Two competitive ratio approximation schemes for a classical scheduling problem*

Christian Cöster
christian.coester@tu-dortmund.de

Matt McCutchen
matt@mattmccutchen.net

Revision of January 25, 2015

1 Introduction

For a long time, optimal competitive factors of online algorithms used to be approximated by manually finding and improving upper and lower bounds. The concept of a *competitive ratio approximation scheme (CRAS)*, introduced by Günther et al. in 2012 [13], defines a method of reducing this process to a task that can be executed automatically by a computer up to an arbitrarily good precision: A CRAS for an online problem is an algorithm that, for any $\varepsilon > 0$, computes the optimal competitive ratio for this problem up to a multiplicative error of $1 + \varepsilon$, and also computes an algorithm that achieves the computed competitive ratio. We describe a CRAS for the classical scheduling problem $P||C_{\max}$ for any fixed number m of identical machines: A sequence of jobs must be scheduled on the machines with the objective of minimizing the makespan. The jobs arrive over a list and the processing time of each job is known upon arrival, but the processing times of future jobs and the length of the list are unknown; the next job (or the end of the list) is only revealed once the current job has been assigned to a machine. This is called the *online-list* model. In contrast, in the *online-time* model, jobs arrive over time (possibly multiple jobs at once) and one can wait until a later time to assign a job to a machine to start processing.

1.1 Related Work

Graham's list scheduling algorithm [12] is a simple algorithm for $P||C_{\max}$ that achieves a competitive ratio of $2 - \frac{1}{m}$. This is optimal for $m = 2, 3$ [8]. Decades later, better algorithms were found for some specific values of m [5, 10, 18] and for general m [3, 15, 2, 9]. Also lower bounds on the optimal competitive ratio were improved [4, 2, 11, 14]. The best known upper and lower bounds for general m are 1.9201 [9] and 1.88 [14] respectively.

Günther et al. introduced CRASes in [13]. They proved their existence for scheduling in the online-time model in the cases of parallel, related and unrelated machines for several objective functions. CRASes for $P||C_{\max}$ in the online-list model are given in [16] and [6], theoretically allowing to close the gap between upper and lower bound on the optimal competitive ratio for any fixed m. The algorithm from [6] constitutes the core of this paper.

CRASes have also been developed for non-scheduling problems such as online bin packing [7] and the k-server problem in a finite metric space [17].

1.2 Outline

In Section 2 we describe a CRAS based on the work by Chen, Ye and Zhang in 2013 [6]. We change several details of their algorithm and analysis that allow us to present a simpler proof and make it easier to understand. The running time of the CRAS is polynomial in the number of machines. We arrive at a time bound of $\min\{m^{2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}))}}, 2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}))m}\}$. This is better than the bound of $m^{2^{O(\varepsilon^{-2}\log^2(\varepsilon^{-1}))}}$ stated in [6]. From a practical point of view, the dependency on ε^{-1} seems to set a strong limit, though. In Section 3 we present a different CRAS for the same problem, based on one by Megow and Wiese [16] from 2013, and explain its relations to the algorithm by Chen, Ye and Zhang. Megow and Wiese's CRAS is easier to understand, but does not achieve polynomial time.

*Final project paper in the Advanced Algorithms class at Massachusetts Institute of Technology, Fall 2014

2 A polynomial time CRAS

Let ρ_m^* be the infimum of competitive ratios achieved by online algorithms for $P||C_{\max}$ with m machines. In this section, following Chen, Ye, and Zhang [6] with simplifications, we construct for given $m \geq 2$ and sufficiently small $\varepsilon > 0$ an online algorithm for $P||C_{\max}$ with m machines that is $(1 + O(\varepsilon))\rho_m^*$-competitive. This proves the existence of a CRAS for $P||C_{\max}$. For fixed ε, the running time of the CRAS is polynomial in m. For the remainder of the section, we treat m and ε as fixed.

The basic approach is to view the online $P||C_{\max}$ problem as a game between an adversary, who chooses the size of each job to release, and the scheduler, which assigns the jobs to machines. This game is infinitely large, but we show that it is approximated by a simplified game of size polynomial in m in the sense that a strategy for either game can be applied to the other, losing a factor of $1 + O(\varepsilon)$ in the competitive ratio. Hence we can find the optimal strategy for the simplified game by brute force, and it gives us a strategy within a factor $1 + O(\varepsilon)$ of optimal for the original game.

To reach the simplified game, we approximate each *state* of the original problem (describing the jobs on each machine so far) by a *trimmed state* that omits the details of "small" jobs on each machine and remembers only their total load on the machine. Roughly speaking, we observe that trimmed states that differ by permutation of the machines or by multiplying all job sizes by a constant are equivalent, and we show that we need not consider trimmed states with highly imbalanced load because the optimal strategy does not use them; this leaves us with a number of states polynomial in m. Finally, we give general techniques to handle very small and large jobs so we can consider only a constant number of possible job sizes for the adversary to release.

2.1 States and the scheduling game

The first step in simplifying the input is a geometric rounding technique that is also used in polynomial time approximation schemes for offline scheduling problems (see e. g. [1]). By the *size* of a job we mean its processing time.

2.1 Lemma. *Let $\mathcal{A}$ be a $(1 + O(\varepsilon))\rho_m^*$-competitive algorithm for the restriction of $P||C_{\max}$ to instances where all job sizes are of the form $(1 + \varepsilon)^i$ for $i \in \mathbb{Z}$. Let $\mathcal{A}'$ be the algorithm for general input that first rounds a job size up to the next power of $(1 + \varepsilon)$ and then assigns it according to $\mathcal{A}$. Then $\mathcal{A}'$ is $(1 + O(\varepsilon))\rho_m^*$-competitive.*

Proof. Given a job sequence σ, let σ' be the sequence with job sizes rounded up to the next power of $(1 + \varepsilon)$. We denote the maximum machine load of the optimal offline schedules for σ and σ' by $\mathsf{OPT}(\sigma)$ and $\mathsf{OPT}(\sigma')$ respectively. Since jobs in σ cannot be larger than the corresponding jobs in σ', the maximum machine load of the schedule computed by $\mathcal{A}'$ for σ is no more than the maximum machine load of the schedule computed by $\mathcal{A}$ for σ', which is at most $(1+O(\varepsilon))\rho_m^*\mathsf{OPT}(\sigma')$. However, $\mathsf{OPT}(\sigma') \leq (1+\varepsilon)\mathsf{OPT}(\sigma)$, since the optimal schedule for σ can be interpreted as a feasible schedule for σ' where each job is a factor of at most $1 + \varepsilon$ larger. The lemma follows since $(1 + O(\varepsilon))(1 + \varepsilon) = 1 + O(\varepsilon)$. □

Thanks to this lemma, it suffices to consider the restricted problem where all job sizes are of the form $(1 + \varepsilon)^i$ for $i \in \mathbb{Z}$. With this restriction, the *state* of an algorithm after scheduling a sequence of jobs is given by the infinite matrix

$$\psi = (\psi_i^h)_{i \in \mathbb{Z},\ h=1,\ldots,m}$$

where ψ_i^h denotes the number of jobs of size $(1 + \varepsilon)^i$ on machine h. We use the following definitions:

- $J(\psi)$ is the set of jobs in the sequence that led to state ψ. We identify a job with its size so that $J(\psi)$ can also be thought of as a multiset of job sizes.
- $C_{\max}(\psi)$ is the maximum load of a machine in state ψ.
- $\mathsf{OPT}(\psi)$ is the value $C_{\max}$ of an (offline) optimal schedule of $J(\psi)$.
- $\rho(\psi) = \frac{C_{\max}(\psi)}{\mathsf{OPT}(\psi)}$ is the approximation ratio that the algorithm achieves if the input ends while it is in state ψ, assuming that $J(\psi) \neq \emptyset$.

Since the input can end at any time, the competitive ratio of a (deterministic) algorithm is the supremum of $\rho(\psi)$ over all states ψ that are reached by the algorithm on some input sequence.

In general, we can think of an online problem as a game between an algorithm and an adversary: the algorithm tries to minimize the approximation ratio of the final state while the adversary tries to maximize it. In this setting, the adversary chooses an integer i giving the size $(1+\varepsilon)^i$ of the next job to release, and the algorithm (the scheduler) chooses a machine $h \in \{1, \dots, m\}$ to place the job on. This cycle repeats until the adversary declares the end of the input. Let $\mathit{Assign}(\psi, i, h)$ denote the state after a job of size $(1+\varepsilon)^i$ is added to machine h in ψ, i.e., ψ_i^h is increased by 1. If we look only n jobs ahead, then we can write a recurrence for the worst approximation ratio $\rho^n(\psi)$ that the adversary can force starting from a given state ψ by releasing at most n jobs:

$$\rho^0(\psi) = \rho(\psi), \quad \rho^n(\psi) = \max\Big(\rho(\psi),\ \sup_{i\in\mathbb{Z}} \min_{1\le h\le m} \rho^{n-1}(\mathit{Assign}(\psi, i, h))\Big) \text{ for } n > 0$$

Given an integer i for the next job, the scheduler chooses a machine h so as to minimize the approximation ratio the adversary will be able to force using the remaining $n-1$ jobs; the adversary chooses i (or chooses to end the game immediately) to maximize the ratio he will be able to achieve against best play by the scheduler.

We would like to evaluate this recurrence to find the optimal strategy for each player, but we can only do so if there are finitely many moves and values of n to consider, and we will ensure that we only need to consider finitely many values of n by having finitely many states. So the meat of the construction will be to establish a relationship between the original game and a game with these finiteness properties, using the ideas mentioned at the beginning of Section 2.

2.2 Simulating states with trimmed states

Let $c \in \mathbb{N}$ and $\omega \in \mathbb{N}$ be the smallest integers such that $(1+\varepsilon)^{-c} \le \varepsilon$ and $(1+\varepsilon)^\omega \ge 2$. We call an infinite matrix

$$\phi = (\phi_i^h)_{i=-c,\dots,\infty}^{h=1,\dots,m}$$

with non-negative integer entries ϕ_i^h a *trimmed state*. Since a trimmed state can be viewed as a state (by setting $\phi_i^h = 0$ for $i < -c$), the definitions $J(\phi)$, $C_{\max}(\phi)$, $\mathit{OPT}(\phi)$ and $\rho(\phi)$ also make sense for trimmed states. To distinguish between states and trimmed states, we may also call the former *real states*.

We are particularly interested in trimmed states that are simulating as per the following definition: Given a real state ψ and a trimmed state ϕ we write $\phi \overset{\ell}{\propto} \psi$ or just $\phi \propto \psi$ and say that ϕ *simulates* ψ *with shift* $\ell \in \mathbb{Z}$ if

$$\phi_i^h = \psi_{\ell\omega+i}^h \quad \text{for } i > -c \text{ and}$$
$$0 \le \phi_{-c}^h - \sum_{d=0}^{\infty} \psi_{\ell\omega-c-d}^h (1+\varepsilon)^{-d} < 2.$$

for all $h = 1, \dots, m$. We refer to jobs of size $(1+\varepsilon)^{-c}$ in $J(\phi)$ and jobs of size at most $(1+\varepsilon)^{\ell\omega-c}$ in $J(\psi)$ as *small* and other jobs as *big*. So the definition says that subject to the shift, the big jobs match exactly between ψ and ϕ and the total load of small jobs on each machine is the same or is slightly greater in ϕ.

To prove relationships between OPT values of different job sets, the following lemma will come in handy.

2.2 Lemma. *Let B and C be sets of jobs, viewed as multisets of the job sizes, and let $B + C$ denote the combined set. Then*

$$\mathit{OPT}(B + C) \le \max\Big(\mathit{OPT}(B),\ \frac{1}{m}\sum(B + C) + \max(C)\Big),$$

where OPT denotes the maximum machine load of the optimum schedule for the jobs, $\sum(B+C)$ denotes the sum of the job sizes in B and C, and $\max(C)$ denotes the size of the largest job in C.

Proof. Construct a feasible schedule S for $B+C$ by taking an optimal schedule for B and adding each job of C greedily to the least loaded machine. If the greedy addition does not increase $C_{\max}(S)$, then we still have $C_{\max}(S) = \mathit{OPT}(B)$ and the claim holds. Otherwise, let p be the size of the last job of C placed on the most heavily loaded machine in S. Before this job was added, that machine was the least loaded with load $C_{\max}(S) - p$, so the total load at that time was at least $m(C_{\max}(S) - p)$. But the total load can never be more than $\sum(B + C)$. Since $p \le \max(C)$, it follows that $C_{\max}(S) \le \frac{1}{m}\sum(B + C) + \max(C)$ and the claim holds. □

As we will see next, we can almost recompute $\mathsf{OPT}(\psi)$ from the information contained in a simulating state ϕ, making an error on the order of the size of a small job.

2.3 Lemma. *If $\phi \overset{\ell}{\propto} \psi$, then*

$$\mathsf{OPT}(\psi) - (1+\varepsilon)^{\ell\omega-c} \le (1+\varepsilon)^{\ell\omega}\mathsf{OPT}(\phi) \le \mathsf{OPT}(\psi) + 3(1+\varepsilon)^{\ell\omega-c},$$

Proof. Let $\bar{\phi}$ denote the state obtained by multiplying each job size in ϕ by $(1+\varepsilon)^{\ell\omega}$ to match the scale of ψ. Clearly $\mathsf{OPT}(\bar{\phi}) = (1+\varepsilon)^{\ell\omega}\mathsf{OPT}(\phi)$. It follows from the definition of $\phi \overset{\ell}{\propto} \psi$ that $J(\bar{\phi})$ contains the same big jobs as $J(\psi)$ and the average machine load of $J(\bar{\phi})$ exceeds the average machine load of $J(\psi)$ by between 0 and $2(1+\varepsilon)^{\ell\omega-c}$.

For the first inequality, apply Lemma 2.2 with B being the common set of big jobs and C being the set of small jobs in $J(\psi)$, so $B + C = J(\psi)$. Clearly $\mathsf{OPT}(B) \le \mathsf{OPT}(\bar{\phi})$. Since $\mathsf{OPT}(\bar{\phi})$ is lower bounded by the average load of $J(\bar{\phi})$, which is at least that of $J(\psi)$, we have $\frac{1}{m}\sum(B+C) \le \mathsf{OPT}(\bar{\phi})$. Finally we observe that $\max(C) \le (1+\varepsilon)^{\ell\omega-c}$, so the statement of Lemma 2.2 becomes

$$\mathsf{OPT}(\psi) \le \max(\mathsf{OPT}(\bar{\phi}), \mathsf{OPT}(\bar{\phi}) + (1+\varepsilon)^{\ell\omega-c})$$

and the inequality follows.

For the second inequality, apply Lemma 2.2 with B being the common set of big jobs and C being the set of small jobs in $J(\bar{\phi})$. The argument is the same except that now the average load of $J(\bar{\phi})$ can exceed that of $J(\psi)$ by $2(1+\varepsilon)^{\ell\omega-c}$, so we obtain $\frac{1}{m}\sum(B+C) \le \mathsf{OPT}(\psi) + 2(1+\varepsilon)^{\ell\omega-c}$. □

The idea is that if $\phi \overset{\ell}{\propto} \psi$, then $\mathsf{OPT}(\psi)$ is approximately $(1+\varepsilon)^{\ell\omega}\mathsf{OPT}(\phi)$. We want ℓ to be small enough for the error to be small; specifically, we want ℓ to be small enough for $\mathsf{OPT}(\phi)$ to be greater than 1. Indeed, if this is the case then the approximation ratio of a real state and a trimmed state that simulates it are very close:

2.4 Lemma. *If $\phi \overset{\ell}{\propto} \psi$ and $\mathsf{OPT}(\phi) \ge 1$, then $\rho(\psi) \le \rho(\phi)(1+O(\varepsilon))$ and $\rho(\phi) \le \rho(\psi)(1+O(\varepsilon))$.*

Proof. It is immediate from the definition of $\phi \overset{\ell}{\propto} \psi$ that

$$0 \le (1+\varepsilon)^{\ell\omega}C_{\max}(\phi) - C_{\max}(\psi) < 2(1+\varepsilon)^{\ell\omega-c}.$$

The lemma follows by combining this property with Lemma 2.3 and the facts that $C_{\max}(\phi) \ge \mathsf{OPT}(\phi) \ge 1$ and $(1+\varepsilon)^{-c} \le \varepsilon$. We omit the details of the calculation. □

We can add jobs to a simulating pair of a real and a trimmed state without violating any of our desired properties (we will see later how to handle real jobs smaller than $(1+\varepsilon)^{\ell\omega-c}$):

2.5 Lemma. *For every $i \ge -c$ and $h \in \{1,\dots,m\}$:*

- *if $\phi \overset{\ell}{\propto} \psi$, then $\mathsf{Assign}(\phi,i,h) \overset{\ell}{\propto} \mathsf{Assign}(\psi,\ell\omega+i,h)$;*
- *if $\mathsf{OPT}(\phi) \ge 1$, then $\mathsf{OPT}(\mathsf{Assign}(\phi,i,h)) \ge 1$.*

Proof. Immediate. □

We now have a set of states to work with where the job sizes are bounded downwards. To further reduce the set of states, for a given real state ψ, we would like to use a simulating ϕ that is *well-scaled*, meaning that $1 \le \mathsf{OPT}(\phi) < (1+\varepsilon)^{\omega+4}$. There are only finitely many well-scaled trimmed states, since job sizes have to be between $(1+\varepsilon)^{-c}$ and $(1+\varepsilon)^{\omega+3}$ and there can only be finitely many jobs of any size without violating the upper bound on $\mathsf{OPT}(\phi)$. We will count the states more carefully later after we have reduced the set even further.

In order to keep the current simulating state well-scaled as jobs get added, we may need to increase the shift ℓ. We define a function $\mathsf{Rescale}$ to do this. Let ϕ be a trimmed state with $\mathsf{OPT}(\phi) \ge 1$. If ϕ is already well-scaled, let $\mathsf{Rescale}(\phi) = \phi$. Otherwise, let $k \ge 1$ be the integer such that $(1+\varepsilon)^{k\omega+1} \le \mathsf{OPT}(\phi) < (1+\varepsilon)^{(k+1)\omega+1}$. Then $\mathsf{Rescale}(\phi) = \phi'$ where

$$\phi'^h_i = \phi^h_{k\omega+i} \quad \text{for } i > -c \text{ and}$$
$$\phi'^h_{-c} = \left\lceil \sum_{d=0}^{k\omega} \phi^h_{k\omega-c-d}(1+\varepsilon)^{-d} \right\rceil.$$

The effect is to multiply each job's size by $(1+\varepsilon)^{-k\omega}$ and then collect all jobs of size $(1+\varepsilon)^{-c}$ or smaller and round their total load up to an integer number of small jobs.

2.6 Lemma. *If ϕ is a trimmed state with $\mathsf{OPT}(\phi) \geq 1$, then $\mathsf{Rescale}(\phi)$ is well-scaled.*

Proof. If ϕ is already well-scaled, the statement is trivial. Otherwise let $\phi' = \mathsf{Rescale}(\phi)$ and let k be as in the definition of *Rescale*. If we view ϕ as a real state, then by comparing the definitions of $\propto$ and *Rescale* we see that $\phi' \overset{k}{\propto} \phi$. Now, Lemma 2.3 implies that

$$(1+\varepsilon)^{-k\omega}\mathsf{OPT}(\phi) - (1+\varepsilon)^{-c} \leq \mathsf{OPT}(\phi') \leq (1+\varepsilon)^{-k\omega}\mathsf{OPT}(\phi) + 3(1+\varepsilon)^{-c}.$$

By choice of k and c, the left hand side is at least 1 and the right hand side is less than

$$(1+\varepsilon)^{\omega+1} + 3\varepsilon < (1+\varepsilon)^{\omega+4}.$$

□

We show next that *Rescale* maintains the simulation relation.

2.7 Lemma. *If $\phi \propto \psi$ then $\mathsf{Rescale}(\phi) \propto \psi$.*

Proof. If ϕ is already well-scaled, the statement is trivial. Otherwise let $\phi' = \mathsf{Rescale}(\phi)$, let k be as in the definition of *Rescale* and suppose $\phi \overset{\ell}{\propto} \psi$. We claim that $\phi' \overset{\ell+k}{\propto} \psi$. For $i > -c$, we have ${\phi'}^h_i = \phi^h_{k\omega+i} = \psi^h_{(\ell+k)\omega+i}$, since $\phi \overset{\ell}{\propto} \psi$. Substituting the definition of ${\phi'}^h_{-c}$, we need to show that

$$0 \leq \left\lceil \sum_{d=0}^{k\omega} \phi^h_{k\omega-c-d}(1+\varepsilon)^{-d} \right\rceil - \sum_{d=0}^{\infty} \psi^h_{(\ell+k)\omega-c-d}(1+\varepsilon)^{-d} < 2.$$

Since taking the ceiling increases the value by less than 1, it suffices to show that

$$0 \leq \sum_{d=0}^{k\omega} \phi^h_{k\omega-c-d}(1+\varepsilon)^{-d} - \sum_{d=0}^{\infty} \psi^h_{(\ell+k)\omega-c-d}(1+\varepsilon)^{-d} \leq 1. \tag{1}$$

Since $\phi \overset{\ell}{\propto} \psi$, we have $\phi^h_{k\omega-c-d} = \psi_{(\ell+k)\omega-c-d}$ for $d < k\omega$. Thus the middle term can be simplified to

$$\phi^h_{-c}(1+\varepsilon)^{-k\omega} - \sum_{d=k\omega}^{\infty} \psi^h_{(\ell+k)\omega-c-d}(1+\varepsilon)^{-d} = \left(\phi^h_{-c} - \sum_{d=0}^{\infty} \psi^h_{\ell\omega-c-d}(1+\varepsilon)^{-d}\right)(1+\varepsilon)^{-k\omega}.$$

By the definition of $\phi \overset{\ell}{\propto} \psi$, the first factor is between 0 and 2. By choice of ω we have $1/2 \geq (1+\varepsilon)^{-\omega} \geq (1+\varepsilon)^{-k\omega} \geq 0$, so both inequalities in (1) are satisfied. □

We now have the ability to maintain a well-scaled trimmed state that simulates a real state as we add jobs of size at least $(1+\varepsilon)^{\ell\omega-c}$ to the real state. In the next section, we will use this technique to analyze the scheduling game, in the process describing how to handle smaller jobs.

2.3 Relating real and trimmed scheduler strategies

We are ready to define a "trimmed" version of the scheduling game. This game is played using well-scaled trimmed states. The adversary begins by choosing $i \in \{0, \ldots, \omega-1\}$; the first job is released with size $(1+\varepsilon)^i$ and the scheduler assigns it to a machine, leading to a trimmed state ϕ. Since $\mathsf{OPT}(\phi)$ is simply the size of this job, ϕ is well-scaled. Now, the following cycle repeats until the adversary ends the game: First the adversary chooses an arbitrary integer $i \geq -c$ giving the size $(1+\varepsilon)^i$ of the next job to release. Then the scheduler chooses a machine h to place it on and the game passes to the state $\mathsf{Rescale}(\mathsf{Assign}(\phi, i, h))$, which is well-scaled by Lemmas 2.5 and 2.6.

A *real* or *trimmed scheduler strategy* (referring to the real or trimmed game) is a function S that takes a state of the game and a value of i legal for the game, indicating release of a job of size $(1+\varepsilon)^i$, and returns the machine h on which the job is placed. Let $\mathcal{R}(S)$ denote the set of states that are *reachable* by S, i.e., there exists a sequence of jobs that leads the strategy into this state. We can show that the possible scheduler strategies for the two games are, indeed, related:

2.8 Lemma.

(a) Let T be a trimmed scheduler strategy. Then there exists a real scheduler strategy S such that for every $\psi \in \mathcal{R}(S)$, there exists $\phi \in \mathcal{R}(T)$ such that $\phi \propto \psi$.

(b) Let S be a real scheduler strategy. Then there exists a trimmed scheduler strategy T such that for every $\phi \in \mathcal{R}(T)$, there exists $\psi \in \mathcal{R}(S)$ such that $\phi \propto \psi$.

Proof. (a): We will define S for every real state ψ for which there exists at least one $\phi \in \mathcal{R}(T)$ such that $\phi \propto \psi$; we will simultaneously show inductively that no other states are reached, which proves the claim. For each real state ψ with this property, arbitrarily choose a $\phi \in \mathcal{R}(T)$ such that $\phi \propto \psi$, and let ℓ be the shift of this simulation. Suppose a job of size $(1+\varepsilon)^{\ell\omega+i}$ is released. If $i \geq -c$, we assign it to the same machine to which T, starting from ϕ, would assign a job of size $(1+\varepsilon)^i$. Let ψ' and ϕ' be the resulting states; clearly $\phi' \in \mathcal{R}(T)$, and by Lemmas 2.5 and 2.7, $\phi' \propto \psi'$. Note that ϕ' may not be the same simulating state of ψ' that we use to define S on ψ'. This is okay; we only care that at least one such state exists.

If $i < -c$, we must be more careful. If there exists a machine h such that $\phi \overset{\ell}{\propto} \mathsf{Assign}(\psi, \ell\omega + i, h)$, then we assign the job to that machine; after rescaling, we are still in a state ψ' such that $\phi \propto \psi'$. If every machine h has the property that adding the job would violate the simulation of ϕ, that means that

$$0 \leq \phi^h_{-c} - \sum_{d=0}^{\infty} \psi^h_{\ell\omega-c-d}(1+\varepsilon)^{-d} < 2 \quad \text{but} \quad \phi^h_{-c} - \left(\sum_{d=0}^{\infty} \psi^h_{\ell\omega-c-d}(1+\varepsilon)^{-d} + (1+\varepsilon)^{i+c} \right) < 0.$$

Since $(1+\varepsilon)^{i+c} \leq 1$, we have for every h:

$$0 \leq (\phi^h_{-c} + 1) - \left(\sum_{d=0}^{\infty} \psi^h_{\ell\omega-c-d}(1+\varepsilon)^{-d} + (1+\varepsilon)^{i+c} \right) < 1 < 2 \tag{2}$$

So we release a job of size $(1+\varepsilon)^{-c}$ against T and assign the real job to the same machine h to which T assigns the trimmed job. (2) is precisely the condition for the resulting states (before rescaling) to be in simulation with respect to small jobs on machine h, and nothing else changes. So after rescaling, we again have $\phi' \in \mathcal{R}(T)$ and $\phi' \propto \psi'$.

Finally, we must define S for the first job. Let $(1+\varepsilon)^i$ be its size. At the beginning of the trimmed game, we release a job of size $(1+\varepsilon)^{(i \bmod \omega)}$ against T and assign the real job to the same machine to which T assigns the trimmed job. We then have $\phi \overset{\lfloor i/\omega \rfloor}{\propto} \psi$.

Every state $\psi \in \mathcal{R}(S)$ is reached by applying some sequence of the above cases, so the claim is proved.

(b): Analogously to (a), we will define T for every trimmed state ϕ for which there exists at least one $\psi \in \mathcal{R}(S)$ such that $\phi \propto \psi$. For each trimmed state ϕ with this property, arbitrarily choose a $\psi \in \mathcal{R}(S)$ such that $\phi \propto \psi$, and let ℓ be the shift of this simulation. When a job of size $(1+\varepsilon)^i$ is released, we release a job of size $(1+\varepsilon)^{\ell\omega+i}$ against S and assign the trimmed job to the same machine; there are no extra cases to consider. If the first job has size $(1+\varepsilon)^i$, we release a job of the same size to S and assign the trimmed job to the same machine. The rest of the proof is the same as in (a). □

The competitive ratio of a real scheduler strategy S is $\rho(S) = \sup_{\psi \in \mathcal{R}(S)} \rho(\psi)$, and analogously for trimmed scheduler strategies. Our work on the trimmed game now pays off:

2.9 Lemma. *Let T^* be an optimal trimmed scheduler strategy. Both T^* and the corresponding real scheduler strategy S given by Lemma 2.8(a) are $(1+O(\varepsilon))\rho^*_m$-competitive in their respective games.*

Proof. Since ρ^*_m is defined as an infimum, we do not know whether there is a ρ^*_m-competitive real scheduler strategy, but we can let S^* be a $(1+\varepsilon)\rho^*_m$-competitive real scheduler strategy. Let T be the corresponding trimmed scheduler strategy given by Lemma 2.8(b). For every $\phi \in \mathcal{R}(T)$, there is a $\psi \in \mathcal{R}(S^*)$ such that $\phi \propto \psi$; by definition of $\rho(S^*)$ we have $\rho(\psi) \leq \rho(S^*) \leq (1+\varepsilon)\rho^*_m$. By Lemma 2.4, $\rho(\phi) \leq (1+O(\varepsilon))\rho(\psi) \leq (1+O(\varepsilon))\rho^*_m$. Since T^* is the optimal trimmed scheduler strategy, $\rho(T^*) \leq \rho(T) = \sup_{\phi \in \mathcal{R}(T)} \rho(\phi) \leq (1+O(\varepsilon))\rho^*_m$. This proves the statement for T^*.

By a similar argument using the other inequality of Lemma 2.4, $\rho(S) \leq (1+O(\varepsilon))\rho(T^*)$. Chaining the inequalities gives $\rho(S) \leq (1+O(\varepsilon))\rho^*_m$ as desired. □

The hardest work is done. To solve $P||C_{\max}$, we will compute the optimal trimmed scheduler strategy and then execute the corresponding real scheduler strategy.

2.4 Simplifications to the trimmed game

As mentioned before, there are only finitely many well-scaled trimmed states, so apart from the issue that the adversary can release infinitely many possible job sizes (which we address last), we could already compute the optimal trimmed strategy in finite time by a brute force approach. In this section, we will reduce the game size to polynomial in m in order to compute the optimal trimmed scheduler strategy in time polynomial in m.

We say a trimmed state ϕ is *feasible* if $C_{\max}(\phi) \le 2\mathit{OPT}(\phi)$.

2.10 Lemma. *Any optimal trimmed scheduler strategy reaches only feasible states.*

Proof. Since $\rho_m^* < 1.9201$ (see [9]) and given our assumption that ε is sufficiently small, the competitive ratio of an optimal trimmed scheduler strategy is at most 2 by Lemma 2.9. Hence it only reaches states with $C_{\max}(\phi) \le 2\mathit{OPT}(\phi)$. □

Thus, it is enough to consider the *feasible trimmed game*: the restriction of the trimmed game to feasible states, with scheduler moves that would enter an infeasible state removed. One can show that given any feasible state and job size, the scheduler always has a move to a feasible state by assigning the job to the least loaded machine ([6] Lemma 4.3). Instead of proving this, we say that the game ends with $\rho = \infty$ if the scheduler is left without a feasible move.

To reduce the number of states even further, we introduce an equivalence relation on trimmed states: We say that two trimmed states are *equivalent* if they are the same up to permutation of machines. Intuitively, it seems there can be no benefit to treating equivalent states differently. Indeed, it is easy to verify that any strategy can be changed to respect the equivalence relation without worsening its competitive ratio. Thus, it is sufficient to find the optimal strategy in the *standard trimmed game* played on equivalence classes of feasible, well-scaled states. A unique representative of each equivalence class can be found by reordering the machines according to a lexicographic ordering of machine load vectors. We call these representative states *standard* and denote by $\mathit{Sort}(\phi)$ the representative of the equivalence class of ϕ. If ϕ is a standard state, the standard trimmed game passes to $\mathit{Sort}(\mathit{Rescale}(\mathit{Assign}(\phi, i, h)))$ when a job of size $(1+\varepsilon)^i$ is assigned to machine h.

2.11 Lemma. *The number of standard states is* $\min\left\{m^{2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}))}}, 2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}))m}\right\}$*, i. e., polynomial in m for fixed ε and exponential in $\varepsilon^{-1}\log(\varepsilon^{-1})$ for fixed m.*

Proof. A feasible state has $C_{\max}(\phi) \le 2\mathit{OPT}(\phi)$, and a well-scaled state has $\mathit{OPT}(\phi) < (1+\varepsilon)^{\omega+4}$, so the load on a machine cannot exceed $2(1+\varepsilon)^{\omega+4} = O(1)$. The smallest job size is $(1+\varepsilon)^{-c} = \Omega(\varepsilon)$, so there can be no more than $O(\varepsilon^{-1})$ jobs on a machine. Job sizes in well-scaled states are between $(1+\varepsilon)^{-c}$ and $(1+\varepsilon)^{\omega+3}$, so the number of possible job sizes is

$$\omega + c + 4 = \lceil \log_{1+\varepsilon} 2 \rceil + \lceil \log_{1+\varepsilon} \varepsilon^{-1} \rceil + 4 = O(\varepsilon^{-1}) + O(\varepsilon^{-1}\log\varepsilon^{-1}) = O(\varepsilon^{-1}\log\varepsilon^{-1}).$$

If we specify the set of jobs on a machine by listing their sizes, the number of such lists is at most

$$O(\varepsilon^{-1}\log\varepsilon^{-1})^{O(\varepsilon^{-1})} = 2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}\log\varepsilon^{-1}))} = 2^{O(\varepsilon^{-1}(\log\varepsilon^{-1}+\log\log\varepsilon^{-1}))} = 2^{O(\varepsilon^{-1}\log\varepsilon^{-1})}.$$

A standard state is uniquely determined by the number of machines with each job combination, which is between 0 and m, so $(m+1)^{2^{O(\varepsilon^{-1}\log\varepsilon^{-1})}}$ is an upper bound on the number of standard states. We can absorb the "+ 1" into the exponent. The other bound holds because for each machine there are $2^{O(\varepsilon^{-1}\log\varepsilon^{-1})}$ possible job combinations.□

The adversary still has an infinite number of possible job sizes to release starting from a given standard state. Each job size presents the scheduler with a set of standard states that the scheduler can reach by assigning the job, and there can only be finitely many such sets, but to compute the optimal strategy, we would still need to show that we can determine which ones they are. In fact, it is enough to consider finitely many job sizes, because any new job larger than a certain size will cause all previous jobs on each machine to rescale into a single small job, and the new job has finitely many possible sizes after rescaling. Formally:

2.12 Lemma. *Let μ be the smallest multiple of ω greater than or equal to $c+3\omega+4$. Let ϕ be a feasible, well-scaled trimmed state. Then for every $i \ge \mu$,*

$$\mathit{Rescale}(\mathit{Assign}(\phi, i, h)) = \mathit{Rescale}(\mathit{Assign}(\phi, \mu + (i \bmod \omega), h)).$$

Proof. Let $\phi' = \mathsf{Assign}(\phi, i, h)$. We claim that $\mathsf{OPT}(\phi') = (1+\varepsilon)^i$. Clearly $\mathsf{OPT}(\phi')$ is lower bounded by the new job of size $(1+\varepsilon)^i$. Furthermore, we can construct a schedule ϕ'^* for $J(\phi')$ with $C_{\max}(\phi'^*) = (1+\varepsilon)^i$ as follows: Starting with ϕ, move all jobs from the first machine to the second machine, and place the new job on the first machine. Now the first machine has load $(1+\varepsilon)^i$, and since ϕ is feasible and well-scaled, each of the other machines has load at most $4\mathsf{OPT}(\phi) < 4(1+\varepsilon)^{\omega+4} \le (1+\varepsilon)^{3\omega+4} < (1+\varepsilon)^i$.

Thus, the value of k chosen in the definition $\mathsf{Rescale}(\phi')$ is $k = \lfloor (i-1)/\omega \rfloor$. Note that $i - \omega \le k\omega \le i - 1$. Let $\phi'' = \mathsf{Rescale}(\phi')$. We claim that ϕ'' has the following form: ${\phi''}^h_{1+((i-1) \bmod \omega)} = 1$, ${\phi''}^{h'}_{-c} = 1$ if machine h' is nonempty in ϕ or 0 if it is empty, and all other entries zero. The lemma follows because this form is the same for any two values of $i \ge \mu$ that differ by a multiple of ω. We prove this as follows:

- For $j > -c$ and any machine h', ${\phi''}^{h'}_j$ is defined as ${\phi'}^{h'}_{k\omega+j}$. We have

  $$k\omega + j \ge \omega\lfloor (i-1)/\omega \rfloor - c + 1 \ge i - \omega - c + 1 \ge 2\omega + 5 > \omega + 3,$$

  but ϕ is well-scaled, so $\phi^h_{k\omega+j} = 0$. Thus ${\phi'}^{h'}_{k\omega+j} = 1$ if $k\omega + j = i$ and $h' = h$ (reflecting the newly assigned job) and 0 otherwise. Since $k\omega + j = i$ holds for $j = 1 + ((i-1) \bmod \omega)$, the claim for coordinates greater than $-c$ follows.

- For the coordinate for small jobs of a machine h' we have

  $${\phi''}^{h'}_{-c} = \left\lceil \sum_{d=0}^{k\omega} {\phi'}^{h'}_{k\omega-c-d}(1+\varepsilon)^{-d} \right\rceil = \left\lceil (1+\varepsilon)^{c-k\omega} \sum_{j=-c}^{-c+k\omega} {\phi'}^{h'}_j (1+\varepsilon)^j \right\rceil = \left\lceil (1+\varepsilon)^{c-k\omega} \sum_{j=-c}^{-c+k\omega} \phi^{h'}_j (1+\varepsilon)^j \right\rceil,$$

  since ${\phi'}^{h'}_j = \phi^{h'}_j$ except when $j = i$ and $h' = h$, but $i > -c + k\omega$, so this case does not occur in the sum. Furthermore, since ϕ is well-scaled and $-c + k\omega \ge \omega + 3$, we are summing over all values of j for which $\phi^{h'}_j$ can possibly be nonzero, hence the sum represents the load on machine h' in ϕ. If the machine is empty, we get ${\phi''}^{h'}_{-c} = 0$ as claimed. Otherwise, since ϕ is feasible, its load is at most $2\mathsf{OPT}(\phi) < (1+\varepsilon)^{2\omega+4}$, so the quantity inside the ceiling is less than $(1+\varepsilon)^{c-k\omega+2\omega+4}$. The exponent is upper bounded by

  $$c - (i - \omega) + 2\omega + 4 \le c + 3\omega + 4 - \mu \le 0.$$

  So ${\phi''}^{h'}_{-c} = 1$ as claimed. □

So it is sufficient to consider the restriction of the standard trimmed game to job sizes less than $(1+\varepsilon)^{\mu+\omega}$, which we call the *limited trimmed game*. (Of course, the corresponding real job size increases over time as the state is rescaled.) By Lemma 2.12, a strategy for this game can be applied to the standard trimmed game with no additional reachable states, so no increase in competitive ratio. The size of the limited trimmed game is finally polynomial in m.

2.5 Computing the optimal trimmed scheduler strategy

Following the idea in Section 2.1, for a standard state ϕ, let $\rho^n(\phi)$ be the worst approximation ratio that the adversary can force in the limited trimmed game starting from ϕ by releasing at most n jobs. It is given by the recurrence:

$$\rho^0(\phi) = \rho(\phi), \quad \rho^n(\phi) = \max\left(\rho(\phi), \max_{-c \le i < \mu+\omega} \min_{\substack{1 \le h \le m \\ \text{feasible}}} \rho^{n-1}(\mathsf{Sort}(\mathsf{Rescale}(\mathsf{Assign}(\psi, i, h)))) \right) \text{ for } n > 0$$

As per the definition of the game, we take the min to be ∞ if the scheduler has no feasible move.

First we make a list of the standard states by running through the counting procedures in Lemma 2.11 and compute $\rho(\phi)$ for each of them. To do this, we need to know $\mathsf{OPT}(\phi)$. But the offline optimal schedule for $J(\phi)$ is itself represented by a standard state, so we can group the standard states by job set and use the lowest value of $C_{\max}$ in each group as OPT for all members of the group.

If we view ρ^n as a vector across standard states, we can express the structure of the recurrence as a function f independent of n such that $\rho^n = f(\rho^{n-1})$. This means that if $\rho^{N+1} = \rho^N$ for some N, then $\rho^{N+2} = f(\rho^{N+1}) =$

$f(\rho^N) = \rho^{N+1}$, and by induction $\rho^n = \rho^N$ for $n \geq N$. We also know that $\rho^{n+1}(\phi) \geq \rho^n(\phi)$, because anything the adversary can do with a limit of n jobs, he can also do with a limit of $n+1$. That is to say, the vectors ρ^n are coordinate-wise nondecreasing as n increases. Let Σ be the number of standard states. It is evident from the recurrence that $\rho^n(\phi)$ is always of the form ∞ or $\rho(\phi')$ for some standard state ϕ', so there are at most $\Sigma+1$ distinct possible values, hence any given coordinate $\rho^n(\phi)$ increases at most Σ times. ρ^n has Σ coordinates, so ρ^n can increase at most Σ^2 times before it becomes constant: i.e., $\rho^n = \rho^N$ for $n \geq N = \Sigma^2$. Thus we can compute ρ^N in time $O(B^3)$, where B is the bound for Σ in Lemma 2.11. (The iteration over values of i and h is dominated by B.)

Since $\rho^n = \rho^N$ for $n \geq N$, $\rho^N(\phi)$ gives the worst approximation ratio that the adversary can force from ϕ with an unlimited number of jobs. Let $\rho^*_{m,\text{trim}}$ be the maximum of $\rho^N(\phi)$ over all ϕ that result from first jobs of sizes $(1+\varepsilon)^i$ for $0 \leq i < \omega$ (for a standard state, the first job must be placed on the first machine). Then $\rho^*_{m,\text{trim}}$ is the optimal competitive ratio for the limited trimmed game, because the adversary can force it in $N+1$ jobs and the following scheduler strategy T^* achieves it: starting from a state ϕ with $\rho^N(\phi) \leq \rho^*_{m,\text{trim}}$, when a job of size $(1+\varepsilon)^i$ is released, always move to a state ϕ' with $\rho^N(\phi') \leq \rho^*_{m,\text{trim}}$. Such a state must exist; if not, the recurrence for $\rho^{N+1}(\phi)$ would evaluate to greater than $\rho^*_{m,\text{trim}}$, but we know $\rho^{N+1} = \rho^N$.

So our online algorithm for $P||C_{\max}$ with m machines is to compute T^* as above and then apply a corresponding real scheduler strategy S from Lemma 2.8(a) to the unrounded jobs. We do so by keeping track of a feasible, well-scaled trimmed state $\phi \propto \psi$ such that $\mathit{Sort}(\phi) \in \mathcal{R}(T^*)$, and acting based on how T^* acts on $\mathit{Sort}(\phi)$. By Lemmas 2.9 and 2.1, this strategy is $(1+O(\varepsilon))\rho^*_m$-competitive, as desired. We have shown:

2.13 Theorem. *For any m, there exists a CRAS for $P||C_{\max}$ on m machines that runs in time* $\min\{m^{2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}))}}, 2^{O(\varepsilon^{-1}\log(\varepsilon^{-1}))m}\}$.

3 A slightly different approach to a CRAS

In this section, we present a CRAS for $P||C_{\max}$ based on the one by Megow and Wiese [16], which is arguably simpler than the CRAS of Section 2 but does not run in time polynomial in m for fixed ε, as the CRAS of Section 2 does. The analysis in [16] is rather loose, but fine if the goal is only to compute a near-optimal strategy in finite time. We present a tighter analysis that is closer in many details to the analysis of Section 2 to highlight the essential difference that remains: we reduce to a finite number of states not by tracking the total load of small jobs on each machine, but by ignoring small jobs completely. For this cruder approach to work, we must use a lower threshold for "small" than in Section 2. Consequently, the number of combinations of big jobs that we must track as separate states is larger, and in particular not polynomial in m.

By Lemma 2.1, it suffices to solve the problem where all job sizes are of the form $(1+\varepsilon)^i$. We continue to use the definitions related to real states from Section 2. As in Section 2, let ω be the smallest integer such that $(1+\varepsilon)^\omega \geq 2$. We say a job set is *sparse* if for every integer k, the total number of jobs of size between $(1+\varepsilon)^{k\omega}$ and $(1+\varepsilon)^{(k+1)\omega-1}$ inclusive is at most $\lceil m/\varepsilon \rceil$. Then:

3.1 Lemma. *Suppose $\mathcal{A}'$ is an algorithm that achieves a competitive ratio of ρ on sparse job sets. Then we can construct an algorithm $\mathcal{A}$ that achieves a competitive ratio of $(1+O(\varepsilon))\rho$ on arbitrary job sets.*

Proof. Algorithm $\mathcal{A}$ works by running an instance of $\mathcal{A}'$ and releasing a sparse sequence of jobs to it. Based on the decisions of $\mathcal{A}'$, $\mathcal{A}$ decides where to put its own jobs.

When $\mathcal{A}$ receives a job, let k denote the largest integer such that $\lceil m/\varepsilon \rceil$ jobs of size between $(1+\varepsilon)^{k\omega}$ and $(1+\varepsilon)^{(k+1)\omega-1}$ inclusive have been released to $\mathcal{A}'$, or $-\infty$ if no such integer exists. Clearly k is nondecreasing. Let $\mathit{Load}(\mathcal{A}^h)$ and $\mathit{Load}(\mathcal{A}'^h)$ denote the load on machine h from the point of view of $\mathcal{A}$ and $\mathcal{A}'$, respectively. We maintain the following invariant for every machine h:

$$0 \leq \mathit{Load}(\mathcal{A}^h) - \mathit{Load}(\mathcal{A}'^h) \leq (1+\varepsilon)^{(k+1)\omega}$$

We can think of this as true initially with $k = -\infty$.

When $\mathcal{A}$ receives a job of size $(1+\varepsilon)^{(k+1)\omega}$ or larger (which we call *big* for the purpose of this lemma), we release it to $\mathcal{A}'$ (which is legal by definition of k) and assign it to the same machine that $\mathcal{A}'$ does. When $\mathcal{A}$ receives a job of size $p < (1+\varepsilon)^{(k+1)\omega}$ (which we call *small*), if we can assign it to some machine without violating the invariant, we do so. If not, then every machine h satisfies:

$$\mathit{Load}(\mathcal{A}^h) - \mathit{Load}(\mathcal{A}'^h) > (1+\varepsilon)^{(k+1)\omega} - p$$

We release a *container job* of size $(1+\varepsilon)^{(k+1)\omega}$ to $\mathcal{A}'$ and assign the original job to the same machine to which $\mathcal{A}'$ assigns the container job. One may easily verify that the invariant holds after we do this. (Note the similarity to the proof of Lemma 2.8(a).) After $\mathcal{A}$ finishes processing the received job, we increase k if appropriate, which only helps the invariant.

At the end of the input, let ψ and ψ' denote the states of $\mathcal{A}$ and $\mathcal{A}'$, respectively. We claim (similar to what is used in Lemma 2.4):

$$C_{\max}(\psi') \le \rho \mathit{OPT}(\psi') \tag{3}$$

$$C_{\max}(\psi) \le C_{\max}(\psi') + (1+\varepsilon)^{(k+1)\omega} \tag{4}$$

$$\mathit{OPT}(\psi') \le \mathit{OPT}(\psi) + (1+\varepsilon)^{(k+1)\omega} \tag{5}$$

$$\mathit{OPT}(\psi') \ge \varepsilon^{-1}(1+\varepsilon)^{k\omega} \tag{6}$$

Inequality (3) states that $\mathcal{A}'$ is ρ-competitive. Inequality (4) follows immediately from the invariant. To prove (5), we apply Lemma 2.2 with B being the set of big jobs (as determined when they were released) and C being the set of container jobs, so $B + C = J(\psi')$. Clearly $\mathit{OPT}(B) \le \mathit{OPT}(\psi)$. Since $\mathit{OPT}(\psi)$ is lower bounded by the average load of ψ, which is at least that of ψ' by the invariant, we have $\frac{1}{m}\sum(B+C) \le \mathit{OPT}(\psi)$. Finally $\max(C) \le (1+\varepsilon)^{(k+1)\omega}$, so the statement of Lemma 2.2 becomes

$$\mathit{OPT}(\psi') \le \max(\mathit{OPT}(\psi), \mathit{OPT}(\psi) + (1+\varepsilon)^{(k+1)\omega})$$

and (5) follows. (6) is true because ψ' contains $\lceil m/\varepsilon \rceil$ jobs of size between $(1+\varepsilon)^{k\omega}$ and $(1+\varepsilon)^{(k+1)\omega-1}$ inclusive, hence the average load of ψ' is at least $\varepsilon^{-1}(1+\varepsilon)^{k\omega}$, which is a lower bound on $\mathit{OPT}(\psi')$. A calculation using (3)–(6) and the fact that $(1+\varepsilon)^{\omega} = O(1)$ now shows that $C_{\max}(\psi) \le (1+O(\varepsilon))\rho\mathit{OPT}(\psi)$, as desired. □

So it suffices to solve the problem for sparse job sets, and from here on we consider only states ψ for which $J(\psi)$ is sparse. Let s be the smallest integer such that $(1+\varepsilon)^{-s\omega} \le \varepsilon^2/m$. If ψ is a nonempty state, let $\mathit{ScaleTrim}(\psi)$ denote the state obtained from ψ using the following two steps: First divide all job sizes in ψ by $(1+\varepsilon)^{\ell\omega}$ for an integer $\ell = \ell(\psi)$ so that the largest resulting job size is between 1 and $(1+\varepsilon)^{\omega-1}$ inclusive. Then remove all jobs that are smaller than $(1+\varepsilon)^{-s\omega}$ (these jobs are called *irrelevant*). We call states of the form $\mathit{ScaleTrim}(\psi)$ *well-scaled trimmed states*. These are simply the states ϕ such that $\mathit{ScaleTrim}(\phi) = \phi$. As in Section 2, there are only finitely many well-scaled trimmed states, since there are finitely many possible job sizes and job sets are sparse.

The function $\mathit{ScaleTrim}$ plays the role of the relation $\propto$ from Section 2; here we can define it as a function since the well-scaled trimmed state that simulates a real state is unique. We now prove that the approximation ratios of simulating states are close (compare to Lemma 2.4):

3.2 Lemma. *If ψ is a state such that $J(\psi)$ is nonempty (and sparse), then*

$$\rho(\psi) \le (1+O(\varepsilon))\rho(\mathit{ScaleTrim}(\psi)) \text{ and } \rho(\mathit{ScaleTrim}(\psi)) \le (1+O(\varepsilon))\rho(\psi).$$

Proof. For convenience, let $\phi = \mathit{ScaleTrim}(\psi)$. Let $\bar{\phi}$ denote the state obtained by multiplying each job size in ϕ by $(1+\varepsilon)^{\ell(\psi)\omega}$ to match the scale of ψ. Clearly $\rho(\bar{\phi}) = \rho(\phi)$. Then $\bar{\phi}$ contains exactly the relevant jobs of ψ: those of size at least $(1+\varepsilon)^{(\ell(\psi)-s)\omega}$. For each integer k, since $J(\psi)$ is sparse, ψ contains at most $\lceil m/\varepsilon \rceil$ jobs of size between $(1+\varepsilon)^{k\omega}$ and $(1+\varepsilon)^{(k+1)\omega-1}$ inclusive. The total size of these jobs is upper bounded by $\lceil m/\varepsilon \rceil (1+\varepsilon)^{(k+1)\omega}$. Summing this bound for all $k < \ell(\psi) - s$, the total size of the irrelevant jobs of ψ is upper bounded by

$$\sum_{k=-\infty}^{\ell(\psi)-s-1} \lceil m/\varepsilon \rceil (1+\varepsilon)^{(k+1)\omega} = \lceil m/\varepsilon \rceil \frac{(1+\varepsilon)^{(\ell(\psi)-s)\omega}}{1-(1+\varepsilon)^{-\omega}} \le \lceil m/\varepsilon \rceil \frac{(1+\varepsilon)^{\ell(\psi)\omega}(\varepsilon^2/m)}{1-1/2} = O(\varepsilon)(1+\varepsilon)^{\ell(\psi)\omega}.$$

Hence, no matter what machines the irrelevant jobs are on in ψ, we have

$$C_{\max}(\bar{\phi}) \le C_{\max}(\psi) \le C_{\max}(\bar{\phi}) + O(\varepsilon)(1+\varepsilon)^{\ell(\psi)\omega}.$$

Similarly, we have

$$\mathit{OPT}(\bar{\phi}) \le \mathit{OPT}(\psi) \le \mathit{OPT}(\bar{\phi}) + O(\varepsilon)(1+\varepsilon)^{\ell(\psi)\omega}$$

because we can add the irrelevant jobs anywhere in an optimal schedule for $\bar{\phi}$. Finally, by choice of ℓ, we know that the largest job size in ψ is at least $(1+\varepsilon)^{\ell(\psi)\omega}$, which provides a lower bound on $\mathit{OPT}(\bar{\phi})$ because $\bar{\phi}$ contains this job. The lemma now follows from these inequalities. □

We define the trimmed scheduling game in the same way as it was defined in Section 2.3, except that *ScaleTrim* replaces *Rescale* and job sizes after the initial job may be *any* power of $1+\varepsilon$. Let $\mathcal{R}(S)$ continue to denote the set of states reachable by a strategy S. We now show the analogue of Lemma 2.8:

3.3 Lemma.

(a) Let T be a trimmed scheduler strategy. Then there exists a real scheduler strategy S such that for every $\psi \in \mathcal{R}(S)$, $\mathsf{ScaleTrim}(\psi) \in \mathcal{R}(T)$.

(b) Let S be a real scheduler strategy. Then there exists a trimmed scheduler strategy T such that for every $\phi \in \mathcal{R}(T)$, there exists $\psi \in \mathcal{R}(S)$ such that $\mathsf{ScaleTrim}(\psi) = \phi$.

Proof. For (a), suppose we are in a state ψ such that $\mathsf{ScaleTrim}(\psi) \in \mathcal{R}(T)$ and a job of size $(1+\varepsilon)^i$ is released. We release a job of size $(1+\varepsilon)^{i-\ell(\psi)\omega}$ against T and assign the real job to the same machine. Likewise, for (b), suppose we are in a state ϕ such that there exists $\psi \in \mathcal{R}(S)$ such that $\mathsf{ScaleTrim}(\psi) = \phi$; choose an arbitrary such ψ. When a job of size $(1+\varepsilon)^i$ is released, we release a job of size $(1+\varepsilon)^{i+\ell(\psi)\omega}$ against S and assign the trimmed job to the same machine. It is easy to verify that we maintain the desired properties. □

3.4 Lemma. *Let T^* be an optimal trimmed scheduler strategy. Then the corresponding real scheduler strategy given by Lemma 3.3(a) is $(1+O(\varepsilon))\rho_m^*$-competitive.*

Proof. Analogous to Lemma 2.9. □

3.5 Lemma. *The number of well-scaled trimmed states is $2^{O(m\varepsilon^{-2}(\log(\varepsilon^{-1})+\log m)^2)}$.*

Proof. In a well-scaled trimmed state, all job sizes are between $(1+\varepsilon)^{-s\omega}$ and $(1+\varepsilon)^{\omega-1}$ inclusive; this is $(s+1)\omega$ possible sizes. We can describe the state by giving the number of jobs of each size on each machine. Since the job set is sparse, each of these numbers is between 0 and $\lceil m/\varepsilon \rceil$. Thus, the number of well-scaled trimmed states is upper bounded by $(\lceil m/\varepsilon \rceil + 1)^{m(s+1)\omega}$. We have $\lceil m/\varepsilon \rceil + 1 = 2^{O(\log(\varepsilon^{-1})+\log m)}$, $\omega = O(\varepsilon^{-1})$, and $s+1 = O(\varepsilon^{-1}(\log(\varepsilon^{-1}) + \log m))$, so the lemma follows. □

We observe that in the trimmed game, releasing an irrelevant job (smaller than $(1+\varepsilon)^{-s\omega}$) has no effect, and releasing a job of size $(1+\varepsilon)^i$ where $i \geq (s+2)\omega$ has the same effect as a job of size $(1+\varepsilon)^{(s+1)\omega+(i \bmod \omega)}$: all existing jobs become irrelevant and the new job is scaled to size $(1+\varepsilon)^{i \bmod \omega}$. (Compare to Lemma 2.12.) Thus, we have a finite number of adversary moves to consider. We can now compute the optimal trimmed strategy by brute force as in Section 2.

Note where the non-polynomial dependence of the number of well-scaled trimmed states on m comes from. In Lemma 3.1, we need m/ε jobs of size between $(1+\varepsilon)^{k\omega}$ and $(1+\varepsilon)^{(k+1)\omega-1}$ before we can argue that $\mathsf{OPT}(\psi')$ is lower bounded by the average load of $\varepsilon^{-1}(1+\varepsilon)^{k\omega}$, and thereby justify introducing an error of $(1+\varepsilon)^{(k+1)\omega} = O(\varepsilon\mathsf{OPT}(\psi'))$ to be able to handle any more such jobs without passing them to $\mathcal{A}'$. The average load calculation takes into account the possibility of spreading the jobs across all m machines. However, when converting a trimmed strategy to a real strategy, since the trimmed strategy does not track jobs at all once they become irrelevant, we have no way to ensure we do not put all m/ε irrelevant jobs on the same machine, so to bound the error by $O(\varepsilon\mathsf{OPT}(\phi))$ we have to set the size threshold for irrelevance at $\Theta((\varepsilon^2/m)\mathsf{OPT}(\phi))$. Hence, the number of job sizes in a well-scaled trimmed state depends on m, so the number of job combinations for a machine depends on m. Contrast to Section 2, in which jobs smaller than $\Theta(\varepsilon\mathsf{OPT}(\phi))$ are considered small and the number of job combinations for a machine depends only on ε, so we could bound the number of states by a polynomial in m by counting the number of machines with each job combination (Lemma 2.11).

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 32–43, 1999.

[2] S. Albers. Better bounds for online scheduling. In *SIAM Journal on Computing*, pages 130–139, 1997.

[3] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51(3):359–366, 1995.

[4] Y. Bartal, H. Karloff, and Y. Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50:113–116, 1994.

[5] B. Chen, A. van Vliet, and G. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16(4):221–230, 1994.

[6] L. Chen, D. Ye, and G. Zhang. Approximating the optimal competitive ratio for an ancient online scheduling problem. *CoRR*, abs/1302.3946, 2013.

[7] L. Chen, D. Ye, and G. Zhang. An asymptotic competitive scheme for online bin packing. In Z. Zhang, L. Wu, W. Xu, and D. Du, editors, *Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, pages 13–24. Springer International Publishing, 2014.

[8] U. Faigle, W. Kern, and G. Turan. On the performance of on-line algorithms for partition problems. *Acta Cybern.*, 9(2):107–119, 1989.

[9] R. Fleischer and M. Wahl. Online scheduling revisited. In *Algorithms-ESA 2000*, pages 202–210. Springer Berlin Heidelberg, 2000.

[10] G. Galambos and G. Woeginger. An on-line scheduling heuristic with better worst-case ratio than grahams list scheduling. *SIAM Journal on Computing*, 22(2):349–355, 1993.

[11] T. Gormley, N. Reingold, E. Torng, and J. Westbrook. Generating adversaries for request-answer games. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 564–565. Society for Industrial and Applied Mathematics, 2000.

[12] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

[13] E. Günther, O. Maurer, N. Megow, and A. Wiese. A new approach to online scheduling: Approximating the optimal competitive ratio. In *Proceedings of the 24st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 118–128, 2013.

[14] J. F. Rudin III. Improved bound for the online scheduling problem. PhD thesis. The University of Texas at Dallas, 2001.

[15] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20(2):400–430, 1996.

[16] N. Megow and A. Wiese. Competitive-ratio approximation schemes for minimizing the makespan in the online-list model. *CoRR*, abs/1303.1912, 2013.

[17] T. Mömke. A competitive ratio approximation scheme for the k-server problem in fixed finite metrics. *CoRR*, abs/1303.2963, 2013.

[18] J. F. Rudin III and R. Chandrasekaran. Improved bounds for the online scheduling problem. *SIAM J. Comput.*, 32(3):717–735, 2003.