same time, then they will use the same set of bins and, hence, have the same cost. The following four lemmas describe the relationship between the costs incurred by R and OPT when one of them has run out of one size of items.

Lemma 4. *For any sequence of bins* σ *and any integers* $s, \ell \geq 0$, $R(\sigma, s, \ell) \leq OPT(\sigma, s + 2\ell, 0) + \ell(2S - 3)$.

Lemma 5. *For any sequence of bins* σ *and any integers* $s, \ell \geq 0$, *if* $2k = s + 2\ell$, *then* $R(\sigma, s, \ell) \leq OPT(\sigma, 0, k) + (s + \ell - k - 1)L + M$.

Lemma 6. *For any sequence of bins* σ *and any integers* $s, \ell \geq 0$, $R(\sigma, s + 2\ell, 0) \leq OPT(\sigma, s, \ell) + (\ell - 1)L + M$.

Lemma 7. *For any sequence of bins* σ *and any integers* $s, \ell \geq 0$, *if* $2k = s + 2\ell$, *then* $R(\sigma, 0, k) \leq OPT(\sigma, \min\{s, \ell\}, \ell) + (k - \ell)M$.

5 Simple Non-Optimal Algorithms

It is helpful to understand why simple reasonable algorithms are not optimal for the Restricted Grid Scheduling problem. The following examples show why a number of natural candidates don't work well enough.

Example 1. Consider the reasonable algorithm that always uses items of size S when there is a choice. Let $s = 2\ell$ and let σ consist of ℓ bins of size $2S$, followed by s bins of size S, and then ℓ bins of size M. For this instance, the algorithm has a performance ratio of $\frac{\ell \cdot 2S + \ell \cdot M}{\ell \cdot 2S + s \cdot S} = \frac{3}{2} - \frac{3}{4S}$, which is greater than $5/4$ for large S.

Example 2. Consider the reasonable algorithm that always uses items of size L when there is a choice. Let $s = 2\ell$ and let σ consist of ℓ bins of size $2S$, followed by $s-1$ bins of size L and then 1 bin of size M. For this instance, the algorithm has a performance ratio of $\frac{\ell \cdot 2S + (s-1) \cdot L + M}{\ell \cdot 2S + \ell \cdot L} = \frac{3}{2} + \frac{1}{\ell} - \frac{1/\ell + 1/2}{2S-1}$, which is also greater than $5/4$ for large S.

For the two instances considered above, the reasonable algorithm which alternates between using two items of size S and one item of size L when it has a choice would do well, achieving a performance ratio of $5/4$. However, it doesn't do as well on other instances.

Example 3. Let $2s = 3\ell$ and let σ consist of ℓ bins of size $2S$, followed by s bins of size S and $\ell/2$ bins of size M. For this instance, the algorithm which alternates between using two items of size S and one item of size L when it has a choice has a performance ratio of $\frac{\ell \cdot 2S + (s-\ell) \cdot S + \ell/2 \cdot M}{\ell \cdot 2S + s \cdot S} = 1 + \frac{\frac{\ell}{2} \cdot (M - 2S)}{(2\ell + s) \cdot S} = 1 + \frac{2}{7} - \frac{3}{7S}$, which is larger than $5/4$ for S sufficiently large.

As the above examples partially illustrate, once either the online algorithm or OPT has run out of one type of item (items of size S or items of size L), the adversary can give bins which make the online algorithm waste a lot of space. The algorithm we present in the next section aims to postpone this situation long enough to get a good ratio. The following example indicates the need for a second phase in order to obtain the optimal competitive ratios, which are less than $5/4$ in some cases.

Example 4. Consider the reasonable algorithm which uses one item of size L twice as often as two items of size S, when it has a choice. Let $3s = 2\ell$ and let σ consist of ℓ bins of size $2S$, followed by s bins of size S and $\ell/3$ bins of size M. For this instance, the algorithm packs $\ell/3 = s/2$ bins of size $2S$ with two items of size S, so it has a performance ratio of $\frac{\ell \cdot 2S + (\ell/3) \cdot M}{\ell \cdot 2S + s \cdot S} = 1 + \frac{\frac{s}{2} \cdot M - sS}{(2\ell + s) \cdot S} = 1 + \frac{s \cdot S - 3/2}{4 \cdot S} = \frac{5}{4} - \frac{3}{8S}$, which exceeds the lower bound of $1 + \frac{2}{3 + 6\ell/s} = \frac{7}{6}$ for S sufficiently large.

6 A Matching Upper Bound

In this section, we present a reasonable algorithm, 2-Phase-Packer, for the Restricted Grid Scheduling problem. It is asymptotically optimal: the competitive ratio matches the lower bound in Section 3 for all three ranges of the ratio s/ℓ of the initial numbers of items of size S and items of size L. Not surprisingly, 2-Phase-Packer has two phases.

In the first phase, it attempts to balance the number of items of size S and the number of items of size L it uses, aiming for the ratio indicated by the lower bound. When it receives bins where it has a choice of using one item of size L or two items of size S in part or all of that bin (i.e., bins with sizes in the ranges $[2S, 3S-2]$ and $[3S, 4S-3]$), it uses one item of size L in a certain fraction of them (and increments the variable L-bins) and uses two items of size S in the remaining fraction (and increments S-bins). The fraction varies depending on the original ratio s/ℓ: at least 2, between 2 and 6/5, and less than 6/5. It is enforced by a macro called UseLs, which indicates that an item of size L should be used if and only if L-bins $\leq r$S-bins, where r is the target ratio of L-bins to S-bins. For example, when equal numbers of each should be used, we have UseLs = (L-bins $\leq$ S-bins). Both L-bins and S-bins start at zero, so, in this case, 2-Phase-Packer starts by choosing to use one item of size L and then alternates. Note that S-bins and L-bins do not change after Phase 1 is completed.

In the middle range for the ratio s/ℓ, there are two different fractions used. The first fraction is used until S-bins reaches a specific value. Afterwards, its choices alternate. To do so, for the rest of Phase 1, it records the number of times it chooses to pack two items of size S in a bin and the number of times it chooses to pack one item of size L in late-S-bins and late-L-bins, respectively. The variables late-S-bins and late-L-bins are also zero initially.

2-Phase-Packer uses countS and countL throughout the algorithm to keep track of the total numbers of items of size S and items of size L it has used, whether or not it had a choice. (Specifically, countS is incremented every time an item of size S is used and countL is incremented every time an item of size L is used.) It continues with Phase 1 until it has used a certain number of items of size S or items of size L (depending on the relationship between s and ℓ). For each of the three ranges of s/ℓ, we define a different condition for ending Phase 1. In Phase 2, only items of size S or only items of size L are used where a reasonable algorithm has a choice, depending on whether one would expect an excess of items of size S or items of size L, given the ratio s/ℓ.

The definitions of the end of Phase 1 for the various ranges of s/ℓ imply these inequalities.

Lemma 8. *If* $s/2 < \ell \leq 5s/6$, *then S-bins* $\leq 3s/8 - \ell/4 + 1$.
If $s < 6\ell/5$, *then S-bins* $\leq s/6 + 1$.

The following simple invariants can be proved inductively.

Lemma 9. *L-bins* $\leq$ *countL and* 2*S-bins* $\leq$ *countS.*
If $\ell \leq s/2$*, then S-bins* $\leq$ *L-bins* $\leq$ *S-bins* $+ 1$*.*
If $s/2 < \ell \leq 5s/6$*, then late-S-bins* $\leq$ *late-L-bins* $\leq$ *late-S-bins* $+ 1$ *and* $\lfloor c(S\text{-}bins - 1) \rfloor + 1 \leq L\text{-}bins \leq \lfloor c\ S\text{-}bins \rfloor + 1$*, where* $1 < c = \frac{10\ell - 3s}{s + 2\ell} \leq 2$*.*
If $5s/6 < \ell$*, then* $2S\text{-}bins \leq L\text{-}bins \leq 2S\text{-}bins + 1$*.*

$$\textbf{macro}\ \text{Phase1done} = \begin{cases} \text{countL} \geq \lfloor \ell/2 \rfloor & \text{if } \ell \leq s/2 \\ \text{countS} \geq \lfloor 3s/4 - \ell/2 \rfloor & \text{if } s/2 < \ell \leq 5s/6 \\ \text{countS} \geq \lfloor s/3 \rfloor & \text{if } 5s/6 < \ell \end{cases}$$

$$\textbf{macro}\ \text{UseLs} = \begin{cases} \text{L-bins} \leq \text{S-bins} & \text{if } \ell \leq s/2 \\ \textbf{if}\ (\text{S-bins} < \lfloor (s + 2\ell)/16 \rfloor) & \text{if } s/2 < \ell \leq 5s/6 \\ \textbf{then}\ \text{L-bins} \leq (10\ell - 3s)\text{S-bins}/(s + 2\ell) & \\ \textbf{else}\ \text{late-L-bins} \leq \text{late-S-bins} & \\ \text{L-bins} \leq 2\text{S-bins} & \text{if } 5s/6 < \ell \end{cases}$$

countS % counts the number of items of size S used; initially 0
countL % counts the number of items of size L used; initially 0
S-bins $\leftarrow$ L-bins $\leftarrow$ late-S-bins $\leftarrow$ late-L-bins $\leftarrow 0$

for each arriving bin b
  if only one type of item still remains **then** use as many items as fit in b
  else if $size(b) \in [S, 2S - 2]$ **then** use 1 item of size S
  else if $size(b) = 2S - 1$ **then** use 1 item of size L
  else if $size(b) = 3S - 1$ **then** use 1 item of size L and 1 item of size S
  else if only one item of size S still remains **then**
    use 1 item of size L
    if remaining space in b is at least S **then** use 1 item of size S
  else if only two items of size S still remain and $size(b) \in [3S, 4S - 3]$ **then**
    use 1 item of size L and 1 item of size S
  else if (not Phase1done) **then**
    % Use range determined ratio
    if $6\ell/5 \leq s < 2\ell$ and S-bins $\geq \lfloor (s + 2\ell)/16 \rfloor$ **then**
      if UseLs **then** late-L-bins ++
        else late-S-bins ++
    if $size(b) \in [2S, 3S - 2]$ **then**
      if UseLs **then** use 1 item of size L; L-bins ++;
      else use 2 items of size S; S-bins ++;
    if $size(b) \in [3S, 4S - 3]$ **then**
      if UseLs **then** use 1 item of size L and 1 item of size S; L-bins ++;
      else use 3 items of size S; S-bins ++;
  else % In Phase 2
    if $\ell \leq s/2$ **then** use as many items of size S as fit in bin b
    else use 1 item of size L
      if remaining space in b is at least S **then** use 1 item of size S
end for

Fig. 1. The algorithm 2-Phase-Packer

We analyse the performance of 2-Phase-Packer as compared to the cost of an optimal reasonable packing on an arbitrary sequence, σ, using the three different ranges for the relationship between s and ℓ in the lower bound.

Both 2-Phase-Packer and OPT use exactly the same bins until one of them runs out of either items of size L or items of size S. Thus, there are four cases to consider.

1. OPT runs out of items of size L at or before the point where 2-Phase-Packer runs out of anything.
2. OPT runs out of items of size S at or before the point where 2-Phase-Packer runs out of anything.
3. 2-Phase-Packer runs out of items of size L before OPT runs out of anything.
4. 2-Phase-Packer runs out of items of size S before OPT runs out of anything.

We only present the analysis for Case 1. The remaining cases are similar.

Consider an arbitrary sequence of bins σ in which 2-Phase-Packer packs s items of size S and ℓ items of size L. Suppose that, in this instance, OPT packs its last item of size L in bin b', but prior to bin b', both OPT and 2-Phase-Packer have unpacked items of both sizes. Let countL$'$ and countS$'$ denote the number of items of size L and items of size S, respectively, that 2-Phase-Packer has used up to and including bin b'. Let S-bins$'$ denote the number of bins at or before b' where 2-Phase-Packer had a choice and used two items of size S instead of one of size L and let L-bins$'$ denote the number where it used one item of size L and could have used two of size S instead. Let σ' denote the portion of σ after b'.

Both algorithms use all bins up to and including bin b', since all bins have size at least S. If X is the total cost of these bins, then $X \geq (\text{countS}')S + (\text{countL}')L$ and 2-Phase-Packer $(\sigma, s, \ell) = X +$ 2-Phase-Packer $(\sigma', s - \text{countS}', \ell - \text{countL}')$. In this case, Lemma 3 implies that, after bin b', OPT has $(s - \text{countS}') + 2(\ell - \text{countL}')$ items of size S remaining, so $OPT(\sigma, s, \ell) = X + OPT(\sigma', (s - \text{countS}') + 2(\ell - \text{countL}'), 0)$.

Since 2-Phase-Packer is reasonable, Lemma 4 implies that 2-Phase-Packer$(\sigma', s - \text{countS}', \ell - \text{countL}') \leq R(\sigma', s - \text{countS}', \ell - \text{countL}') \leq \text{OPT}(\sigma', (s - \text{countS}') + 2(\ell - \text{countL}'), 0) + (\ell - \text{countL}')(2S - 3)$, so 2-Phase-Packer$(\sigma, s, \ell) \leq \text{OPT}(\sigma, s, \ell) + (\ell - \text{countL}')(2S - 3)$. By Proposition 1, $\text{OPT}(\sigma, s, \ell) \geq sS + \ell L \geq sL/2 + \ell L = (s + 2\ell)L/2$. When computing the competitive ratio, we will choose the additive constant to be at least L, so we subtract L from 2-Phase-Packer's cost in the ratio. Thus,

$$\begin{aligned}
\frac{\text{2-Phase-Packer}(\sigma, s, \ell) - L}{\text{OPT}(\sigma, s, \ell)} &\leq \frac{\text{OPT}(\sigma, s, \ell) + (\ell - \text{countL}')(2S - 3) - L}{\text{OPT}(\sigma, s, \ell)} \\
&\leq 1 + \frac{(\ell - \text{countL}' - 1)(L - 2)}{(s + 2\ell)L/2} \\
&\leq 1 + \frac{2(\ell - \text{countL}' - 1)}{s + 2\ell}.
\end{aligned}$$

It remains to bound $\ell - \text{countL}' - 1$ in each of the three ranges for the ratio s/ℓ. We use the fact that, immediately after bin b', OPT has run out of items of size L, so at least ℓ bins of size at least L are in σ up to and including bin b'.

First, consider the case when $\ell \leq s/2$. If $\text{countL}' < \lfloor \ell/2 \rfloor$, then Phase 1 was not completed when bin b' arrived. Therefore, each time a bin of size at least L arrives up to and including bin b', 2-Phase-Packer either packs an item of size L in it and, hence, increments countL, or it increments S-bins. Hence, $\text{countL}' + \text{S-bins}' \geq \ell$. By Lemma 9, $\text{L-bins}' \geq \text{S-bins}'$. Since $\text{countL}' \geq \text{L-bins}'$, by Lemma 9, it follows that $\text{countL}' \geq \ell/2 \geq \lfloor \ell/2 \rfloor$. This is a contradiction. Thus, $\text{countL}' \geq \lfloor \ell/2 \rfloor$ and

$$\begin{aligned}\frac{\text{2-Phase-Packer}(\sigma, s, \ell) - L}{\text{OPT}(\sigma, s, \ell)} &\leq 1 + \frac{2(\ell - \text{countL}' - 1)}{s + 2\ell} \\ &\leq 1 + \frac{\ell}{s + 2\ell} = 1 + \frac{1}{2 + s/\ell}.\end{aligned}$$

So, suppose that $s/2 < \ell$. During Phase 1, each time a bin of size at least L arrives, 2-Phase-Packer either packs an item of size L in it and, hence, increments countL, or it increments S-bins. During Phase 2, 2-Phase-Packer packs an item of size L in each such bin until it runs out of items of size L. Therefore, $\text{countL}' + \text{S-bins}' \geq \ell$.

If $\ell \leq 5s/6$, then Lemma 8 implies that $\ell - \text{countL}' \leq \text{S-bins}' \leq 3s/8 - \ell/4 + 1$ and

$$\begin{aligned}\frac{\text{2-Phase-Packer}(\sigma, s, \ell) - L}{\text{OPT}(\sigma, s, \ell)} &\leq 1 + \frac{2(\ell - \text{countL}' - 1)}{s + 2\ell} \;\leq\; 1 + \frac{2(3s/8 - \ell/4)}{s + 2\ell} \\ &= 1 + \frac{s/4 + \ell/2 + s/2 - \ell}{s + 2\ell} \;<\; 1 + \frac{s/4 + \ell/2}{s + 2\ell} = \frac{5}{4}.\end{aligned}$$

Similarly, if $s < 6\ell/5$, then $\ell - \text{countL}' \leq \text{S-bins}' \leq s/6 + 1$, by Lemma 8, and

$$\begin{aligned}\frac{\text{2-Phase-Packer}(\sigma, s, \ell) - L}{\text{OPT}(\sigma, s, \ell)} &\leq 1 + \frac{2(\ell - \text{countL}' - 1)}{s + 2\ell} \\ &\leq 1 + \frac{s/3}{s + 2\ell} \;<\; 1 + \frac{2}{3 + 6\ell/s}.\end{aligned}$$

7 Conclusions and Open Problems

We have shown that varying the proportion of items of size S to items of size L does not lead to a larger competitive ratio if the maximum bin size is at most $4S - 3$. This may also be the case for an arbitrary maximum bin size, but there are complications. First, in bins of size $2kS$, where $k \geq 2S - 1$, it is possible to pack more than k items of size L. In addition, it may be an advantage to

have mixed bins which contain more than one item of size S and more than one item of size L. So, when bins can be large, one needs to consider how to maintain a good ratio of items of size L to items of size S, as they are used. We conjecture that maintaining the ratios specified in 2-Phase-Packer is sufficient.

One could also consider the the Grid Scheduling problem for two bin sizes, S and $L \neq 2S - 1$. For example, when L is a multiple of S, then the algorithm of Example 2, which always uses as many items of size L as possible, is optimal and, hence, has competitive ratio 1. It would be interesting to see if changing the ratio of S to L could improve the lower bound. We conjecture that it does not.

For the general problem, where there could be more than two sizes of items, we would like to close the gap between our lower bound and the upper bound of $\frac{13}{7}$ in [1]. Using many item sizes, it is not hard to prove a lower bound of $\frac{3}{2}$ on the strict competitive ratio (the competitive ratio where the additive constant in the definition is zero). However, the strict competitive ratio is not very interesting for this problem, since the ratio can be made larger simply by increasing the size of M and giving a last bin of size M.

Finally, it would be interesting to consider the competitive ratio of randomized algorithms for the Grid Scheduling problem or Restricted Grid Scheduling problem against an oblivious adversary.

References

1. J. Boyar and L. M. Favrholdt. Scheduling jobs on Grid processors. *Algorithmica*, 57(4):819–847, 2010.
2. J. Boyar and L. M. Favrholdt. A new variable-sized bin packing problem. *Journal of Scheduling*, 15:273–287, 2012.
3. Leah Epstein and Asaf Levin. More online bin packing with two item sizes. *Discrete Optimization*, 5(4):705–713, 2008.
4. Gregory Gutin, Tommy R. Jensen, and Anders Yeo. On-line bin packing with two item sizes. *Algorithmic Operations Research*, 1(2), 2006.
5. A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.
6. Frank M. Liang. A lower bound for on-line bin packing. *Discrete Optimization*, 5(4):705–713, 2008.
7. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. of the ACM*, 28(2):202–208, 1985.
8. G. Zhang. A new version of on-line variable-sized bin packing. *Discrete Applied Mathematics*, 72:193–197, 1997.