The Variable-Processor Cup Game

William Kuszmaul*[1], Alek Westover†[1]

[1]MIT
kuszmaul@mit.edu, alek.westover@gmail.com

Abstract

The problem of scheduling tasks on p processors so that no task ever gets too far behind is often described as a game with cups and water. In the p-processor cup game on n cups, there are two players, a filler and an emptier, that take turns adding and removing water from a set of n cups. In each turn, the filler adds p units of water to the cups, placing at most 1 unit of water in each cup, and then the emptier selects p cups to remove up to 1 unit of water from. The emptier's goal is to minimize the backlog, which is the height of the fullest cup.

The p-processor cup game has been studied in many different settings, dating back to the late 1960's. All of the past work shares one common assumption: that p is fixed. This paper initiates the study of what happens when the number of available processors p varies over time, resulting in what we call the *variable-processor cup game*.

Remarkably, the optimal bounds for the variable-processor cup game differ dramatically from its classical counterpart. Whereas the p-processor cup has optimal backlog $\Theta(\log n)$, the variable-processor game has optimal backlog $\Theta(n)$. Moreover, there is an efficient filling strategy that yields backlog $\Omega(n^{1-\varepsilon})$ in quazi-polynomial time against any deterministic emptying strategy.

We additionally show that straightforward uses of randomization cannot be used to help the emptier. In particular, for any positive constant Δ, and any Δ-greedy-like randomized emptying algorithm $\mathcal{A}$, there is a filling strategy that achieves backlog $\Omega(n^{1-\varepsilon})$ against $\mathcal{A}$ in quazi-polynomial time.

1 Introduction

A fundamental challenge in processor scheduling is how to perform load balancing, that is, how to share processors among tasks in order to keep any one task from getting too far behind. Consider n tasks executing in time slices on $p < n$ processors. During each time slice, a scheduler must select p (distinct) tasks that will be executed during the slice; up to one unit of work is completed on each executed task. During the same time slice, however, up to p units of *new work* may be allocated to the tasks, with different tasks receiving different amounts of work. The goal of a load-balancing scheduler is to bound the backlog of the system, which is defined to be the maximum amount of uncompleted work for any task.

As a convention, the load balancing problem is often described as a game involving water and cups. The ***p-processor cup game*** is a multi-round game with two players, an ***emptier*** and a ***filler***, that takes place on n initially empty cups. At the beginning of each round, the filler adds up to p units of water to the cups, subject to the constraint that each cup receives at most 1 unit of water. The emptier then selects up to p distinct cups and removes up to 1 unit of water from each of them. The emptier's goal is to minimize the amount of water in the fullest cup, also known as the ***backlog***. In terms of processor scheduling, the cups represent tasks, the water represents work assigned to each task, and the emptier represents a scheduling algorithm.

Starting with the seminal paper of Liu [30], work on the p processor cup game has spanned more than five decades [7, 20, 8, 29, 27, 33, 6, 23, 30, 31, 16, 10, 25, 1, 15, 28]. In addition to processor scheduling [7, 20,

*Supported by a Hertz fellowship and a NSF GRFP fellowship
†Partially supported by MIT PRIMES

8, 29, 27, 33, 6, 23, 30, 31, 1, 28, 16], applications include network-switch buffer management [21, 4, 35, 19], quality of service guarantees [7, 1, 28], and data structure deamortization [2, 16, 15, 3, 34, 22, 17, 24, 9].

The game has also been studied in many different forms. Researchers have studied the game with a fixed-filling-rate constraint [7, 20, 8, 29, 27, 33, 6, 23, 30, 31], with various forms of resource augmentation [10, 25, 28, 16], with both oblivious and adaptive adversaries [1, 7, 30, 10, 25], with smoothed analysis [25, 10], with a semi-clairvoyant emptier [28], with competitive analysis [5, 18, 14], etc.

For the plain form of the p-processor cup game, the greedy emptying algorithm (i.e., always empty from the fullest cups) is known to be asymptotically optimal [1, 10, 25], achieving backlog $O(\log n)$. The optimal backlog for randomized emptying algorithms remains an open question [16, 10, 25] and is known to be between $\Omega(\log \log n)$ and $O(\log \log n + \log p)$ [25].

This paper: varying resources. Although cup games have been studied in many forms, all of the prior work on cup games shares one common assumption: the number p of processors is fixed.

In modern computing, however, computers are often shared among multiple applications, users, and even virtual OS's. The result is that the amount of resources (e.g., memory, processors, cache, etc.) available to a given application may fluctuate over time. The problem of handling cache fluctuations has received extensive attention in recent years (see work on cache-adaptive analysis [32, 11, 12, 13, 26]), but the problem of handling a varying number of processors remains largely unstudied.

This paper introduces the ***variable-processor cup game***, in which the filler is allowed to *change* p (the total amount of water that the filler adds, and the emptier removes, from the cups per round) at the beginning of each round. Note that we do not allow the resources of the filler and emptier to vary separately. That is, as in the standard game, we take the value of p for the filler and emptier to be identical in each round. This restriction is crucial since, if the filler is allowed more resources than the emptier, then the filler could trivially achieve unbounded backlog.

A priori having variable resources offers neither player a clear advantage. When the number p of processors is fixed, the greedy emptying algorithm (i.e., always empty from the fullest cups), is known to achieve backlog $O(\log n)$ [1, 10, 25] regardless of the value of p. This seems to suggest that, when p varies, the correct backlog should remain $O(\log n)$. In fact, when we began this project, we hoped for a straightforward reduction between the two versions of the game.

Results. We show that the variable-processor cup game is *not* equivalent to the standard p-processor game. By strategically controlling the number p of processors, the filler can achieve substantially larger backlog than would otherwise be possible.

We begin by constructing filling strategies against deterministic emptying algorithms. We show that for any positive constant ε, there is a filling strategy that achieves backlog $\Omega(n^{1-\varepsilon})$ within $2^{\text{polylog}(n)}$ rounds. Moreover, if we allow for $n!$ rounds, then there is a filling strategy that achieves backlog $\Omega(n)$. In contrast, for the p-processor cup game with any fixed p, the backlog never exceeds $O(\log n)$.

Our lower-bound construction is asymptotically optimal. By introducing a novel set of invariants, we deduce that the greedy emptying algorithm never lets backlog exceed $O(n)$.

A natural question is whether *randomized* emptying algorithms can do better. In particular, when the emptier is randomized, the filler is taken to be ***oblivious***, meaning that the filler cannot see what the emptier does at each step. Thus the emptier can potentially use randomization to obfuscate their behavior from the filler, preventing the filler from being able to predict the heights of cups.

When studying randomized emptying strategies, we restrict ourselves to the class of ***greedy-like emptying strategies***. This means that the emptier never chooses to empty from a cup c over another cup c' whose fill is more than $\omega(1)$ greater than the fill of c. All of the known randomized emptying strategies for the classic p-processor cup game are greedy-like [10, 25].

Remarkably, the power of randomization does not help the emptier very much in the variable-processor cup game. For any constant $\varepsilon > 0$, we give an oblivious filling strategy that achieves backlog $\Omega(n^{1-\varepsilon})$ in quazi-polynomial time (with probability $1 - 2^{-\text{polylog}\, n}$), no matter what (possibly randomized) greedy-like strategy the emptier follows.

Our results combine to tell a surprising story. They suggest that the problem of varying resources poses

a serious theoretical challenge for the design and analysis of load-balancing scheduling algorithms. There are many possible avenues for future work. Can techniques from beyond worst-case analysis (e.g., smoothing, resource augmentation, etc.) be used to achieve better bounds on backlog? What about placing restrictions on the filler (e.g., bounds on how fast p can change), allowing the emptier to be able to be semi-clairvoyant, or making stochastic assumptions on the filler? We believe that all of these questions warrant further attention.

2 Preliminaries

The cup game consists of a sequence of rounds. Let S_t denote the state of of the cups at the start of round t. At the beginning of the round, the filler chooses the number of processors p_t for the round. Next, the filler distributes p_t units of water among the cups (with at most 1 unit of water to any particular cup). Now the game is at the ***intermediate state*** for round t, which we call state I_t. Finally the emptier chooses p_t cups to empty at most 1 unit of water from, which marks the conclusion of round t. The state is then S_{t+1}.

If the emptier empties from a cup c on round t such that the fill of c is less than 1 in state I_t, then c now has 0 fill (not negative fill); we say that the emptier ***zeroes out*** c on round t. Note that on any round where the emptier zeroes out a cup the emptier has removed less total fill from the cups than the filler has added to the cups; hence the average fill of the cups has increased.

We denote the fill of a cup c at state S by $\text{fill}_S(c)$. Let the ***mass*** of a set of cups X at state S be $m_S(X) = \sum_{c \in X} \text{fill}_S(c)$. Denote the average fill of a set of cups X at state S by $\mu_S(X)$. Note that $\mu_S(X)|X| = m_S(X)$. Let the ***backlog*** at state S be $\max_c \text{fill}_S(c)$, let the ***anti-backlog*** at state S be $\min_c \text{fill}_S(c)$. Let the ***rank*** of a cup at a given state be its position in the list of the cups sorted by fill at the given state, breaking ties arbitrarily but consistently. For example, the fullest cup at a state has rank 1, and the least full cup has rank n. Let $[n] = \{1, 2, \ldots, n\}$, let $i + [n] = \{i + 1, i + 2, \ldots, i + n\}$. For a state S, let $S(r)$ denote the rank r cup at state S, and let $S(\{r_1, r_2, \ldots, r_m\})$ denote the set of cups of ranks $r_1, r_2, \ldots r_m$.

As a tool in the analysis we define a new variant of the cup game: the ***negative-fill*** cup game. In the negative-fill cup game, when the emptier empties from a cup, the cup's fill always decreases by exactly 1, i.e. there is no zeroing out. We refer to the standard version of the cup game where cups can zero out as the ***standard-fill*** cup game when necessary for clarity.

The notion of negative fill will be useful in our lower-bound constructions, in which we want to construct a strategy for the filler that achieves large backlog. By analyzing a filling strategy on the negative-fill game, we can then reason about what happens if we apply the same filling strategy recursively to a set of cups S whose average fill μ is larger than 0; in the recursive application, the average fill μ acts as the "new 0", and fills less than μ act as negative fills.

Note that it is strictly easier for the filler to achieve high backlog in the standard-fill cup game than in the negative-fill cup game; hence a lower bound on backlog in the negative-fill cup game also serves as a lower bound on backlog in the standard-fill cup game. On the other hand, during the upper bound proof we use the standard-fill cup game: this makes it harder for the emptier to guarantee its upper bound.

Other Conventions. When discussing the state of the cups ***at a round t***, we will take it as given that we are referring to the starting state S_t of the round. Also, when discussing sets, we will use XY as a shorthand for $X \cup Y$. Finally, when discussing the average fill $\mu_{S_t}(X)$ of a set of cups, we will sometimes ommit the subscript S_t when the round number is clear.

3 Technical Overview

In this section we present proof sketches and discuss the main technical ideas for our results.

3.1 Adaptive Lower Bound

In Section 4 we provide an adaptive filling strategy that achieves backlog $\Omega(n^{1-\varepsilon})$; in this subsection we sketch the proof of the result.

First we note that there is a trivial algorithm, that we call **trivalg**, for achieving backlog at least 1/2 on at least 2 cups in time $O(1)$.

The essential ingredient in our lower-bound construction is what we call the ***Amplification Lemma***. The lemma takes as input a filling strategy that achieves some backlog curve f (i.e., on n cups, the strategy achieves backlog $f(n)$), and outputs a new filling strategy that achieves a new amplified backlog curve f'.

Lemma 3.1 (Lemma 4.1)**.** *Let* $\mathrm{alg}(f)$ *be a filling strategy that achieves backlog* $f(n)$ *on* n *cups (in the negative-fill cup game). There exists a filling strategy* $\mathrm{alg}(f')$*, the* ***amplification*** *of* $\mathrm{alg}(f)$*, that achieves backlog at least*

$$f'(n) \geq (1-\delta)f(\lfloor(1-\delta)n\rfloor) + f(\lceil\delta n\rceil).$$

Proof Sketch. The filler designates an ***anchor set*** A of size $\lceil\delta n\rceil$ and a ***non-anchor set*** B of size $\lfloor(1-\delta)n\rfloor$.

The filler's strategy begins with M phases, for some parameter M to be determined later. In each phase, the filler applies $\mathrm{alg}(f)$ to the non-anchor set B, while simultaneously placing 1 unit of water into each cup of A on each step. If there is ever a step during the phase in which the emptier does not remove water from every cup in A, then the phase is said to be ***emptier neglected***. On the other hand, if a phase is not emptier neglected, then at the end of the phase, the filler swaps the cup in B with largest fill with the cup in A whose fill is smallest.

After the M phases are complete, the filler then recursively applies $\mathrm{alg}(f)$ to the cups A. This completes the filling strategy $\mathrm{alg}(f')$.

The key to analyzing $\mathrm{alg}(f')$ is to show that, at the end of the M-th phase, the average fill of the cups A satisfies $\mu(A) \geq (1-\delta)f(|B|)$. This, in turn, means that the recursive application of $\mathrm{alg}(f)$ to A will achieve backlog $(1-\delta)f(|B|) + f(|A|)$, as desired.

Now let us reason about $\mu(A)$. If a phase is emptier neglected, then the total amount of water placed into A during the phase is at least 1 greater than the total amount of water removed. Hence $\mu(A)$ increases by at least $1/|A|$. On the other hand, if a phase is not emptier neglected, then $\mathrm{alg}(f)$ will successfully achieve backlog $\mu(B) + f(|B|)$ on the cups B during the phase. At the end of the phase, the filler will then swap a cup from B with large fill with a cup from A. Thus, in each phase, we either have that $\mu(A)$ increases by $1/|A|$, or that a cup with large fill gets swapped into A. After sufficiently many phases, one can show that $\mu(A)$ is guaranteed to become at least $(1-\delta)f(|B|) + f(|A|)$.

□

We use the Amplification Lemma to give two lower bounds on backlog: one with reasonable running time, the other with slightly better backlog.

Theorem 3.1 (Theorem 4.1)**.** *There is an adaptive filling strategy for achieving backlog* $\Omega(n^{1-\varepsilon})$ *for constant* $\varepsilon \in (0, 1/2)$ *in running time* $2^{O(\log^2 n)}$.

Proof Sketch. We construct a sequence of filling strategies with $\mathrm{alg}(f_{i+1})$ the amplification of $\mathrm{alg}(f_i)$ using $\delta = \Theta(1)$ determined as a function of ε, and $\mathrm{alg}(f_0) = \mathrm{trivalg}$. Choosing δ appropriately as a function of ε, and letting c be some (small) positive constant, we show by induction on i that, for all $k \leq 2^{ci}$, $\mathrm{alg}(f_i)$ achieves backlog $\Omega(k^{1-\varepsilon})$ on k cups in running time $2^{O(\log^2 k)}$. Taking $i = \Theta(\log n)$ completes the proof. □

Theorem 3.2 (Theorem 4.2)**.** *There is an adaptive filling strategy for achieving backlog* $\Omega(n)$ *in running time* $O(n!)$.

Proof Sketch. We construct a sequence of filling strategies with $\mathrm{alg}(f_{i+1})$ the amplification of $\mathrm{alg}(f_i)$ using $\delta = 1/(i+1)$, and $\mathrm{alg}(f_0)$ a filling strategy for achieving backlog 1 on $O(1)$ cups in $O(1)$ time (this is a slight modification of trivalg). We show by induction that $\mathrm{alg}(f_{\Theta(n)})$ achieves backlog $\Omega(n)$ in running time $O(n!)$. □

3.2 Upper Bound

In Section 5 we prove that a greedy emptier, i.e. an emptier that always empties from the p fullest cups, never lets backlog exceed $O(n)$; in this subsection we sketch the proof of this result.

Theorem 3.3 gives a system of invariants on the state of the cups after each step. By considering the invariant at $k = 1$, we achieve a bound of $O(n)$ on the backlog.

Theorem 3.3 (Theorem 5.1)**.** *For all $k \leq n$, the average fill of the k fullest cups never exceeds $2n - k$ at the beginning of any round.*

Proof Sketch. The proof is by induction on the round. Fix some round t and assume that the result holds for all k at the beginning of round t. Fix some k; we aim to prove that the average fill of the k fullest cups is at most $2n - k$ at the start of round $t + 1$.

Let A be the cups that satisfy the following three properties: they are among the k fullest cups in I_t, they are emptied from in step t, and they are among the k fullest cups in S_{t+1}. Let B be the cups that satisfy the following three properties: they are among the k fullest cups in state I_t, they are emptied from in step t, and are *not* among the k fullest cups in S_{t+1}. Finally, let C be the cups with ranks $|A| + |B| + 1, \ldots, k + |B|$ in state I_t. The set C is defined so that the k fullest cups in state S_{t+1} are given by AC.

For simplicity, throughout the rest of the proof we will make the following assumption: the rank r cup at state S_t is also the rank r cup at state I_t for all ranks $r \in [n]$. In the full version of the proof, we show that this assumption is actually without loss of generality.

We break the rest of the proof into three cases. Let $a = |A|$, $b = |B|$, and $c = |C|$.

Case 1: Some cup in A zeroes out in round t.
Analysis: The fill of all cups in C must be at most 1 at state I_t to be less than the fill of the cup in A that zeroed out. Furthermore, the average fill of A at the beginning of step $t + 1$ must be at most the average fill of the $a - 1$ fullest cups in S_t (due to one of the cups being zeroed out), and thus is at most $2n - (a - 1)$. Combined with some algebra, these facts imply that the average fill in AC in not too large, in particular not larger than $2n - k$.

Case 2: We have $b = 0$ and no cups in A zero out in round t.
Analysis: In this case the set of cups of ranks in $[k]$ at state S_t is the same as the set of cups of ranks in $[k]$ at state S_{t+1}, and these sets are both given by AC. During round t the emptier removes a units of fill from the cups A. The filler cannot have added more than k fill to the cups AC, because it can add at most 1 fill to any given cup. Also, the filler cannot have added more than p_t fill to the cups because this is the total amount of fill that the filler is allowed to add. Hence the filler adds at most $\min(p_t, k) = a + b = a + 0 = a$ fill to the cups AC. It follows that the emptier removes at least as many units of water from the cups AC as the filler adds, so the average fill of AC has not increased and is still at most $2n - k$.

Case 3: We have $b > 0$ and no cups in A zero out in round t.
Analysis: This is the most interesting of the three cases. Consider $m_{S_{t+1}}(AC)$, which is the mass of the k fullest cups after step t. Each cup in A was emptied from in step t, and the filler adds at most $\min(k, p_t) = a + b$ fill to cups AC during the step. Hence,

$$m_{S_{t+1}}(AC) \leq m_{S_t}(AC) + b. \tag{1}$$

Using the fact that $\mu_{S_t}(B) \geq \mu_{S_t}(C)$, we have that,

$$m_{S_t}(C) \leq \frac{c}{b+c} m_{S_t}(BC) = \frac{c}{b+c}(m_{S_t}(ABC) - m_{S_t}(A)).$$

Thus

$$m_{S_t}(AC) = m_{S_t}(A) + m_{S_t}(C) \leq \frac{c}{b+c} m_{S_t}(ABC) + \frac{b}{b+c} m_{S_t}(A). \tag{2}$$

The system of invariants at at the beginning of step t lets us bound $m_{S_t}(A)$ by $|A|(2n-|A|)$ and $m_{S_t}(ABC)$ by $|ABC|(2n-|ABC|)$, allowing for us to obtain a bound on $m_{S_t}(AC)$ in terms of a, b, c, namely,

$$m_{S_t}(AC) \leq \frac{c(a+b+c)(2n-a-b-c)}{b+c} + \frac{ba(2n-a)}{b+c}. \tag{3}$$

By algebraic manipulation, and substituting $k = a + c$, (3) reduces to

$$m_{S_t}(AC) \leq k(2n-k) - cb. \tag{4}$$

The transformation from (2) to (4) might at first seems somewhat mysterious; in the full version of the proof we also give an alternative version of the transformation that elucidates some of the underlying combinatorial structure.

Combined with (1), and the fact $c > 0$ (which follows from $b > 0$), (4) implies that $m_{S_{t+1}}(AC) \leq k(2n-k)$ and thus that the average fill of the k fullest cups in state S_{t+1} is at most $2n-k$, as desired. □

3.3 Oblivious Lower Bound

In Section 6 we consider what happens if the filler is an ***oblivious*** adversary, meaning that the filler cannot see what the emptier does at each step. The emptier, in turn, is permitted to use randomization in order to make its behavior unpredictable to the filler. We focus on randomized emptying algorithms that satisfy the so-called ***Δ-greedy-like*** property: the emptier never empties from a cup c over another cup c' whose fill is more than Δ greater than the fill of c.

The main theorem in Section 6 gives an oblivious filling strategy that achieves backlog $\Omega(n^{1-\varepsilon})$ against any Δ-greedy-like emptier for any $\Delta \in \Omega(1)$ (or, more precisely, any $\Delta \leq \frac{1}{128}\log\log\log n$).

Theorem 3.4 (Theorem 6.1)**.** *There is an oblivious filling strategy for the variable-processor cup game on N cups that achieves backlog at least $\Omega(N^{1-\varepsilon})$ for any constant $\varepsilon > 0$ in running time $2^{polylog(N)}$ with probability at least $1 - 2^{-\,polylog(N)}$ against a Δ-greedy-like emptier with $\Delta \leq \frac{1}{128}\log\log\log N$.*

Note that Theorem 3.4 uses N for the number of cups rather than using n. When describing the recursive strategy that the filler uses, we will use N to denote the true total number of cups, and n to denote the number of cups within the recursive subproblem currently being discussed.

The filling strategy used in Theorem 3.4 is closely related the adaptive filling strategy described in Section 3.1. The fact that the filling strategy must now be *oblivious*, however, introduces several new technical obstacles.

Problem 1: Distinguishing between neglected and non-neglected phases. Recall that the Amplification Lemma in Section 3.1 proceeds in phases, where the filler behaves differently at the end of each phase depending on whether or not the emptier ever neglected the anchor set A during that phase. If the filler is oblivious, however, then it cannot detect which phases are neglected.

To solve this problem, the first thing to notice is that the *total* number of times that the emptier can neglect the anchor set (within a given recursive subproblem of the Amplification Lemma) is, without loss of generality, at most N^2. Indeed, if the emptier neglects the anchor set more than N^2 times, then the total amount of water in cups A will be at least N^2. Since the amount of water in the system as a whole is non-decreasing, there will subsequently always be at least one cup in the system with fill N or larger, and thus the filler's strategy trivially achieves backlog N.

Assume that there are at most N^2 phases that the emptier neglects. The filler does not know which phases these are, and the filler does not wish to ever move a cup from the non-anchor set to the anchor set during a phase that the emptier neglected (since, during such a phase, there is no guarantee on the amount of water in the cup from B). To solve this problem, we increase the total number of phases in the Amplification Lemma to be some very large number $M = 2^{\text{polylog}\,N}$, and we have the filler select $|A|$ random phases at the end of which to move a cup from the non-anchor set to the anchor set. With high probability, none of the $|A|$ phases that the filler selects are neglected by the emptier.

Problem 2: Handling the probability of failure. Because the filler is now oblivious (and the emptier is randomized) the guarantee offered by the filling strategy is necessarily probabilistic. This makes the Amplification Lemma somewhat more difficult, since each application of $\text{alg}(f)$ now has some probability of failure.

We ensure that the applications of $\text{alg}(f)$ succeed with such high probability that we can ignore the possibility of any of them failing on phases when we need them to succeed. This necessitates making sure that the base-case construction $\text{alg}(f_0)$ succeeds with very high probability.

Fortunately, we can take a base-case construction alg_0 that succeeds with only constant (or even sub-constant) probability, and perform an Amplification-Lemma-like construction in order to obtain a new filling strategy alg_1 that achieves slightly *smaller* backlog, but that has a very high probability of succeeding.

To construct alg_1, we begin by performing the Amplification-Lemma construction on alg_0, but without recursing after the final phase. Even though many of the applications of alg_0 may fail, with high probability at least one application succeeds. This results in some cup c_* in A having high fill. Unfortunately, the filler does not know which cup has high fill, so it cannot simply take c_*. What the filler can do, however, is select some cup c, decrease the number of processors to 1, and then spend a large number of steps simply placing 1 unit of water into cup c in each step. By the Δ-greediness of the emptier, the emptier is guaranteed to focus on emptying from cup c_* (rather than cup c) until c attains large fill. This allows for the filler to obtain a cup c that the filler *knows* contains a large amount of water (with high probability). We use this approach to construct a base-case filling strategy $\text{alg}(f_0')$ that succeeds with high probability.

Problem 3: Non-flat starting states. The next problem that we encounter is that, at the beginning of any given phase, the cups in the non-anchor set may not start off with equal heights. Instead, some cups may contain very large amounts of water while others contain very small (and even negative) amounts of water[1]. This is not a problem for an adaptive filler, since the filler knows which cups contain small/large amounts of water, but it is a problem for an oblivious filler.

To avoid the scenario in which the cups in B are highly unequal, we begin each phase by first performing a ***flattening construction*** on the cups B, which causes the cups in B to all have roughly equal fills (up to $\pm O(\Delta)$). The flattening construction uses the fact that the emptier is Δ-greedy-like to ensure that cups which are overly full get flattened out by the emptier.

Putting the pieces together. By combining the ideas above, as well as handling other issues that arise (e.g., one must be careful to ensure that the average fills of A and B do not drift apart in unpredictable ways), one can prove Theorem 3.4. In Section 6, we give the full proof, which is by far the most technically involved result in the paper.

4 Adaptive Filler Lower Bound

In this section we give a $2^{\text{polylog}\, n}$-time filling strategy that achieves backlog $n^{1-\varepsilon}$ for any positive constant ε. We also give a $O(n!)$-time filling strategy that achieves backlog $\Omega(n)$.

We begin with a trivial filling strategy that we call **trivalg** that gives backlog at least $1/2$ when applied to at least 2 cups.

Proposition 4.1. *Consider an instance of the negative-fill* 1*-processor cup game on* n *cups, and let the cups start in any state with average fill is* 0*. If* $n \geq 2$*, there is an* $O(1)$*-step adaptive filling strategy* trivalg *that achieves backlog at least* $1/2$*. If* $n = 1$*,* trivalg *achieves backlog* 0 *in running time* 0*.*

Proof. If $n = 1$, trivalg does nothing and achieves backlog 0; for the rest of the proof we consider the case $n \geq 2$.

Let a and b be the fullest and second fullest cups in the in the starting configuration, and let their initial fills be $\text{fill}(a) = \alpha, \text{fill}(b) = \beta$. If $\alpha \geq 1/2$ the filler need not do anything, the desired backlog is already

[1]Recall that, in order for our lower-bound construction to be able to call itself recursively, we must analyze the construction in the negative-fill version of the variable-processor cup game.

achieved. Otherwise, if $\alpha \in [0, 1/2]$, the filler places $1/2 - \alpha$ fill into a and $1/2 + \alpha$ fill into b (which is possible as both fills are in $[0, 1]$, and they sum to 1). Since $\alpha + \beta \geq 0$ we have $\beta \geq -\alpha$. Clearly a and b now both have fill at least $1/2$. The emptier cannot empty from both a and b as $p = 1$, so even after the emptier empties from a cup we still have backlog $1/2$, as desired. □

Next we prove the ***Amplification Lemma***, which takes as input a filling strategy $\text{alg}(f)$ and outputs a new filling strategy $\text{alg}(f')$ that we call the ***amplification*** of $\text{alg}(f)$. $\text{alg}(f')$ is able to achieve higher fill than $\text{alg}(f)$; in particular, we will show that by starting with a filling strategy $\text{alg}(f_0)$ for achieving constant backlog and then forming a sufficiently long sequence of filling strategies $\text{alg}(f_0), \text{alg}(f_1), \ldots, \text{alg}(f_{i_*})$ with $\text{alg}(f_{i+1})$ the amplification of $\text{alg}(f_i)$, we get a filling strategy for achieving $\text{poly}(n)$ backlog.

Lemma 4.1 (Adaptive Amplification Lemma)**.** *Let $\delta \in (0, 1/2]$ be a parameter. Let $\text{alg}(f)$ be an adaptive filling strategy that achieves backlog $f(n) < n$ in the negative-fill variable-processor cup game on n cups in running time $T(n)$ starting from any initial cup state where the average fill is 0.*

Then there exists an adaptive filling strategy $\text{alg}(f')$ that achieves backlog $f'(n)$ satisfying

$$f'(n) \geq (1 - \delta) f(\lfloor (1 - \delta) n \rfloor) + f(\lceil \delta n \rceil)$$

and $f'(n) \geq f(n)$ in the negative-fill variable-processor cup game on n cups in running time

$$T'(n) \leq n \lceil \delta n \rceil \cdot T(\lfloor (1 - \delta) n \rfloor) + T(\lceil \delta n \rceil)$$

starting from any initial cup state where the average fill is 0.

Proof. Let $n_A = \lceil \delta n \rceil, n_B = n - n_A = \lfloor (1 - \delta) n \rfloor$.

The filler defaults to using $\text{alg}(f)$ if

$$f(n) \geq (1 - \delta) f(n_B) + f(n_A).$$

In this case using $\text{alg}(f)$ achieves the desired backlog in the desired running time. In the rest of the proof, we describe our strategy for the case where we cannot simply use $\text{alg}(f)$ to achieve the desired backlog.

Let A, the ***anchor set***, be initialized to consist of the n_A fullest cups, and let B the ***non-anchor set*** be initialized to consist of the rest of the cups (so $|B| = n_B$). Let $h = (1 - \delta) f(n_B)$.

The filler's strategy can be summarized as follows:

Step 1: Make $\mu(A) \geq h$ by using $\text{alg}(f)$ repeatedly on B to achieve cups with fill at least $\mu(B) + f(n_B)$ in B and then swapping these into A. While doing this the filler always places 1 unit of fill in each anchor cup.

Step 2: Use $\text{alg}(f)$ once on A to obtain some cup with fill $\mu(A) + f(n_A)$.
Note that in order to use $\text{alg}(f)$ on subsets of the cups the filler will need to vary p.

We now describe how to achieve Step 1, which is complicated by the fact that the emptier may attempt to prevent the filler from achieving high fill in a cup in B.

The filling strategy always places 1 unit of water in each anchor cup. This ensures that no cups in the anchor set ever have their fill decrease. If the emptier wishes to keep the average fill of the anchor cups from increasing, then emptier must empty from every anchor cup on each step. If the emptier fails to do this on a given round, then we say that the emptier has ***neglected*** the anchor cups.

We say that the filler ***applies*** $\text{alg}(f)$ to B if it follows the filling strategy $\text{alg}(f)$ on B while placing 1 unit of water in each anchor cup. An application of $\text{alg}(f)$ to B is said to be ***successful*** if A is never neglected during the application of $\text{alg}(f)$ to B. The filler uses a series of phases that we call ***swapping-processes*** to achieve the desired average fill in A. In a swapping-process, the filler repeatedly applies $\text{alg}(f)$ to B until a successful application occurs, and then takes the cup generated by $\text{alg}(f)$ within B on this successful application with fill at least $\mu(B) + f(|B|)$ and swaps it with the least full cup in A so long as the swap increases $\mu(A)$. If the average fill in A ever reaches h, then the algorithm immediately halts (even if it is in the middle of a swapping-process) and is complete.

Algorithm 1 Adaptive Amplification (Step 1)

Input: alg$(f), \delta$, set of n cups
Output: Guarantees that $\mu(A) \geq h$

```
A ← n_A fullest cups, B ← rest of the cups
Always place 1 fill in each cup in A
while μ(A) < h do                                   ▷ Swapping-Processes
    Immediately exit this loop if ever μ(A) ≥ h
    successful ← false
    while not successful do
        Apply alg(f) to B, alg(f) gives cup c
        if fill(c) ≥ h then
            successful ← true
    Swap c with least full cup in A
```

We give pseudocode for the filling strategy in Algorithm 1.

Note that

$$\mu(A) \cdot |A| + \mu(B) \cdot |B| = \mu(AB) \geq 0,$$

as $\mu(AB)$ starts as 0, but could become positive if the emptier skips emptyings. Thus we have

$$\mu(A) \geq -\mu(B) \cdot \frac{\lfloor (1-\delta)n \rfloor}{\lceil \delta n \rceil} \geq -\frac{1-\delta}{\delta}\mu(B).$$

Thus, if at any point B has average fill lower than $-h \cdot \delta/(1-\delta)$, then A has average fill at least h, so the algorithm is finished. Thus we can assume in our analysis that

$$\mu(B) \geq -h \cdot \delta/(1-\delta). \tag{5}$$

We will now show that the filler applies alg(f) to B at most hn_A total times. Each time the emptier neglects the anchor set, the mass of the anchor set increases by 1. If the emptier neglects the anchor set hn_A times, then the average fill in the anchor set increases by h. Since $\mu(A)$ started as at least 0, and since $\mu(A)$ never decreases (note in particular that cups are only swapped into A if doing so will increase $\mu(A)$), an increase of h in $\mu(A)$ implies that $\mu(A) \geq h$, as desired.

Consider the fill of a cup c swapped into A at the end of a swapping-process. Cup c's fill is at least $\mu(B) + f(n_B)$, which by (5) is at least

$$-h \cdot \frac{\delta}{1-\delta} + f(n_B) = (1-\delta)f(n_B) = h.$$

Thus the algorithm for Step 1 succeeds within $|A|$ swapping-processes, since at the end of the $|A|$-th swapping process either every cup in A has fill at least h, or the algorithm halted before $|A|$ swapping-processes because it already achieved $\mu(A) \geq h$.

After achieving $\mu(A) \geq h$, the filler performs Step 2, i.e. the filler applies alg(f) to A, and hence achieves a cup with fill at least

$$\mu(A) + f(|A|) \geq (1-\delta)f(n_B) + f(n_A),$$

as desired.

Now we analyze the running time of the filling strategy alg(f'). First, recall that in Step 1 alg(f') calls alg(f) on B, which has size n_B, as many as hn_A times. Because we mandate that $h < n$, Step 1 contributes no more than $(n \cdot n_A) \cdot T(n_B)$ to the running time. Step 2 requires applying alg(f) to A, which has size n_A, once, and hence contributes $T(n_A)$ to the running time. Summing these we have

$$T'(n) \leq n \cdot n_A \cdot T(n_B) + T(n_A).$$

□

We next show that by recursively using the Amplification Lemma we can achieve backlog $n^{1-\varepsilon}$.

Theorem 4.1. *There is an adaptive filling strategy for the variable-processor cup game on n cups that achieves backlog $\Omega(n^{1-\varepsilon})$ for any constant $\varepsilon > 0$ of our choice in running time $2^{O(\log^2 n)}$.*

Proof. Take constant $\varepsilon \in (0, 1/2)$. Let c, δ be constants that will be chosen (later) as functions of ε satisfying $c \in (0, 1), 0 < \delta \ll 1/2$. We show how to achieve backlog at least $cn^{1-\varepsilon} - 1$.

Let $\mathrm{alg}(f_0) = \mathrm{trivalg}$, the algorithm given by Proposition 4.1; recall trivalg achieves backlog $f_0(k) \geq 1/2$ for all $k \geq 2$, and $f_0(1) = 0$. Next, using the Amplification Lemma we recursively construct $\mathrm{alg}(f_{i+1})$ as the amplification of $\mathrm{alg}(f_i)$ for $i \geq 0$. Define a sequence g_i with

$$g_i = \begin{cases} \lceil 16/\delta \rceil, & i = 0, \\ \lfloor g_{i-1}/(1-\delta) \rfloor & i \geq 1. \end{cases}$$

We claim the following regarding this construction:

Claim 4.1. *For all $i \geq 0$,*

$$f_i(k) \geq ck^{1-\varepsilon} - 1 \quad \textit{for all} \quad k \in [g_i]. \tag{6}$$

Proof. We prove Claim 4.1 by induction on i. For $i = 0$, the base case, (6) can be made true by taking c sufficiently small; in particular, taking $c < 1$ makes (6) hold for $k = 1$, and, as $g_0 > 2$, taking c small enough to make $cg_0^{1-\varepsilon} - 1 \leq f_0(g_0) = 1/2$ makes (6) hold for $k \in [2, g_0]$ by monotonicity of $k \mapsto ck^{1-\varepsilon} - 1$[2].

As our inductive hypothesis we assume (6) for f_i; we aim to show that (6) holds for f_{i+1}. Note that, by design of g_i, if $k \leq g_{i+1}$ then $\lfloor k \cdot (1-\delta) \rfloor \leq g_i$. Consider any $k \in [g_{i+1}]$. First we deal with the trivial case where $k \leq g_0$. In this case

$$f_{i+1}(k) \geq f_i(k) \geq \cdots \geq f_0(k) \geq ck^{1-\varepsilon} - 1.$$

Now we consider the case where $k \geq g_0$. Since f_{i+1} is the amplification of f_i we have

$$f_{i+1}(k) \geq (1-\delta) f_i(\lfloor (1-\delta)k \rfloor) + f_i(\lceil \delta k \rceil).$$

By our inductive hypothesis, which applies as $\lceil \delta k \rceil \leq g_i, \lfloor k \cdot (1-\delta) \rfloor \leq g_i$, we have

$$f_{i+1}(k) \geq (1-\delta)(c \cdot \lfloor (1-\delta)k \rfloor^{1-\varepsilon} - 1) + c \lceil \delta k \rceil^{1-\varepsilon} - 1.$$

Dropping the floor and ceiling, incurring a -1 for dropping the floor, we have

$$f_{i+1}(k) \geq (1-\delta)(c \cdot ((1-\delta)k - 1)^{1-\varepsilon} - 1) + c(\delta k)^{1-\varepsilon} - 1.$$

Because $(x-1)^{1-\varepsilon} \geq x^{1-\varepsilon} - 1$, as $x \mapsto x^{1-\varepsilon}$ is a sub-linear sub-additive function, we have

$$f_{i+1}(k) \geq (1-\delta)c \cdot (((1-\delta)k)^{1-\varepsilon} - 2) + c(\delta k)^{1-\varepsilon} - 1.$$

Moving the $ck^{1-\varepsilon}$ to the front we have

$$f_{i+1}(k) \geq ck^{1-\varepsilon} \cdot \left((1-\delta)^{2-\varepsilon} + \delta^{1-\varepsilon} - \frac{2(1-\delta)}{k^{1-\varepsilon}} \right) - 1.$$

Because $(1-\delta)^{2-\varepsilon} \geq 1 - (2-\varepsilon)\delta$, a fact called Bernoulli's Identity, we have

$$f_{i+1}(k) \geq ck^{1-\varepsilon} \cdot \left(1 - (2-\varepsilon)\delta + \delta^{1-\varepsilon} - \frac{2(1-\delta)}{k^{1-\varepsilon}} \right) - 1.$$

Of course $-2(1-\delta) \geq -2$, so

$$f_{i+1}(k) \geq ck^{1-\varepsilon} \cdot \left(1 - (2-\varepsilon)\delta + \delta^{1-\varepsilon} - \frac{2}{k^{1-\varepsilon}} \right) - 1.$$

[2]Note that it is important here that ε and δ are constants, that way c is also a constant.

Because

$$\frac{-2}{k^{1-\varepsilon}} \geq \frac{-2}{g_0^{1-\varepsilon}} \geq -2(\delta/16)^{1-\varepsilon} \geq -\delta^{1-\varepsilon}/2,$$

which follows from our choice of $g_0 = \lceil 16/\delta \rceil$ and the restriction $\varepsilon < 1/2$, we have

$$f_{i+1}(k) \geq ck^{1-\varepsilon} \cdot \left(1 - (2-\varepsilon)\delta + \delta^{1-\varepsilon} - \delta^{1-\varepsilon}/2\right) - 1.$$

Finally, combining terms we have

$$f_{i+1}(k) \geq ck^{1-\varepsilon} \cdot \left(1 - (2-\varepsilon)\delta + \delta^{1-\varepsilon}/2\right) - 1.$$

Because $\delta^{1-\varepsilon}$ dominates δ for sufficiently small δ, there is a choice of $\delta = \Theta(1)$ such that

$$1 - (2-\varepsilon)\delta + \delta^{1-\varepsilon}/2 \geq 1.$$

Taking δ to be this small we have,

$$f_{i+1}(k) \geq ck^{1-\varepsilon} - 1,$$

completing the proof. We remark that the choices of c, δ are the same for every i in the inductive proof, and depend only on ε. □

To complete the proof, we will show that g_i grows exponentially in i. This implies that there exists $i_* \leq O(\log n)$ such that $g_{i_*} \geq n$, and hence we have an algorithm $\mathrm{alg}(f_{i_*})$ that achieves backlog $cn^{1-\varepsilon} - 1$ on n cups, as desired.

We lower bound the sequence g_i with another sequence g'_i defined as

$$g'_i = \begin{cases} 4/\delta, & i = 0 \\ g'_{i-1}/(1-\delta) - 1, & i > 0. \end{cases}$$

Solving this recurrence, we find

$$g'_i = \frac{4 - (1-\delta)^2}{\delta} \frac{1}{(1-\delta)^i} \geq \frac{1}{(1-\delta)^i},$$

which clearly exhibits exponential growth. In particular, let $i_* = \left\lceil \log_{1/(1-\delta)} n \right\rceil$. Then, $g_{i_*} \geq g'_{i_*} \geq n$, as desired.

Let the running time of $f_i(n)$ be $T_i(n)$. From the Amplification Lemma we have following recurrence bounding $T_i(n)$:

$$\begin{aligned} T_i(n) &\leq n \lceil \delta n \rceil \cdot T_{i-1}(\lfloor (1-\delta)n \rfloor) + T_{i-1}(\lceil \delta n \rceil) \\ &\leq 2n^2 T_{i-1}(\lfloor (1-\delta)n \rfloor). \end{aligned}$$

It follows that $\mathrm{alg}(f_{i_*})$, recalling that $i_* \leq O(\log n)$, has running time

$$T_{i_*}(n) \leq (2n^2)^{O(\log n)} \leq 2^{O(\log^2 n)},$$

as desired.

□

Now we provide a construction that can achieve backlog $\Omega(n)$ in very long games. The construction can be interpreted as the same argument as in Theorem 4.1 but with an extremal setting of δ to $\Theta(1/n)$[3].

[3] Or more precisely, setting δ in each level of recursion to be $\Theta(1/n)$, where n is the subproblem size; note in particular that δ changes between levels of recursion, which was not the case in the proof of Theorem 4.1.

Theorem 4.2. *There is an adaptive filling strategy that achieves backlog $\Omega(n)$ in time $O(n!)$.*

Proof. First we construct a slightly stronger version of trivalg that achieves backlog 1 on $n \geq n_0 = 8$ cups, instead of just backlog 1/2; this simplifies the analysis.

Claim 4.2. *There is a filling algorithm* trivalg_2 *that achieves backlog at least* 1 *on* $n_0 = 8$ *cups.*

Proof. Let trivalg_1 be the amplification of trivalg using $\delta = 1/2$. On 4 cups trivalg_1 achieves backlog at least $(1/2)(1/2) + 1/2 = 3/4$. Let trivalg_2 be the amplification of trivalg_1 using $\delta = 1/2$. On 8 cups trivalg_2 achieves backlog at least $(1/2)(3/4) + 3/4 \geq 1$. □

Let $\text{alg}(f_0) = \text{trivalg}_2$; we have $f_0(k) \geq 1$ for all $k \geq n_0$. For $i > 0$ we construct $\text{alg}(f_i)$ as the amplification of $\text{alg}(f_{i-1})$ using the Amplification Lemma with parameter $\delta = 1/(i+1)$.

We claim the following regarding this construction:

Claim 4.3. *For all $i \geq 0$,*

$$f_i((i+1)n_0) \geq \sum_{j=0}^{i} \left(1 - \frac{j}{i+1}\right). \tag{7}$$

Proof. We prove Claim 4.3 by induction on i. When $i = 0$, the base case, (7) becomes $f_0(n_0) \geq 1$ which is true. Assuming (7) for f_{i-1}, we now show (7) holds for f_i. Because f_i is the amplification of f_{i-1} with $\delta = 1/(i+1)$, we have by the Amplification Lemma

$$f_i((i+1) \cdot n_0) \geq \left(1 - \frac{1}{i+1}\right) f_{i-1}(i \cdot n_0) + f_{i-1}(n_0).$$

Since $f_{i-1}(n_0) \geq f_0(n_0) \geq 1$ we have

$$f_i((i+1) \cdot n_0) \geq \left(1 - \frac{1}{i+1}\right) f_{i-1}(i \cdot n_0) + 1.$$

Using the inductive hypothesis we have

$$f_i((i+1) \cdot n_0) \geq \left(1 - \frac{1}{i+1}\right) \sum_{j=0}^{i-1} \left(1 - \frac{j}{i}\right) + 1.$$

Note that

$$\left(1 - \frac{1}{i+1}\right) \cdot \left(1 - \frac{j}{i}\right) = \frac{i}{i+1} \cdot \frac{i-j}{i} = \frac{i-j}{i+1} = 1 - \frac{j+1}{i+1}.$$

Thus we have the desired bound:

$$f_i((i+1) \cdot n_0) \geq \sum_{j=1}^{i} \left(1 - \frac{j}{i+1}\right) + 1 = \sum_{j=0}^{i} \left(1 - \frac{j}{i+1}\right).$$

□

Let $i_* = \lfloor n/n_0 \rfloor - 1$, which by design satisfies $(i_* + 1)n_0 \leq n$. By Claim 4.3 we have

$$f_{i_*}((i_* + 1) \cdot n_0) \geq \sum_{j=0}^{i_*} \left(1 - \frac{j}{i_* + 1}\right) = i_*/2 + 1.$$

As $i_* = \Theta(n)$, we have thus shown that $\text{alg}(f_{i_*})$ can achieve backlog $\Omega(n)$ on n cups.

Let T_i be the running time of $\text{alg}(f_i)$. The recurrence for the running running time of f_{i_*} is

$$T_i(n) \leq n \cdot n_0 T_{i-1}(n - n_0) + O(1).$$

Clearly $T_{i_*}(n) \leq O(n!)$.

□

5 Upper Bound

In this section we analyze the ***greedy emptier***, which always empties from the p fullest cups. We prove in Corollary 5.1 that the greedy emptier prevents backlog from exceeding $O(n)$. In order to analyze the greedy emptier, we establish a system of invariants that hold at every step of the game.

The main result of the section is the following theorem[4].

Theorem 5.1. *In the variable-processor cup game on n cups, the greedy emptier maintains, at every step t, the invariants*

$$\mu_{S_t}(S_t([k])) \leq 2n - k \tag{8}$$

for all $k \in [n]$.

By applying Theorem 5.1 to the case of $k = 1$, we arrive at a bound on backlog:

Corollary 5.1. *In the variable-processor cup game on n cups, the greedy emptying strategy never lets backlog exceed $O(n)$.*

Proof of Theorem 5.1. We prove the invariants by induction on t. The invariants hold trivially for $t = 1$ (the base case for the inductive proof): the cups start empty so $\mu_{S_1}(S_1([k])) = 0 \leq 2n - k$ for all $k \in [n]$.

Fix a round $t \geq 1$, and any $k \in [n]$. We assume the invariant for all values of $k' \in [n]$ for state S_t (we will only explicitly use two of the invariants for each k, but the invariants that we need depend on the choice of p_t by the filler) and show that the invariant on the k fullest cups holds on round $t + 1$, i.e. that

$$\mu_{S_{t+1}}(S_{t+1}([k])) \leq 2n - k.$$

Note that because the emptier is greedy it always empties from the cups $I_t([p_t])$. Let A, with $a = |A|$, be $A = I_t([\min(k, p_t)]) \cap S_{t+1}([k])$; A consists of the cups that are among the k fullest cups in I_t, are emptied from, and are among the k fullest cups in S_{t+1}. Let B, with $b = |B|$, be $I_t([\min(k, p_t)]) \setminus A$; B consists of the cups that are among the k fullest cups in state I_t, are emptied from, and are not among the k fullest cups in S_{t+1}. Let $C = I_t(a + b + [k - a])$, with $c = k - a = |C|$; C consists of the cups with ranks $a + b + 1, \ldots, k + b$ in state I_t. The set C is defined so that $S_{t+1}([k]) = AC$, since once the cups in B are emptied from, the cups in B are not among the k fullest cups, so cups in C take their places among the k fullest cups.

Note that $k - a \geq 0$ as $a + b \leq k$, and also $|ABC| = k + b \leq n$, because by definition the b cups in B must not be among the k fullest cups in state S_{t+1} so there are at least $k + b$ cups. Note that $a + b = \min(k, p_t)$. We also have that $A = I_t([a])$ and $B = I_t(a + [b])$, as every cup in A must have higher fill than all cups in B in order to remain above the cups in B after 1 unit of water is removed from all cups in AB.

We now establish the following claim, which we call the ***interchangeability of cups***:

Claim 5.1. *There exists a cup state S'_t such that: (a) S'_t satisfies the invariants (8), (b) $S'_t(r) = I_t(r)$ for all ranks $r \in [n]$, and (c) the filler can legally place water into cups in order to transform S'_t into I_t.*

Proof. Fix $r \in [n]$. We will show that S_t can be transformed into a state S^r_t by relabelling only cups with ranks in $[r]$ such that (a) S^r_t satisfies the invariants (8), (b) $S^r_t([r]) = I_t([r])$ and (c) the filler can legally place water into cups in order to transform S^r_t into I_t.

Say there are cups x, y with $x \in S_t([r]) \setminus I_t([r]), y \in I_t([r]) \setminus S_t([r])$. Let the fills of cups x, y at state S_t be f_x, f_y; note that

$$f_x > f_y. \tag{9}$$

Let the amount of fill that the filler adds to these cups be $\Delta_x, \Delta_y \in [0, 1]$; note that

$$f_x + \Delta_x < f_y + \Delta_y. \tag{10}$$

[4]Recall that we use $\mu_S(X)$ and $m_S(X)$ to denote the average fill and the mass, respectively, of a set of cups X at state S. Note that in the lower bound proofs (i.e. Section 4 and Section 6) when we use the notation m (for mass) and μ (for average fill), we omit the subscript indicating the state at which the properties are measured. In those proofs the state is implicitly clear. However, in Section 5 it is necessary to make the state S explicit in the notation.

Define a new state S'_t where cup x has fill f_y and cup y has fill f_x. Note that the filler can transform state S'_t into state I_t by placing water into cups as before, except changing the amount of water placed into cups x and y to be $f_x - f_y + \Delta_x$ and $f_y - f_x + \Delta_y$, respectively.

In order to verify that the transformation from S'_t to I_t is a valid step for the filler, one must check three conditions. First, the amount of water placed by the filler is unchanged: this is because $(f_x - f_y + \Delta_x) + (f_y - f_x + \Delta_y) = \Delta_x + \Delta_y$. Second, the fills placed in cups x and y are at most 1: this is because $f_x - f_y + \Delta_x < \Delta_y \leq 1$ (by (10)) and $f_y - f_x + \Delta_x < \Delta_x \leq 1$ (by (9)). Third, the fills placed in cups x and y are non-negative: this is because $f_x - f_y + \Delta_x > \Delta_x \geq 0$ (by (9)) and $f_y - f_x + \Delta_y > \Delta_x \geq 0$ (by (10)).

We can repeatedly apply this process to swap each cup in $I_t([r]) \setminus S_t([r])$ into being in $S'_t([r])$. At the end of this process we will have some state S^r_t for which $S^r_t([r]) = I_t([r])$. Note that S^r_t is simply a relabeling of S_t, hence it must satisfy the same invariants (8) satisfied by S_t. Further, S^r_t can be transformed into I_t by a valid filling step.

Now we repeatedly apply this process, in descending order of ranks. In particular, we have the following process: create a sequence of states by starting with S^{n-1}_t, and to get to state S^r_t from state S^{r+1}_t apply the process described above. Note that S^{n-1}_t satisfies $S^{n-1}_t([n-1]) = I_t([n-1])$ and thus also $S^{n-1}_t(n) = I_t(n)$. If S^{r+1}_t satisfies $S^{r+1}_t(r') = I_t(r')$ for all $r' > r+1$ then S^r_t satisfies $S^r_t(r') = I_t(r')$ for all $r > r$, because the transition from S^{r+1}_t to S^r_t has not changed the labels of any cups with ranks in $(r+1, n]$, but the transition does enforce $S^r_t([r]) = I_t([r])$, and consequently $S^r_t(r+1) = I_t(r+1)$. We continue with the sequential process until arriving at state S^1_t in which we have $S^1_t(r) = I_t(r)$ for all r. Throughout the process each S^r_t has satisfied the invariants (8), so S^1_t satisfies the invariants (8). Further, throughout the process from each S^r_t it is possible to legally place water into cups in order to transform S^r_t into I_t.

Hence S^1_t satisfies all the properties desired, and the proof of Claim 5.1 is complete.

□

Claim 5.1 tells us that we may assume without loss of generality that $S_t(r) = I_t(r)$ for each rank $r \in [n]$. We will make this assumption for the rest of the proof.

In order to complete the proof of the theorem, we break it into three cases.

Claim 5.2. *If some cup in A zeroes out in round t, then the invariant $\mu_{S_{t+1}}(S_{t+1}([k])) \leq 2n - k$ holds.*

Proof. Say a cup in A zeroes out in step t. Of course

$$m_{S_{t+1}}(I_t([a-1])) \leq (a-1)(2n-(a-1))$$

because the $a-1$ fullest cups must have satisfied the invariant (with $k = a-1$) on round t. Moreover, because $\text{fill}_{S_{t+1}}(I_{t+1}(a)) = 0$

$$m_{S_{t+1}}(I_t([a])) = m_{S_{t+1}}(I_t([a-1])).$$

Combining the above equations, we get that

$$m_{S_{t+1}}(A) \leq (a-1)(2n-(a-1)).$$

Furthermore, the fill of all cups in C must be at most 1 at state I_t to be less than the fill of the cup in A that zeroed out. Thus,

$$\begin{aligned} m_{S_{t+1}}(S_{t+1}([k])) &= m_{S_{t+1}}(AC) \\ &\leq (a-1)(2n-(a-1)) + k - a \\ &= a(2n-a) + a - 2n + a - 1 + k - a \\ &= a(2n-a) + (k-n) + (a-n) - 1 \\ &< a(2n-a) \end{aligned}$$

as desired. As k increases from 1 to n, $k(2n-k)$ strictly increases (it is a quadratic in k that achieves its maximum value at $k = n$). Thus $a(2n-a) \leq k(2n-k)$ because $a \leq k$. Therefore,

$$m_{S_{t+1}}(S_{t+1}([k])) \leq k(2n-k).$$

□

Claim 5.3. *If no cups in A zero out in round t and $b = 0$, then the invariant $\mu_{S_{t+1}}(S_{t+1}([k])) \leq 2n - k$ holds.*

Proof. If $b = 0$, then $S_{t+1}([k]) = S_t([k])$. During round t the emptier removes a units of fill from the cups in $S_t([k])$, specifically the cups in A. The filler cannot have added more than k fill to these cups, because it can add at most 1 fill to any given cup. Also, the filler cannot have added more than p_t fill to the cups because this is the total amount of fill that the filler is allowed to add. Hence the filler adds at most $\min(p_t, k) = a + b = a + 0 = a$ fill to these cups. Thus the invariant holds:

$$m_{S_{t+1}}(S_{t+1}([k])) \leq m_{S_t}(S_t([k])) + a - a \leq k(2n - k).$$

□

The remaining case, in which no cups in A zero out and $b > 0$ is the most technically interesting.

Claim 5.4. *If no cups in A zero out on round t and $b > 0$, then the invariant $\mu_{S_{t+1}}(S_{t+1}([k])) \leq 2n - k$ holds.*

Proof. Because $b > 0$ and $a + b \leq k$ we have that $a < k$, and $c = k - a > 0$. Recall that $S_{t+1}([k]) = AC$, so the mass of the k fullest cups at S_{t+1} is the mass of AC at S_t plus any water added to cups in AC by the filler, minus any water removed from cups in AC by the emptier. The emptier removes exactly a units of water from AC. The filler adds no more than p_t units of water to AC (because the filler adds at most p_t total units of water per round) and the filler also adds no more than $k = |AC|$ units of water to AC (because the filler adds at most 1 unit of water to each of the k cups in AC). Thus, the filler adds no more than $a + b = \min(p_t, k)$ units of water to AC. Combining these observations we have:

$$m_{S_{t+1}}(S_{t+1}([k])) \leq m_{S_t}(AC) + b. \tag{11}$$

The key insight necessary to bound this is to notice that larger values for $m_{S_t}(A)$ correspond to smaller values for $m_{S_t}(C)$ because of the invariants; the higher fill in A ***pushes down*** the fill that C can have. By capturing the pushing-down relationship combinatorially we will achieve the desired inequality.

We can upper bound $m_{S_t}(C)$ by

$$\begin{aligned} m_{S_t}(C) &\leq \frac{c}{b+c} m_{S_t}(BC) \\ &= \frac{c}{b+c}(m_{S_t}(ABC) - m_{S_t}(A)) \end{aligned}$$

because $\mu_{S_t}(C) \leq \mu_{S_t}(B)$ without loss of generality by the interchangeability of cups. Thus we have

$$m_{S_t}(AC) \leq m_{S_t}(A) + \frac{c}{b+c} m_{S_t}(BC) \tag{12}$$

$$= \frac{c}{b+c} m_{S_t}(ABC) + \frac{b}{b+c} m_{S_t}(A). \tag{13}$$

Note that the expression in (13) is monotonically increasing in both $\mu_{S_t}(ABC)$ and $\mu_{S_t}(A)$. Thus, by numerically replacing both average fills with their extremal values, $2n - |ABC|$ and $2n - |A|$. At this point the claim can be verified by straightforward (but quite messy) algebra (and by combining (11) with (13)). We instead give a more intuitive argument, in which we examine the right side of (12) combinatorially.

Consider a new configuration of fills F achieved by starting with state S_t, and moving water from BC into A until $\mu_F(A) = 2n - |A|$. [5] This transformation increases (strictly increases if and only if we move a non-zero amount of water) the right side of (12). In particular, if mass $\Delta \geq 0$ fill is moved from BC to A, then the right side of (12) increases by $\frac{b}{b+c}\Delta \geq 0$. Note that the fact that moving water from BC into A increases the right side of (12) formally captures the way the system of invariants being proven forces a

[5] Note that whether or not F satisfies the invariants is irrelevant.

tradeoff between the fill in A and the fill in BC—that is, higher fill in A pushes down the fill that BC (and consequently C) can have.

Since $\mu_F(A)$ is above $\mu_F(ABC)$, the greater than average fill of A must be counter-balanced by the lower than average fill of BC. In particular we must have

$$(\mu_F(A) - \mu_F(ABC))|A| = (\mu_F(ABC) - \mu_F(BC))|BC|.$$

Note that

$$\begin{aligned}
&\mu_F(A) - \mu_F(ABC) \\
&= (2n - |A|) - \mu_F(ABC) \\
&\geq (2n - |A|) - (2n - |ABC|) \\
&= |BC|.
\end{aligned}$$

Hence we must have

$$\mu_F(ABC) - \mu_F(BC) \geq |A|.$$

Thus

$$\mu_F(BC) \leq \mu_F(ABC) - |A| \leq 2n - |ABC| - |A|. \tag{14}$$

Combing (12) with the fact that the transformation from S_t to F only increases the right side of (12), along with (14), we have the following bound:

$$\begin{aligned}
m_{S_t}(AC) &\leq m_F(A) + c\mu_F(BC) \\
&\leq a(2n - a) + c(2n - |ABC| - a) \\
&\leq (a + c)(2n - a) - c(a + c + b) \\
&\leq (a + c)(2n - a - c) - cb. \qquad (15)
\end{aligned}$$

By (11) and (15), we have that

$$\begin{aligned}
m_{S_{t+1}}(S_{t+1}([k])) &\leq m_{S_t}(AC) + b \\
&\leq (a + c)(2n - a - c) - cb + b \\
&= k(2n - k) - cb + b \\
&\leq k(2n - k),
\end{aligned}$$

where the final inequality uses the fact that $c \geq 1$. This completes the proof of the claim.

□

We have shown the invariant holds for arbitrary k, so given that the invariants all hold at state S_t they also must all hold at state S_{t+1}. Thus, by induction we have the invariant for all rounds $t \in \mathbb{N}$. □

6 Oblivious Filler Lower Bound

In this section we prove that, surprisingly, an oblivious filler can achieve backlog $n^{1-\varepsilon}$ against an arbitrary "greedy-like" emptying algorithm.

We say an emptier is ***Δ-greedy-like*** if, whenever there are two cups with fills that differ by at least Δ, the emptier never empties from the less full cup without also emptying from the more full cup. That is, if on some round t, there are cups c_1, c_2 with $\text{fill}_{I_t}(c_1) > \text{fill}_{I_t}(c_2) + \Delta$, then a Δ-greedy-like emptier doesn't empty from c_2 on round t unless it also empties from c_1 on round t. Note that a perfectly greedy emptier is 0-greedy-like. We call an emptier ***greedy-like*** if in the cup game on n cups it is Δ-greedy-like for $\Delta \leq \frac{1}{128} \log\log\log n$. The main result of this section is an oblivious filling strategy that achieves large backlog against any (possibly randomized) greedy-like emptier.

As a tool in our analysis we define a new variant of the cup game: In the p-processor ***E-extra-emptyings S-skip-emptyings*** negative-fill cup game on n cups, the filler distributes p units of water amongst the cups, but the filler may empty from p' cups for some $p' \neq p$. In particular the emptier is allowed to do E extra emptyings and is also allowed to skip S emptyings over the course of the game. Note that the emptier still cannot empty from the same cup twice on a single round, and also that note that a Δ-greedy-like emptier must take into account extra emptyings and skip emptyings to determine valid moves. Further, note that the emptier is allowed to skip extra emptyings, although skipping extra emptyings looks the same as if the extra-emptyings had simply not been performed. Let the ***regular*** cup game be the 0-extra-emptyings ∞-skip-emptyings cup game: this is the cup game we usually consider. Allowing for some extra emptyings, and bounding the number of skip emptyings is sometimes necessary when analyzing an algorithm that is a subroutine of a larger algorithm however, hence it sometimes makes sense to consider games with different values of E, S. Unless explicitly stated otherwise however we are considering the regular cup game.

The ***fill-range*** of a set of cups at a state S is $\max_c \text{fill}_S(c) - \min_c \text{fill}_S(c)$. We call a cup configuration ***R-flat*** if the fill-range of the cups less than or equal to R; note that in an R-flat cup configuration with average fill 0 all cups have fills in $[-R, R]$.

For a Δ-greedy-like emptier let $R_\Delta = 2(2 + \Delta)$; we now prove a key property of these emptiers: there is an oblivious filling strategy, which we term **flatalg**, that attains an R_Δ-flat cup configuration against a Δ-greedy-like emptier, given cups of a known starting fill-range.

Lemma 6.1. *Consider an R-flat cup configuration in the p-processor E-extra-emptyings S-skip-emptyings negative-fill cup game on $n = 2p$ cups. There is an oblivious filling strategy* **flatalg** *that achieves an R_Δ-flat configuration of cups against a Δ-greedy-like emptier in running time $O(R + E + S)$. Furthermore,* flatalg *guarantees that the cup configuration is R-flat on every round.*

Proof. If $R \leq R_\Delta$ the algorithm does nothing, since the desired fill-range is already achieved; for the rest of the proof we consider $R > R_\Delta$.

The filler's strategy is to distribute fill equally amongst all cups at every round, placing $p/n = 1/2$ fill in each cup. Let $\ell_t = \min_{c \in S_t} \text{fill}_{S_t}(c)$, $u_t = \max_{c \in S_t} \text{fill}_{S_t}(c)$.

First we show that the fill-range of the cups can only increase if the fill-range is very small.

Claim 6.1. *If the fill-range on round $t + 1$ is larger than the fill-range on round t, then $u_{t+1} - \ell_{t+1} \leq R_\Delta$.*

Proof. First we remark that the fill of any cup changes by at most $1/2$ from round to round, and in particular $|u_{t+1} - u_t| \leq 1/2$, $|\ell_{t+1} - \ell_t| \leq 1/2$. In order for the fill-range to increase, the emptier must have emptied from some cup with fill in $[\ell_t, \ell_t + 1]$ without emptying from some cup with fill in $[u_t - 1, u_t]$; if the emptier had not emptied from every cup with fill in $[\ell_t, \ell_t + 1]$ then we would have $\ell_{t+1} = \ell_t + 1/2$ so the fill-range cannot have increased, and similarly if the emptier had emptied from every cup with fill in $[u_t - 1, u_t]$ then we would have $u_{t+1} = u_t - 1/2$ so again the fill-range cannot have increased. Because the emptier is Δ-greedy-like emptying from a cup with fill at most $\ell_t + 1$ and not emptying from a cup with fill at least $u_t - 1$ implies that $u_t - 1$ and $\ell_t + 1$ differ by at most Δ. Thus,

$$u_{t+1} - \ell_{t+1} \leq u_t + 1/2 - (\ell_t - 1/2) \leq \Delta + 3 \leq R_\Delta.$$

□

Claim 6.1 implies that if the fill-range is at most R_Δ it will stay at most R_Δ, because fill-range cannot increase to a value larger than R_Δ. Claim 6.1 also implies that until the fill-range is less than R_Δ the fill-range must not increase. However the claim does not preclude fill-range from remaining constant for many rounds, or decreasing, but only by very small amounts. For this reason we actually do not use Claim 6.1 in the remainder of the proof; nonetheless, the fact that the fill-range cannot increase relative to initial fill-range during flatalg is an important property of flatalg. In the rest of the proof we establish that the fill-range indeed must eventually be at most R_Δ.

Let L_t be the set of cups c with $\text{fill}_{S_t}(c) \leq \ell_t + 2 + \Delta$, and let U_t be the set of cups c with $\text{fill}_{S_t}(c) \geq u_t - 2 - \Delta$. We now prove a key property of the sets U_t and L_t: if a cup is in U_t or L_t it is also in $U_{t'}, L_{t'}$ for all $t' > t$. This follows immediately from Claim 6.2.

Claim 6.2. $U_t \subseteq U_{t+1}, L_t \subseteq L_{t+1}$.

Proof. Consider a cup $c \in U_t$.

If c is not emptied from, i.e. $\text{fill}(c)$ has increased by $1/2$ from the previous round, then clearly $c \in U_{t+1}$, because backlog has increased by at most $1/2$, so $\text{fill}(c)$ must still be within $2 + \Delta$ of the backlog on round $t + 1$.

On the other hand, if c is emptied from, i.e. $\text{fill}(c)$ has decreased by $1/2$, we consider two cases.

Case 1: If $\text{fill}_{S_t}(c) \geq u_t - \Delta - 1$, then $\text{fill}_{S_t}(c)$ is at least 1 above the bottom of the interval defining which cups belong to U_t. The backlog increases by at most $1/2$ and the fill of c decreases by $1/2$, so $\text{fill}_{S_{t+1}}(c)$ is at least $1 - 1/2 - 1/2 = 0$ above the bottom of the interval, i.e. still in the interval.

Case 2: On the other hand, if $\text{fill}_{S_t}(c) < u_t - \Delta - 1$, then every cup with fill in $[u_t - 1, u_t]$ must have been emptied from because the emptier is Δ-greedy-like. Therefore the fullest cup on round $t + 1$ is the same as the fullest cup on round t, because every cup with fill in $[u_t - 1, u_t]$ has had its fill decrease by $1/2$, and no cup with fill less than $u_t - 1$ had its fill increase by more than $1/2$. Hence $u_{t+1} = u_t - 1/2$. Because both $\text{fill}(c)$ and the backlog have decreased by $1/2$, the distance between them is still at most $\Delta + 2$, hence $c \in U_{t+1}$.

The argument for why $L_t \subseteq L_{t+1}$ is symmetric. □

Now we show that under certain conditions u_t decreases and ℓ_t increases.

Claim 6.3. *On any round t where the emptier empties from at least $n/2$ cups, if $|U_t| \leq n/2$ then $u_{t+1} = u_t - 1/2$. On any round t where the emptier empties from at most $n/2$ cups, if $|L_t| \leq n/2$ then $\ell_{t+1} = \ell_t + 1/2$.*

Proof. Consider a round t where the emptier empties from at least $n/2$ cups. If there are at least $n/2$ cups outside of U_t, i.e. cups with fills in $[\ell_t, u_t - 2 - \Delta]$, then all cups with fills in $[u_t - 2, u_t]$ must be emptied from; if one such cup was not emptied from then by the pigeon-hole principle some cup outside of U_t was emptied from, which is impossible as the emptier is Δ-greedy-like. This clearly implies that $u_{t+1} = u_t - 1/2$: no cup with fill less than $u_t - 2$ has gained enough fill to become the fullest cup, and the fullest cup from the previous round has lost $1/2$ unit of fill.

By a symmetric argument, $\ell_{t+1} = \ell_t + 1/2$ on a round t where the emptier empties at most $n/2$ cups and $|L_t| \leq n/2$. □

Now we show that eventually $L_t \cap U_t \neq \varnothing$.

Claim 6.4. *There is a round $t_0 \leq O(R + E + S)$ such that $U_t \cap L_t \neq \varnothing$ for all $t \geq t_0$.*

Proof. We call a round where the emptier empties from $p' \neq p$ cups an ***unbalanced round***; we call a round that is not unbalanced a ***balanced*** round.

Note that there are clearly at most $E + S$ unbalanced rounds. We now associate some unbalanced rounds with balanced rounds; in particular we define what it means for a balanced round to ***cancel*** an unbalanced round. Let $B = 2(R + \lceil (1 + 1/n)(E + S) \rceil)$. For $i = 1, 2, \ldots, B$ (iterating in ascending order of i), if round i is unbalanced then we say that the first balanced round $j > i$ that hasn't already been assigned (earlier in the sequential process) to cancel another unbalanced round $i' < i$, if any such round j exists, ***cancels*** round i. Note that cancellation is a one-to-one relation: each unbalanced round is cancelled by at most one balanced round and each balanced round cancels at most one unbalanced round. We say a balanced round j is ***cancelling*** if it cancels some unbalanced round $i < j$, and ***non-cancelling*** if it does not cancel any unbalanced round.

We claim that there is some $i \in [2(E + S) + 1]$ such that among the rounds $[B + 2(E + S) + i - 1]$ every unbalanced round has been cancelled by a balanced round in $[B + 2(E + S) + i - 1]$, and such that there are B non-cancelling balanced rounds. Note that there are at least $B + (E + S)$ balanced rounds in the first $B + 2(E + S)$ rounds, and thus there are at least B non-cancelling balanced rounds among the first $B + 2(E + S)$ rounds, due to there being at most $E + S$ total unbalanced rounds. Next note that there must be at least 1 non-cancelling balanced round among any set of $2(E + S) + 1$ rounds, because there cannot be more than $E + S$ unbalanced rounds, and hence there also cannot be more than $E + S$ cancelling

rounds. Thus there is some $i \in [2(E+S)+1]$ such that round $B+2(E+S)+i-1$ is balanced and non-cancelling. For this i, we have that all unbalanced rounds in $[B+2(E+S)+i-1]$ are cancelled, and since $B+2(E+S)+i-1 \geq B+2(E+S)$, we have that there are at least B balanced non-cancelling rounds in $[B+2(E+S)+i-1]$. These are the desired properties.

Let t_e be the first round by which there are $B = 2(R+\lceil (E+S)/n \rceil)$ balanced non-cancelling rounds; we have shown that $t_e \leq O(R+E+S)$. Note that the average fill of the cups cannot have decreased by more than E/n from its starting value; similarly the average fill of the cups cannot have increased by more than S/n. Because the cups start R-flat, u_t cannot have decreased by more than $R+E/n$ or else backlog would be less than average fill, and identically ℓ_t cannot have increased by more than $R+S/n$ or else anti-backlog would be larger than average fill. Now, by Claim 6.3 we have that for some $t \leq t_e$, $|L_t| > n/2$: if $|L_t| \leq n/2$ were always true for $t \leq t_e$, then on every balanced round ℓ_t would have increased by $1/2$, and since ℓ_t increases by at most $1/2$ on unbalanced rounds, this implies that in total ℓ_t would have increased by at least $(1/2)2(R+\lceil (E+S)/n \rceil)$, which is impossible. By a symmetric argument it is impossible that $|U_t| \leq n/2$ for all rounds.

Since $|U_{t+1}| \geq |U_t|$ and $|L_{t+1}| \geq |L_t|$ by Claim 6.2, we have that there is some round $t_0 \leq t_e$ such that for all $t \geq t_0$ we have $|U_t| > n/2$ and $|L_t| > n/2$. But then we have $U_t \cap L_t \neq \varnothing$, as desired. □

If there exists a cup $c \in L_t \cap U_t$, then

$$\text{fill}(c) \in [u_t - 2 - \Delta, u_t] \cap [\ell_t, \ell_t + 2 + \Delta].$$

Hence we have that

$$\ell_t + 2 + \Delta \geq u_t - 2 - \Delta.$$

Rearranging,

$$u_t - \ell_t \leq 2(2+\Delta) = R_\Delta.$$

Thus the cup configuration is R_Δ-flat by the end of $O(R+E+S)$ rounds.

□

Next we describe a simple oblivious filling strategy, that we call **randalg**, that will be used as a subroutine in Lemma 6.2; variants of this strategy are well-known, and similar versions of it can be found in [10, 16, 25].

Proposition 6.1. *Consider an R-flat cup configuration in the regular single-processor ∞-extra-emptyings ∞-skip-emptyings negative-fill cup game on n cups with initial average fill μ_0. Let $k \in [n]$ be a parameter. Let $d = \sum_{i=2}^{k} 1/i$.*

There is an oblivious filling strategy **randalg(k)** *with running time $k-1$ that achieves fill at least $\mu_0 - R + d$ in a known cup c with probability at least $1/k!$ if we condition on the emptier not performing extra emptyings.* randalg(k) *achieves fill at most $\mu_0 + R + d$ in this cup (unconditionally).*

Furthermore, when applied against a Δ-greedy-like emptier with $R = R_\Delta$, randalg(k) *guarantees that the cup configuration is $(R+d)$-flat on every round (unconditionally).*

Proof. First we condition on the emptier not using extra emptying and show that in this case the filler has probability at least $1/(k-1)!$ (which we lower bound by $1/k!$ for sake of simplicity) of attaining a cup with fill at least $\mu_0 - R + d$. The filler maintains an ***active set***, initialized to being an arbitrary subset of k of the cups. Every round the filler distributes 1 unit of fill equally among all cups in the active set. Next the emptier removes 1 unit of fill from some cup, or skips its emptying. Then the filler removes a random cup from the active set (chosen uniformly at random from the active set). This continues until a single cup c remains in the active set.

We now bound the probability that c has never been emptied from. Assume that on the i-th step of this process, i.e. when the size of the active set is $n-i+1$, no cups in the active set have ever been emptied from; consider the probability that after the filler removes a cup randomly from the active set there are still no cups in the active set that the emptier has emptied from. If the emptier skips its emptying on this round, or empties from a cup not in the active set then it is trivially still true that no cups in the active set

have been emptied from. If the cup that the emptier empties from is in the active set then with probability $1/(k-i+1)$ it is evicted from the active set, in which case we still have that no cup in the active set has ever been emptied from. Hence with probability at least $1/(k-1)!$ the final cup in the active set, c, has never been emptied from. In this case, c will have gained fill $d = \sum_{i=2}^{k} 1/i$ as claimed. Because c started with fill at least $-R+\mu_0$, c now has fill at least $-R+d+\mu_0$.

Now note that regardless of if the emptier uses extra emptyings c has fill at most $\mu_0 + R + d$, as c starts with fill at most R, and c gains at most $1/(k-i+1)$ fill on the i-th round of this process.

Now we analyze this algorithm specifically for a Δ-greedy-like emptier. Let $\mathcal{A}_t$ be the event that the anti-backlog is smaller in S_{t+1} than in S_t, let $\mathcal{B}_t$ be the event that some cup with fill equal to the backlog in S_{t+1} was emptied from on round t. If $\mathcal{A}_t$ and $\mathcal{B}_t$ are both true on round t, then by greediness the cups are quite flat. In particular, let a be a cup with fill equal to the anti-backlog in state S_{t+1} that was emptied from on round t, and let b be a cup with fill equal to the backlog in state S_{t+1} that was not emptied from on round t. By greediness $\text{fill}_{I_t}(a) + \Delta > \text{fill}_{I_t}(b)$. Of course $\text{fill}_{I_t}(b) = \text{fill}_{S_{t+1}}(b)$, and for b to have fill equal to the backlog on round $t+1$, b must have fill less than 1 below backlog on round t. Of course $\text{fill}_{I_t}(a) \leq \text{fill}_{S_t}(a) + 1$, and for a to have fill equal to the anti-backlog on round $t+1$, a must have fill less than 1 above the anti-backlog on round t. Thus we have that the backlog and anti-backlog differ by at most $\Delta + 3 \leq R_\Delta$ on round t, i.e. the cups are R_Δ-flat.

Consider a round t_1 where the cups are not R_Δ-flat. Let t_0 be the last round that the cups were R_Δ-flat. On all rounds $t \in (t_0, t_1)$ at least one of $\mathcal{A}_t$ or $\mathcal{B}_t$ must not hold. On a round where $\mathcal{A}_t$ does not hold, anti-backlog does not decrease and backlog increases by at most $1/(k-t+1)$, so fill range increases by at most $1/(k-t+1)$. On a round where $\mathcal{B}_t$ does not hold, anti-backlog decreases by at most 1 and backlog decreases by at least $1 - 1/(k-t+1)$, as all cups with fill equal to the backlog in state S_{t+1} were emptied from on round t, so fill-range increases by at most $1/(k-t+1)$. Hence in total fill-range increases by at most $\sum_{i=2}^{k} 1/i$ from R, i.e. the cups are $(R+d)$-flat on round t_1.

□

We now give a general algorithm that specifies a useful transformation of a filling strategy into a new filling strategy by repeatedly applying the strategy to subsets of the cups. The procedure is similar to the procedure used in the Adaptive Amplification Lemma, although more complicated.

Definition 6.1. Let alg_0 be an oblivious filling strategy, that can get high fill (for some definition of high) in some cup against greedy-like emptiers with some probability. We construct a new filling strategy $\mathbf{rep_\delta(alg_0)}$ (rep stands for "repetition") as follows:

Say we have some configuration of $n \leq N$ cups (recall that eventually we aim to get large backlog in N cups). Let $n_A = \lceil \delta n \rceil, n_B = \lfloor (1-\delta)n \rfloor$. Let $M = 2^{\text{polylog}(N)}$ be a chosen parameter. Initialize A to $\varnothing$ and B to being all of the cups. We call A the ***anchor set*** and B the ***non-anchor set***. The filler always places 1 unit of fill in each anchor cup on each round. The filling strategy consists of n_A ***donation-processes***, which are procedures that result in a cup being ***donated*** from B to A (i.e. removed from B and added to A). At the start of each donation-processes the filler chooses a value m_0 uniformly at random from $[M]$. We say that the filler ***applies*** a filling strategy alg to B if the filler uses alg on B while placing 1 unit of fill in each anchor cup. During the donation-process the filler applies alg_0 to B m_0 times, and flattens B by applying flatalg to B for $\Theta(N^2)$ rounds before each application of alg_0. At the end of each donation process the filler takes the cup given by the final application of alg_0 (i.e. the cup that alg_0 guarantees with some probability against a certain class of emptiers to have a certain high fill), and donates this cup to A.

We give pseudocode for this algorithm in Algorithm 2.

We say that the emptier ***neglects*** the anchor set on a round if it does not empty from each anchor cup. We say that an application of alg_0 to B is ***non-emptier-wasted*** if the emptier does not neglect the anchor set during any round of the application of alg_0.

We use the idea of repeating an algorithm in two different contexts. First in Proposition 6.2 we prove a result analogous to that of Proposition 4.1. In particular, we show that we can achieve $\omega(1)$ fill in a known cup (with good probability) by using $\text{rep}_\delta(\text{randalg}(k))$ (for appropriate choice of δ, k) to get large fill in an

Algorithm 2 $\text{rep}_\delta(\text{alg}_0)$

Input: $\text{alg}_0, \delta, N, M$, set of n cups
Output: Guarantees on the sets A, B (will vary based on alg_0)

```
n_A ← ⌈δn⌉ , n_B ← ⌊(1 − δ)n⌋
A ← ∅, B ← all the cups
Always place 1 unit of fill into each cup in A
for i ∈ [n_A] do                                   ▷ Donation-Processes
    m_0 ← random([M])
    for j ∈ [m_0] do
        Apply flatalg to B for Θ(N^2) rounds
        Apply alg_0 to B
    Donate the cup given by alg_0 from B to A
```

unknown cup, and then and then exploiting the emptier's greedy-like nature to get (slightly smaller) fill in a known cup. Second, we use rep in proving the ***Oblivious Amplification Lemma***, a result analogous to the Adaptive Amplification Lemma. In particular, we show how to take an algorithm for achieving some backlog, and then achieve higher backlog by repeating the algorithm many times. Although these results have deterministic analogues, their proofs are different and significantly more complex than the corresponding proofs for the deterministic results.

In the rest of the section our aim is to achieve backlog $N^{1-\varepsilon}$ in N cups. We will use this value N within all of the following proofs. We use N to refer to the true number of cups, and n refer to the size of the current subproblem that we are considering, which is implicitly part of a larger cup game. One benefit of making the true number of cups explicit is that if in a subproblem we ever get mass N^2 in a set of cups, then we are done with the entire construction, as backlog will always be at least N after this. Note that rather than using the negative-fill cup game we opt to explicitly make our results be in terms of the average fill of the cups. Another thing to note is that we are assuming that N is sufficiently large for all of the results, as is standard with asymptotic notation.

Before proving Proposition 6.2 we analyze $\text{rep}_\delta(\text{randalg}(k))$ in Lemma 6.2. We remark that although Proposition 6.2 and Lemma 6.2 do implicitly consider a small cup game that is part of a larger cup game, we do not allow for extra-emptyings in the small cup game. That is, if there are extra-emptyings, then we provide no guarantees on the behavior of the algorithms given in Proposition 6.2 and Lemma 6.2.

Lemma 6.2. *Let $\Delta \le \frac{1}{128}\log\log\log N$, let $h = \frac{1}{16}\log\log\log N$, let $k = \lceil e^{2h+1}\rceil$, let $\delta = \frac{1}{2k}$, let $n = \Theta(\log^5 N)$. Consider an R_Δ-flat cup configuration in the variable-processor cup game on n cups with initial average fill μ_0.*

Against a Δ-greedy-like emptier, $\text{rep}_\delta(\text{randalg}(k))$ using $M = 2^{\Theta(\log^4 N)}$ either achieves mass N^2 in the cups, or with probability at least $1 - 2^{-\Omega(\log^4 N)}$ makes an unknown cup in A have fill at least $h + \mu_0$ while also guaranteeing that $\mu(B) \ge -h/2 + \mu_0$, where A, B are the sets defined in Definition 6.1. The running time of $\text{rep}_\delta(\text{randalg}(k))$ is $2^{O(\log^4 N)}$.

Proof. We use the definitions given in Definition 6.1.

Note that if the emptier neglects the anchor set N^2 times, or skips N^2 emptyings, then the mass of the cups will be at least N^2, so the filler is done. For the rest of the proof we consider the case where the emptier chooses to neglect the anchor set fewer than N^2 times, and chooses to skip fewer than N^2 emptyings.

As in Proposition 6.1, we define $d = \sum_{i=2}^{k} 1/i$; we remark that, because harmonic numbers grow like $x \mapsto \ln x$, it is clear that $d = \Theta(h)$. We say that an application of $\text{randalg}(k)$ to B is ***lucky*** if it achieves backlog at least $\mu_S(B) - R_\Delta + d$ where S denotes the state of the cups at the start of the application of $\text{randalg}(k)$; note that by Proposition 6.1 if we condition on an application of $\text{randalg}(k)$ where B started R_Δ-flat being non-emptier-wasted then the application has at least a $1/k!$ chance of being lucky.

Now we prove several important bounds satisfied by A and B.

Claim 6.5. *All applications of* flatalg *make B be R_Δ-flat and B is always $(R_\Delta + d)$-flat.*

Proof. Given that the application of flatalg immediately prior to an application of randalg(k) made B be R_Δ-flat, by Proposition 6.1 we have that B will stay $(R_\Delta + d)$-flat during the application of randalg(k). Given that the application of randalg(k) immediately prior to an application of flatalg resulted in B being $(R_\Delta + d)$-flat, we have that B remains $(R_\Delta + d)$-flat throughout the duration of the application of flatalg by Lemma 6.1. Given that B is $(R_\Delta + d)$-flat before a donation occurs B is clearly still $(R_\Delta + d)$-flat after the donation, because the only change to B during a donation is that a cup is removed from B which cannot increase the fill-range of B. Note that B started R_Δ-flat at the beginning of the first donation-process. Note that if an application of flatalg begins with B being $(R_\Delta + d)$-flat, then by considering the flattening to happen in the $(|B|/2)$-processor N^2-extra-emptyings N^2-skip-emptyings cup game we ensure that it makes B be R_Δ-flat. Hence we have by induction that B has always been $(R_\Delta + d)$-flat and that all flattening processes have made B be R_Δ-flat. □

Now we aim to show that $\mu(B)$ is never very low, which we need in order to establish that every non-emptier-wasted lucky application of randalg(k) gets a cup with high fill. Interestingly, in order to lower bound $\mu(B)$ we find it convenient to first upper bound $\mu(B)$, which by greediness and flatness of B gives an upper bound on $\mu(A)$ which we then use to get a lower bound on $\mu(B)$.

Claim 6.6. *We have always had*

$$\mu(B) \leq \mu(AB) + 2.$$

Proof. There are two ways that $\mu(B) - \mu(AB)$ can increase:
Case 1: The emptier could empty from 0 cups in B while emptying from every cup in A.
Case 2: The filler could evict a cup with fill lower than $\mu(B)$ from B at the end of a donation-process.

Note that cases are exhaustive, in particular note that if the emptier skips more than 1 emptying then $\mu(B) - \mu(AB)$ must decrease because $|B| > |AB|/2$, as opposed to in Case 1 where $\mu(B) - \mu(AB)$ increases.

In Case 1, because the emptier is Δ-greedy-like,

$$\min_{a \in A} \text{fill}(a) > \max_{b \in B} \text{fill}(b) - \Delta.$$

Thus $\mu(B) \leq \mu(A) + \Delta$. We can use this to get an upper bound on $\mu(B) - \mu(AB)$. We have,

$$\begin{aligned} \mu(B) &= \frac{\mu(AB)|AB| - \mu(A)|A|}{|B|} \\ &\leq \frac{\mu(AB)|AB| - (\mu(B) - \Delta)|A|}{|B|}. \end{aligned}$$

Rearranging terms:

$$\mu(B)\left(1 + \frac{|A|}{|B|}\right) \leq \frac{\mu(AB)|AB| + \Delta|A|}{|B|}.$$

Now, because $|A| \cdot \Delta \leq n_A \cdot \Delta < n$ (by choosing δ very small), we have

$$\mu(B)\frac{|AB|}{|B|} \leq \frac{\mu(AB)|AB| + n}{|B|}.$$

Isolating $\mu(B)$ we have

$$\mu(B) \leq \mu(AB) + 1.$$

Consider the final round on which B is skipped while A is not skipped (or consider the first round if there is no such round).

From this round onward the only increase to $\mu(B) - \mu(AB)$ is due to B evicting cups with fill well below $\mu(B)$. We can upper bound the increase of $\mu(B) - \mu(AB)$ by the increase of $\mu(B)$ as $\mu(AB)$ is strictly increasing.

The cup that B evicts at the end of a donation-process has fill at least $\mu(B) - R_\Delta - (k-1)$, as the running time of randalg(k) is $k-1$, and because B starts R_Δ-flat by Claim 6.5. Evicting a cup with fill $\mu(B) - R_\Delta - (k-1)$ from B changes $\mu(B)$ by $(R_\Delta + k - 1)/(|B| - 1)$ where $|B|$ is the size of B before the cup is evicted from B. Even if this happens on each of the n_A donation-processes $\mu(B)$ cannot rise higher than $n_A(R_\Delta + k - 1)/(n - n_A)$ which by design in choosing $\delta = 1/(2k)$ is at most 1.

Thus $\mu(B) \leq \mu(AB) + 2$ is always true.

□

Now, the upper bound on $\mu(B) - \mu(AB)$ along with the guarantee that B is flat allows us to bound the highest that a cup in A could rise by greediness, which in turn upper bounds $\mu(A)$ which in turn lower bounds $\mu(B)$.

Claim 6.7. *We always have*

$$\mu(B) \geq -h/2 + \mu_0.$$

Proof. By Claim 6.6 and Claim 6.5 we have that no cup in B ever has fill greater than $u_B = \mu(AB) + 2 + R_\Delta + d$. Let $u_A = u_B + \Delta + 1$. We claim that the backlog in A never exceeds u_A. Note that $\mu(AB), u_A, u_B$ are implicitly functions of the round; $\mu(AB)$ can increase from μ_0 if the emptier skips emptyings.

Consider how high the fill of a cup $c \in A$ could be. If c came from B then when it is donated to A its fill is at most u_B; otherwise, c started with fill at most R_Δ. Both of these expressions are less than $u_A - 1$. Now consider how much the fill of c could increase while being in A. Because the emptier is Δ-greedy-like, if a cup $c \in A$ has fill more than Δ higher than the backlog in B then c must be emptied from, so any cup with fill at least $u_B + \Delta = u_A - 1$ must be emptied from, and hence u_A upper bounds the backlog in A.

Of course an upper bound on backlog in A also serves as an upper bound on the average fill of A as well, i.e. $\mu(A) \leq u_A$. Now we have

$$\begin{aligned}
\mu(B) &= -\frac{|A|}{|B|}\mu(A) + \frac{|AB|}{|B|}\mu(AB) \\
&\geq -(\mu(AB) + 3 + R_\Delta + d + \Delta)\frac{|A|}{|B|} + \frac{|AB|}{|B|}\mu(AB) \\
&= -(3 + R_\Delta + d + \Delta)\frac{|A|}{|B|} + \mu(AB) \\
&\geq -h/2 + \mu(AB)
\end{aligned}$$

where the final inequality follows because $\mu(AB) \geq 0$, and because $|A|/|B|$ is sufficiently small by our choice of $\delta = 1/(2k)$. Of course $\mu(AB) \geq \mu_0$ so we have

$$\mu(B) \geq -h/2 + \mu_0.$$

□

Now we show that at least a constant fraction of the donation-processes succeed with exponentially good probability.

Claim 6.8. *By choosing $M = 2^{\Theta(\log^4 N)}$ the filler can guarantee that with probability at least $1 - 2^{-\Omega(\log^4 N)}$, the filler achieves fill $h + \mu_0$ in some cup in A.*

Proof. Now we bound the probability that at least one cup in A has fill at least $\mu_0 + h$. If the emptier does not ***interfere*** with an application of randalg(k), i.e. the emptier does not ever neglect the anchor set and put more resources into B than the filler does during the application, then the application of randalg(k) achieves a cup with fill at least $\mu(B) - R_\Delta + d$. By our lower bound on $\mu(B)$ from Claim 6.7, namely $\mu(B) \geq -h/2 + \mu_0$, and because $d \geq 2h$, and $R_\Delta \leq h/2$ due to $\Delta \leq h/8$, we have that a successful donation process donates a cup with fill at least $\mu_0 + h$ to A. If on each of the n_A donation processes the emptier does

not interfere with the final application of randalg(k), then the probability that at least one donation process successfully donates a cup with fill at least $\mu_0 + h$ to A is at least

$$1 - (1 - 1/k!)^{n_A} \geq 1 - e^{-n\delta/k!}.$$

Now we aim to show that $e^{-n\delta/k!} \leq 2^{-\Omega(\log^4 N)}$. This is equivalent to showing $n\delta/k! \geq \Omega(\log^4 N)$. Using k^k as an upper bound for $k!$, and recalling that $n = \Theta(log^5 N)$, we have that it suffices to show that

$$k^k \leq n\delta/\Omega(\log^4 N) \leq \delta O(\log N).$$

Recalling that $\delta = 1/(2k)$ this is equivalent to showing that

$$k^{k+1} \leq O(\log N).$$

Now recalling that

$$k = \Theta(e^{2h}) = \Theta(e^{\frac{1}{8}\log\log\log N}) = \Theta((\log\log N)^{1/4}),$$

we get

$$k^{k+1} \leq 2^{O((\log\log\log N)(\log\log N)^{1/4})} \leq O(2^{\log\log N}) = O(\log N).$$

Hence we have our desired bound on the probability of getting the desired backlog in a cup, conditional on the emptier not interfering with the final application randalg(k) on each donation process.

However, the emptier is allowed to interfere, and if the emptier interferes with an application of randalg(k) then the application might not get large fill. In fact, the emptier can even interfere conditional on the filler's progress during randalg(k); in particular it could choose to only interfere with applications of randalg(k) that look like they might succeed! However, by applying randalg(k) a random number of times in each donation process, chosen from $[M]$, where $M = 2N^2 k! 2^{\log^4 N}$, by a Chernoff Bound there are at least $N^2 2^{\log^4 N}$ successful applications of randalg(k) in the donation process with incredibly good probability: probability at least $1 - 2^{-2^{\lg^4 N}}$. But since the emptier cannot neglect the anchor set more than N^2 times, the emptier has at best a $2^{-\log^4 N}$ chance of interfering with the final application of randalg(k). Taking a union bound over all donation processes, we have that with probability at least $1 - 2^{-\Omega(\log^4 N)}$ the emptier does not interfere with the final application of randalg(k) on any donation processes. We previously found that the probability of achieving the desired backlog in A conditional on the final application of randalg(k) in each donation process not being interfered with; multiplying these probabilities gives that with probability at least $1 - 2^{-\Omega(\log^4 N)}$ we achieve a cup in A with fill at least $\mu_0 + h$.

□

We now analyze the running time of the filling strategy. There are n_A donation-processes. Each donation-process consists of $O(M)$ applications of randalg(k), which each take time $O(1)$, and $O(M)$ applications of flatalg, which each take $\Theta(N^2)$ time. Thus overall the algorithm takes time

$$n_A \cdot O(M)(O(1) + O(N^2)) = 2^{O(\log^4 N)},$$

as desired.

□

Now, using Lemma 6.2 we show in Proposition 6.2 that an oblivious filler can achieve fill $\omega(1)$ in a known cup.

Proposition 6.2. *Let $H = \frac{1}{128}\log\log\log N$, let $\Delta \leq \frac{1}{128}\log\log\log N$, let $n = \Theta(\log^5 N)$. Consider an R_Δ-flat cup configuration in the variable-processor cup game on n cups with average fill μ_0. There is an oblivious filling strategy that either achieves mass N^2 among the cups, or achieves fill at least $\mu_0 + H$ in a chosen cup in running time $2^{O(\log^4 N)}$ against a Δ-greedy-like emptier with probability at least $1 - 2^{-\Omega(\log^4 N)}$.*

Proof. The filler starts by using $\text{rep}_\delta(\text{randalg}(k))$ with parameter settings as in Lemma 6.2; note that h from Lemma 6.2 is $8H$.

If this results in mass N^2 among the cups then the filler is already done. Otherwise, with probability at least $1 - 2^{-\Omega(\log^4 N)}$, there is some cup in A with fill $\mu_0 + 8H$. We assume for the rest of the proof that there is some cup $c_* \in A$ with $\text{fill}(c_*) \geq \mu_0 + 8H$.

The filler sets $p = 1$, i.e. uses a single processor. Now the filler exploits the emptier's greedy-like nature to to get fill H in a chosen cup $c_0 \in B$. For $5H$ rounds the filler places 1 unit of fill into c_0. Because the emptier is Δ-greedy-like it must empty from c_* while $\text{fill}(c_*) > \text{fill}(c_0) + \Delta$. Within $5H$ rounds the cups $\text{fill}(c_*)$ cannot decrease below $3H + \mu_0 > H + \Delta + \mu_0$. Hence, during these $5H$ rounds, only cups with fills larger than $H + \mu_0$ can be emptied from by greediness. The fill of c_0 started as at least $-4H + \mu_0$ as $\mu(B) \geq -h/2 + \mu_0$ from Lemma 6.2. After $5H$ rounds c_0 has fill at least $H + \mu_0$, because the emptier cannot have emptied c_0 until it attained fill $H + \mu_0$, and if c_0 is never emptied from then it achieves fill $H + \mu_0$. Thus the filler achieves fill $H + \mu_0$ in c_0, a *known* cup, as desired.

The running time is of course still $2^{O(\log^4 N)}$ by Lemma 6.2. □

Next we prove the ***Oblivious Amplification Lemma***. We remark that, like Lemma 6.2, Lemma 6.3 refers to filling strategies in the *regular* cup game (no extra empties). In particular, as in Lemma 6.2, we give no guarantees on the behavior of the algorithms for the E-extra-emptyings cup game with $E > 0$. It is surprising that we do not need to provide any guarantees; for flatalg and randalg we had to prove that the cups have bounded fill-range regardless of extra-emptying. We do not need to establish guarantees on the algorithms when there is extra emptying because there cannot be very much extra emptying, and by randomly choosing which applications of our algorithms are "important" we can guarantee that all important applications of our algorithms succeed (with very high probability).

Lemma 6.3 (Oblivious Amplification Lemma)**.** *Let $\delta \in (0, 1/2)$ be a constant parameter. Let $\Delta \leq O(1)$. Consider a cup configuration in the variable-processor cup game on $n \leq N, n > \Omega(1/\delta^2)$ cups with average fill μ_0 that is R_Δ-flat. Let $\text{alg}(f)$ be an oblivious filling strategy that either achieves mass N^2 or, with failure probability at most p satisfying $p \geq 2^{-O(\log^4 N)}$, achieves backlog $\mu_0 + f(n)$ on such cups in running time $T(n)$ against a Δ-greedy-like emptier. Let $M = 2^{\Theta(\log^9(N))}$.*

Consider a cup configuration in the variable-processor cup game on $n \leq N, n > \Omega(1/\delta^2)$ cups with average fill μ_0 that is R_Δ-flat. There exists an oblivious filling strategy $\text{alg}(f')$ that either achieves mass N^2 or with failure probability at most

$$p' \leq np + 2^{-\log^8 N}$$

achieves backlog $f'(n)$ satisfying

$$f'(n) \geq (1-\delta)^2 f(\lfloor (1-\delta)n \rfloor) + f(\lceil \delta n \rceil) + \mu_0$$

and $f'(n) \geq f(n)$, in running time

$$T'(n) \leq Mn \cdot T(\lfloor (1-\delta)n \rfloor) + T(\lceil \delta n \rceil)$$

against a Δ-greedy-like emptier.

Proof. We use the definitions and notation given in Definition 6.1.

Note that the emptier cannot neglect the anchor set more than N^2 times per donation-process, and the emptier cannot skip more than N^2 emptyings, without causing the mass of the cups to be at least N^2; we assume for the rest of the proof that the emptier chooses not to do this.

The filler simply uses $\text{alg}(f)$ on all the cups if

$$f(n) \geq (1-\delta)^2 f(n_B) + f(n_A).$$

In this case our strategy trivially has the desired guarantees. In the rest of the proof we consider the case where we cannot simply fall back on $\text{alg}(f)$ to achieve the desired backlog.

The filler's strategy can be summarized as follows:
Step 1: Make $\mu(A) \geq (1-\delta)^2 f(n_B)$ by using $\mathrm{rep}_\delta(\mathrm{alg}(f))$ on all the cups, i.e. applying $\mathrm{alg}(f)$ repeatedly to B, flattening B before each application, and then donating a cup from B to A on randomly chosen applications of $\mathrm{alg}(f)$.
Step 2: Flatten A using flatalg, and then use $\mathrm{alg}(f)$ on A.

Now we analyze Step 1, and show that by appropriately choosing parameters it can be made to succeed. For this proof we need all donation-processes to succeed. This necessitates choosing M very large. In particular we choose $M = 2^{\Theta(\log^9 N)}$—recall that $[M]$ is the set from which we randomly choose how many times to apply $\mathrm{alg}(f)$ in a donation-process. By choosing M this large we cannot hope to guarantee that every application of $\mathrm{alg}(f)$ succeeds: there are far too many applications. On the other hand, having M so large allows us to have a very tight concentration bound on how many applications of $\mathrm{alg}(f)$ succeed. Ignoring for a moment the fact that the emptier can choose to neglect the anchor set, i.e. assuming that no applications of $\mathrm{alg}(f)$ are emptier-wasted, the probability that fewer than $M(1-p)(1-\xi)$ of M applications of $\mathrm{alg}(f)$ succeed is at most $e^{-\Theta(M(1-p)\xi^2}$ by a Chernoff bound. However, the emptier is allowed to ***interfere***, i.e. neglect the anchor set and do extra emptying in the non-anchor set. Fortunately the emptier can interfere with at most N^2 of the applications of $\mathrm{alg}(f)$, which is very small compared to $M \cdot (1-2p)$. Thus, if we condition on there being at least $M(1-2p)$ applications that would succeed if the emptier did not interfere, there are at least $M(1-2p) - N^2$ applications of $\mathrm{alg}(f)$ that succeed. Let $\mathcal{W}$ be the event that the donation-process succeeds, i.e. the final application of $\mathrm{alg}(f)$ is not emptier-wasted and succeeds, and let $\mathcal{D}$ be the event that at least $M(1-2p)$ of the M applications of $\mathrm{alg}(f)$ would succeed without interference by the emptier. Let $1-q = \Pr[\mathcal{W}]$. Obviously

$$\Pr[\mathcal{W}] \geq \Pr[\mathcal{W} \wedge \mathcal{D}] = \Pr[\mathcal{D}] \cdot \Pr[\mathcal{W}|\mathcal{D}].$$

Because the filler chooses which application of $\mathrm{alg}(f)$ is the final application uniformly at random from $[M]$ we thus have

$$1 - q \geq (1 - e^{-2Mp^2}) \left(\frac{M \cdot (1-2p) - N^2}{M} \right).$$

Rearranging, and over-estimating (i.e. dropping unnecessary terms to simplify the expression, while maintaining the truth of the expression), we have

$$q \leq e^{-2Mp^2} + 2p + N^2/M.$$

By assumption $p \geq 2^{-O(\log^4 N)}$, so as $M \geq 2^{\Omega(\log^9 N)}$, we have $Mp^2 \geq 2^{\Omega(\log^9 N)}$. Thus,

$$q \leq 2p + 2^{-2^{\Omega(\log^9 N)}} + N^2 2^{-\Omega(\log^9 N)}.$$

For convenience we loosen this to

$$q \leq 2p + \frac{2^{-\log^8 N}}{N+1}.$$

Taking a union bound, we have that with failure probability at most $q \cdot n_A$ all donation-process successfully achieve a cup with fill at least $\mu_{S_0}(B) + f(n_B)$ where $\mu_{S_0}(B)$ refers to the average fill of B measured at the start of the application of $\mathrm{alg}(f)$; now we assume all donation-processes are successful, and demonstrate that this translates into the desired average fill in A.

Let $\mathbf{skips}_t$ denote the number of times that the emptier has skipped the anchor set by round t. Consider how $\mu(B) - \mathrm{skips}/n_B$ changes over the course of the donation processes. As noted above, at the end of each donation-process $\mu(B)$ decreases due to B donating a cup with fill at least $\mu(B) + f(n_B)$. In particular, if S denotes the cup state immediately before a cup is donated on the i-th donation-process, B_0 denotes the set B before the donation and B_1 denotes the set B after the donation, then $\mu_S(B_1) = \mu_S(B_0) - f(n_B)/(n-i)$. Now we claim that $t \mapsto \mu_{S_t}(B) - \mathrm{skips}_t/n_B$ is monotonically decreasing. Clearly donation decreases $\mu(B) - \mathrm{skips}/n_B$. If the anchor set is neglected then $\mu(B)$ decreases, causing $\mu(B) - \mathrm{skips}/n_B$ to decrease. If a skip

occurs, then skips $/n_B$ increases by more than $\mu(B)$ increases, causing $\mu(B) - \text{skips}/n_B$ to decrease. Let t_* be the cup state at the end of all the donation-processes. We have that

$$\mu_{S_{t_*}}(B) - \frac{\text{skips}_{t_*}}{n_B} \leq \mu_0 - \sum_{i=1}^{n_A} \frac{f(n_B)}{n-i}. \tag{16}$$

By conservation of mass we have

$$n_A \cdot \mu_{S_{t_*}}(A) + n_B \cdot \mu_{S_{t_*}}(B) = n\mu_0 + \text{skips}_{t_*}.$$

Rearranging,

$$\mu_{S_{t_*}}(A) = \mu_0 + \frac{n_B}{n_A}\left(\mu_0 + \frac{\text{skips}_{t_*}}{n_B} - \mu_{S_{t_*}}(B)\right). \tag{17}$$

Now we obtain a simpler form of Inequality (16). Let H_n denote the n-th harmonic number. We desire a simpler lower bound for

$$\sum_{i=1}^{n_A} \frac{1}{n-i} = H_{n-1} - H_{n_B-1}.$$

We use the well known fact that

$$\frac{1}{2(n+1)} < H_n - \ln n - \gamma < \frac{1}{2n} \tag{18}$$

where $\gamma = \Theta(1)$ denotes the Euler-Mascheroni constant. Of course $H_{n-1} - H_{n_B-1} \geq H_n - H_{n_B}$. Now using Inequality (18) we have

$$\begin{aligned} H_n - H_{n_B} &> \left(\ln n + \gamma + \frac{1}{2(n+1)}\right) - \left(\ln n_B + \gamma + \frac{1}{2n_B}\right) \\ &> \ln\frac{1}{1-\delta} + \frac{1}{2}\left(\frac{n_B - n - 1}{(n+1)n_B}\right) \\ &> \delta - \Theta\left(\frac{\delta}{(1-\delta)n}\right). \end{aligned}$$

Now using this lower bound on $H_n - H_{n_B}$ in Inequality (17) we have:

$$\begin{aligned} \mu_{t_*}(A) &> \mu_0 + \frac{n_B}{n_A}\left(\delta - \Theta\left(\frac{\delta}{(1-\delta)n}\right)\right) f(n_B) \\ &= \mu_0 + \frac{\lfloor(1-\delta)n\rfloor}{\lceil\delta n\rceil}\left(\delta - \Theta\left(\frac{\delta}{(1-\delta)n}\right)\right) f(n_B) \\ &> \mu_0 + \left(\frac{1-\delta}{\delta} - \frac{1}{\delta^2 n}\right)\left(\delta - \Theta\left(\frac{\delta}{(1-\delta)n}\right)\right) f(n_B) \\ &> \mu_0 + ((1-\delta) - \Theta(1/(\delta n)))\, f(n_B). \end{aligned}$$

Thus, by choosing $n > \Omega(1/\delta^2)$ we have

$$\mu_{t_*}(A) > \mu_0 + (1-\delta)^2 f(n_B).$$

We have shown that in Step 1 the filler achieves average fill $\mu_0 + (1-\delta)f(n_B)$ in A with failure probability at most $q \cdot n_A$. Now the filler flattens A and uses $\text{alg}(f)$ on A. It is clear that this is possible, and succeeds with probability at least p. This gets a cup with fill

$$\mu_0 + (1-\delta)^2 f(n_B) + f(n_A)$$

in A, as desired.

Taking a union bound over the probabilities of Step 1 and Step 2 succeeding gives that the entire procedure fails with probability at most

$$p' \leq p + q \cdot n_A \leq np + 2^{-\log^8 N}.$$

The running time of Step 1 is clearly $M \cdot n \cdot T(\lfloor (1-\delta)n \rfloor)$ and the running time of Step 2 is clearly $T(\lceil \delta n \rceil)$; summing these yields the desired upper bound on running time.

□

Finally we prove that an oblivious filler can achieve backlog $N^{1-\varepsilon}$, just like an adaptive filler despite the oblivious filler's disadvantage. The proof is very similar to the proof of Theorem 4.1, although significant care must be taken in the randomized case to get the desired probabilistic guarantees.

Theorem 6.1. *There is an oblivious filling strategy for the variable-processor cup game on N cups that achieves backlog at least $\Omega(N^{1-\varepsilon})$ for any constant $\varepsilon > 0$ in running time $2^{polylog(N)}$ with probability at least $1 - 2^{-polylog(N)}$ against a Δ-greedy-like emptier with $\Delta \leq \frac{1}{128} \log\log\log N$.*

Proof. We aim to achieve backlog $(N/n_b)^{1-\varepsilon} - 1$ for $n_b = \Theta(\log^5 N)$ on N cups. Let δ be a constant, chosen as a function of ε.

We call the algorithm from Proposition 6.2 $\text{alg}(f_0)$. We remark that in the proof of Theorem 4.1 the size of the base case is constant, whereas now the size of the base case is $\text{polylog}(N)$; the larger base case is necessary, as a constant-size base case would destroy our probability of success. The probability of success achieved by $\text{alg}(f_0)$ is, by construction, sufficient to take a union bound over $2^{\text{polylog}\, N}$ applications of $\text{alg}(f_0)$. Now we construct $\text{alg}(f_{i+1})$ as the amplification of $\text{alg}(f_i)$ using Lemma 6.3 and parameter δ. Define a sequence g_i as

$$g_i = \begin{cases} n_b \lceil 16/\delta \rceil, & i = 0 \\ \lfloor g_{i-1}/(1-\delta) \rfloor, & i \geq 1 \end{cases}.$$

We claim the following regarding our construction:

Claim 6.9.

$$f_i(k) \geq (k/n_b)^{1-\varepsilon} - 1 \text{ for all } k \leq g_i. \tag{19}$$

Proof. We prove Claim 6.9 by induction on i.
For $i = 0$, the base case of our induction, (19) is true as $(k/n_b)^{1-\varepsilon} - 1 \leq O(1)$ for $k \leq g_0$, whereas $\text{alg}(f_0)$ achieves backlog $\Omega(\log\log\log N) > \omega(1)$.
Now we assume (19) for f_i, and aim to establish (19) for f_{i+1}. Note that, by design of g_i, if $k \leq g_{i+1}$ then $\lfloor k \cdot (1-\delta) \rfloor \leq g_i$. Consider any $k \in [g_{i+1}]$. First we deal with the trivial case where $k \leq g_0$. Here,

$$f_{i+1}(k) \geq f_i(k) \geq \cdots \geq f_0(k) \geq (k/n_b)^{1-\varepsilon} - 1.$$

Now we consider $k \geq g_0$. Since f_{i+1} is the amplification of f_i we have by Lemma 6.3 that

$$f_{i+1}(k) \geq (1-\delta)^2 f_i(\lfloor (1-\delta)k \rfloor) + f_i(\lceil \delta k \rceil).$$

By our inductive hypothesis, which applies as $\lceil \delta k \rceil \leq g_i, \lfloor k \cdot (1-\delta) \rfloor \leq g_i$, we have

$$f_{i+1}(k) \geq (1-\delta)^2 (\lfloor (1-\delta)k/n_b \rfloor^{1-\varepsilon} - 1) + \lceil \delta k/n_b \rceil^{1-\varepsilon} - 1.$$

Dropping the floor and ceiling, incurring a -1 for dropping the floor, we have

$$f_{i+1}(k) \geq (1-\delta)^2 (((1-\delta)k/n_b - 1)^{1-\varepsilon} - 1) + (\delta k/n_b)^{1-\varepsilon} - 1.$$

Because $(x-1)^{1-\varepsilon} \geq x^{1-\varepsilon} - 1$, as $x \mapsto x^{1-\varepsilon}$ is a sub-linear sub-additive function, we have

$$f_{i+1}(k) \geq (1-\delta)^2 (((1-\delta)k/n_b)^{1-\varepsilon} - 2) + (\delta k/n_b)^{1-\varepsilon} - 1.$$

Moving the $(k/n_b)^{1-\varepsilon}$ to the front we have

$$f_{i+1}(k) \geq (k/n_b)^{1-\varepsilon} \cdot \left((1-\delta)^{3-\varepsilon} + \delta^{1-\varepsilon} - \frac{2(1-\delta)^2}{(k/n_b)^{1-\varepsilon}} \right) - 1.$$

Because $(1-\delta)^{3-\varepsilon} \geq 1 - (3-\varepsilon)\delta$, a fact called Bernoulli's Identity, we have

$$f_{i+1}(k) \geq (k/n_b)^{1-\varepsilon} \cdot \left(1 - (3-\varepsilon)\delta + \delta^{1-\varepsilon} - \frac{2(1-\delta)^2}{(k/n_b)^{1-\varepsilon}} \right) - 1.$$

Of course $-2(1-\delta)^2 > -2$, so

$$f_{i+1}(k) \geq (k/n_b)^{1-\varepsilon} \cdot \left(1 - (3-\varepsilon)\delta + \delta^{1-\varepsilon} - 2/(k/n_b)^{1-\varepsilon} \right) - 1.$$

Because

$$\frac{-2}{(k/n_b)^{1-\varepsilon}} \geq \frac{-2}{(g_0/n_b)^{1-\varepsilon}} \geq -2(\delta/16)^{1-\varepsilon} \geq -\delta^{1-\varepsilon}/2,$$

which follows from our choice of $g_0 = \lceil 8/\delta \rceil n_b$ and the restriction $\varepsilon < 1/2$, we have

$$f_{i+1}(k) \geq (k/n_b)^{1-\varepsilon} \cdot \left(1 - (3-\varepsilon)\delta + \delta^{1-\varepsilon} - \delta^{1-\varepsilon}/2 \right) - 1.$$

Finally, combining terms we have

$$f_{i+1}(k) \geq (k/n_b)^{1-\varepsilon} \cdot \left(1 - (3-\varepsilon)\delta + \delta^{1-\varepsilon}/2 \right) - 1.$$

Because $\delta^{1-\varepsilon}$ dominates δ for sufficiently small δ, there is a choice of $\delta = \Theta(1)$ such that

$$1 - (3-\varepsilon)\delta + \delta^{1-\varepsilon}/2 \geq 1.$$

Taking δ to be this small we have,

$$f_{i+1}(k) \geq (k/n_b)^{1-\varepsilon} - 1,$$

completing the proof. □

The sequence g_i larger the sequence g_i from the proof of Theorem 4.1 for the same value of δ: it is defined by the same recurrence, except that the first term is n_b times larger. Thus we have that $g_{i_*} \geq N$ for some $i_* \leq O(\log N)$. Hence $\mathrm{alg}(f_{i_*})$ achieves backlog

$$f_{i_*}(N) \geq (N/n_b)^{1-\varepsilon} - 1.$$

Let $\varepsilon' = 2\varepsilon$. Of course $n_b \leq O(N^{\varepsilon})$, so

$$(N/n_b)^{1-\varepsilon} - 1 \geq \Omega(N^{1-\varepsilon'}).$$

Let the running time of $f_i(N)$ be $T_i(N)$. From the Amplification Lemma we have following recurrence bounding $T_i(N)$:

$$\begin{aligned} T_i(n) &\leq 2^{\mathrm{polylog}(N)} \cdot T_{i-1}(\lfloor (1-\delta)n \rfloor) + T_{i-1}(\lceil \delta n \rceil) \\ &\leq 2^{\mathrm{polylog}(N)} T_{i-1}(\lfloor (1-\delta)n \rfloor). \end{aligned}$$

It follows that $\mathrm{alg}(f_{i_*})$, recalling that $i_* \leq O(\log N)$, has running time

$$T_{i_*}(n) \leq (2^{\mathrm{polylog}(N)})^{O(\log N)} \leq 2^{\mathrm{polylog}(N)}$$

as desired.

Now we analyze the probability that the construction fails. Consider the recurrence $a_{i+1} = \alpha a_i + \beta, a_0 = \gamma$; the recurrence bounding failure probability is a special case of this. Expanding, we see that the recurrence solves to $a_k = \Theta(\alpha^{k-1})\beta + \alpha^k \gamma$. In our case we have

$$\alpha \leq N, \beta = 2^{-\log^8 N}, \gamma \leq 2^{-\Omega(\log^4 N)}.$$

Hence the recurrence solves to

$$p_{i_*} \leq 2^{-\operatorname{polylog}(N)},$$

as desired.

□

References

[1] Micah Adler, Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul Goldberg, and Mike Paterson. A proportionate fair scheduling rule with good worst-case performance. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 101–108, 2003.

[2] Amihood Amir, Martin Farach, Ramana M. Idury, Johannes A. La Poutré, and Alejandro A. Schäffer. Improved dynamic dictionary matching. *Inf. Comput.*, 119(2):258–282, 1995.

[3] Amihood Amir, Gianni Franceschini, Roberto Grossi, Tsvi Kopelowitz, Moshe Lewenstein, and Noa Lewenstein. Managing unbounded-length keys in comparison-driven data structures with applications to online indexing. *SIAM Journal on Computing*, 43(4):1396–1416, 2014.

[4] Yossi Azar and Arik Litichevskey. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.

[5] Amotz Bar-Noy, Ari Freund, Shimon Landa, and Joseph (Seffi) Naor. Competitive on-line switching policies. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 525–534, 2002.

[6] Amotz Bar-Noy, Aviv Nisgav, and Boaz Patt-Shamir. Nearly optimal perfectly periodic schedules. *Distributed Computing*, 15(4):207–220, 2002.

[7] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, Jun 1996.

[8] Sanjoy K Baruah, Johannes E Gehrke, and C Greg Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of the 9th International Parallel Processing Symposium*, pages 280–288, 1995.

[9] Michael Bender, Rathish Das, Martín Farach-Colton, Rob Johnson, and William Kuszmaul. Flushing without cascades. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

[10] Michael Bender, Martín Farach-Colton, and William Kuszmaul. Achieving optimal backlog in multiprocessor cup games. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019.

[11] Michael A Bender, Rezaul A Chowdhury, Rathish Das, Rob Johnson, William Kuszmaul, Andrea Lincoln, Quanquan C Liu, Jayson Lynch, and Helen Xu. Closing the gap between cache-oblivious and cache-adaptive analysis. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 63–73, 2020.

[12] Michael A Bender, Rezaul A Chowdhury, Rathish Das, Rob Johnson, William Kuszmaul, Andrea Lincoln, Quanquan C Liu, Jayson Lynch, and Helen Xu. Closing the gap between cache-oblivious and cache-adaptive analysis. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 63–73, 2020.

[13] Michael A Bender, Roozbeh Ebrahimi, Jeremy T Fineman, Golnaz Ghasemiesfeh, Rob Johnson, and Samuel McCauley. Cache-adaptive algorithms. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 958–971. SIAM, 2014.

[14] Peter Damaschke and Zhen Zhou. On queuing lengths in on-line switching. *Theoretical computer science*, 339(2-3):333–343, 2005.

[15] Paul Dietz and Daniel Sleator. Two algorithms for maintaining order in a list. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 365–372, 1987.

[16] Paul F. Dietz and Rajeev Raman. Persistence, amortization and randomization. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 78–88, 1991.

[17] Johannes Fischer and Paweł Gawrychowski. Alphabet-dependent string searching with wexponential search trees. In *Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 160–171, 2015.

[18] Rudolf Fleischer and Hisashi Koga. Balanced scheduling toward loss-free packet queuing and delay fairness. *Algorithmica*, 38(2):363–376, Feb 2004.

[19] H Richard Gail, G Grover, Roch Guérin, Sidney L Hantler, Zvi Rosberg, and Moshe Sidi. Buffer size requirements under longest queue first. *Performance Evaluation*, 18(2):133–140, 1993.

[20] Leszek Gasieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 229–240. Springer, 2017.

[21] Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.

[22] Michael T Goodrich and Paweł Pszona. Streamed graph drawing and the file maintenance problem. In *International Symposium on Graph Drawing*, pages 256–267. Springer, 2013.

[23] Nan Guan and Wang Yi. Fixed-priority multiprocessor scheduling: Critical instant, response time and utilization bound. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 2470–2473. IEEE, 2012.

[24] Tsvi Kopelowitz. On-line indexing for general alphabets via predecessor queries on subsets of an ordered list. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 283–292, 2012.

[25] William Kuszmaul. Achieving optimal backlog in the vanilla multi-processor cup game. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

[26] Andrea Lincoln, Quanquan C Liu, Jayson Lynch, and Helen Xu. Cache-adaptive exploration: Experimental results and scan-hiding for adaptivity. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 213–222, 2018.

[27] Ami Litman and Shiri Moran-Schein. On distributed smooth scheduling. In *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 76–85, 2005.

[28] Ami Litman and Shiri Moran-Schein. Smooth scheduling under variable rates or the analog-digital confinement game. *Theor. Comp. Sys.*, 45(2):325–354, June 2009.

[29] Ami Litman and Shiri Moran-Schein. On centralized smooth scheduling. *Algorithmica*, 60(2):464–480, 2011.

[30] Chung Laung Liu. Scheduling algorithms for multiprocessors in a hard real-time environment. *JPL Space Programs Summary, 1969*, 1969.

[31] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[32] Richard T Mills, Andreas Stathopoulos, and Dimitrios S Nikolopoulos. Adapting to memory pressure from within scientific applications on multiprogrammed cows. In *Proc. 8th International Parallel and Distributed Processing Symposium (IPDPS)*, page 71, 2004.

[33] Mark Moir and Srikanth Ramamurthy. Pfair scheduling of fixed and migrating periodic tasks on multiple resources. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, pages 294–303, 1999.

[34] Christian Worm Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 618–627, 2003.

[35] Michael Rosenblum, Michel X Goemans, and Vahid Tarokh. Universal bounds on buffer size for packetizing fluid policies in input queued, crossbar switches. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1126–1134, 2004.