Variable-Processor Cup Games

William Kuszmaul*[1] and Alek Westover†[2]

MIT[1], MIT PRIMES[2]
kuszmaul@mit.edu, alek.westover@gmail.com

Abstract

In a ***cup game*** a filler and an emptier take turns adding and removing water from cups, subject to certain constraints. In one version of the cup game, the p-processor cup game, the filler distributes p units of water among the n cups with at most 1 unit of water to any particular cup, and the emptier chooses p cups to remove at most one unit of water from. Analysis of the cup game is important for applications in processor scheduling, buffer management in networks, quality of service guarantees, and deamortization.

We investigate a new variant of the classic multi-processor cup game, which we call the ***variable-processor cup game***, in which the resources of the emptier and filler are variable. In particular, in the variable-processor cup game the filler is allowed to change p at the beginning of each round. Although the modification to allow variable resources seems small, we show that it drastically alters the outcome of the game.

We construct a filling strategy that an adaptive filler can use to get backlog $\Omega(n)$ in running time $2^{O(n)}$. We also construct a filling strategy that an adaptive filler can use to get backlog $\Omega(n^{1-\epsilon})$ for any constant $\epsilon > 0$ in running time $2^{O(\log^2 n)}$. Not only is this bound shockingly large, but the steep trade off-curve between running-time and backlog is very surprising: the time required to genuinely get a 1 as the exponent instead of $1-\epsilon$ (e.g. 0.999) goes up from quasi-polynomial time to exponential time!

Furthermore, we demonstrate that this lower bound on backlog is tight: using a novel set of invariants we prove that a greedy emptier never lets backlog exceed $O(n)$.

We also investigate bounds on an oblivious filler. We show, using concentration bounds for random variables (Hoeffding's Inequality), that – surprisingly – an oblivious filler can use very similar techniques as the adaptive filler to achieve very large backlog. In particular, an oblivious filler can achieve backlog $2^{\Omega(\sqrt{\log n})}$ in running time $O(\text{poly}(n))$ with constant probability against any "greedy-like" emptier.

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

1 Introduction

Definition and Motivation. The ***cup game*** is a multi-round game in which the two players – the ***filler*** and the ***emptier*** – take turns adding and removing water from cups. On each round of the classic ***p-processor cup game*** on n cups, the filler first distributes p units of water among the n cups with at most 1 unit to any particular cup (without this restriction the filler can trivially achieve unbounded backlog by placing all of its fill in a single cup every round), and then the emptier removes at most 1 unit of water from each of p cups[1].

The cup game naturally arises in the study of processor-scheduling. The incoming water added by the filler represents work added to the system at time steps. At each time step after the new work comes in each of p processors must be allocated to a task, which they will achieve 1 unit of progress on before the next time step. The assignment of processors to tasks is modeled by the emptier deciding which cups to empty from. The backlog of the system is the largest amount of work left on any given task. To model this, in the cup game, the ***backlog*** of the cups is defined to be the fill of the fullest cup at a given state. It is important to know bounds on how large backlog can get.

Previous Work. The bounds on backlog are well known for the case where $p = 1$, i.e. the ***single-processor cup game***. In the single-processor cup game an adaptive filler can achieve backlog $\Omega(\log n)$ and a greedy emptier never lets backlog exceed $O(\log n)$. The bounds are much better against an oblivious filler. In the randomized version of the single-processor cup game, which can be interpreted as a smoothed analysis of the deterministic version, the emptier never lets backlog exceed $O(\log \log n)$, and a filler can achieve backlog $\Omega(\log \log n)$.

Recently Kuszmaul has achieved bounds on the case where $p > 1$, i.e. the ***multi-processor cup game*** [2]. Kuszmaul showed that in the p-processor cup game on n cups a greedy emptier never lets backlog exceed $O(\log n)$. He also proved a lower bound of $\Omega(\log(n-p))$. Recently we showed a lower bound of $\Omega(\log n - \log(n-p))$. Combined these bounds imply a lower bound of $\Omega(\log n)$. Kuszmaul also established an upper bound of $O(\log \log n + \log p)$ against oblivious fillers, and a lower

[1]note that negative fill is not allowed, so if the emptier empties from a cup with fill below 1 the cups fill will become 0.

bound of $\Omega(\log\log n)$ (note that tight bounds on backlog against an oblivious filler are not yet known for large p).

Our Variant. We investigate a variant of the vanilla multi-processor cup game which we call the ***variable-processor cup game***. In the variable-processor cup game 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; just like in the classic cup game we take the resources of the filler and emptier to be identical. Although this restriction may seem artificial, it is crucial; if the filler has more resources than the emptier, then the filler could trivially achieve unbounded backlog, as average fill will increase by at least some positive constant at each round. Taking the resources of the players to be identical makes the game balanced, and hence interesting.

A priori having variable resources offers neither player a clear advantage: lower values of p mean that the emptier is at more of a discretization disadvantage but also mean that the filler can "anchor" fewer cups[2]. We hoped that the variable-processor cup game could be simulated in the vanilla multi-processor cup game, because the extra ability given to the filler does not seem very strong.

However, we show that attempts at simulating the variable-processor cup game are futile because the variable-processor cup game is–surprisingly–vastly different from the multi-processor cup game.

Outline and Results. In Section 2 we establish the conventions and notations we will use to discuss the variable-processor cup game.

In Section 3 we provide an inductive proof of a lower bound on backlog in Corollary 1. The base case of the argument is a direct consequence of Proposition 1, and the inductive step follows from the "Adaptive Amplification Lemma" (Lemma 1). Corollary 1 gives a lower bound of $\Omega(n)$ on backlog. Corollary 1 provides two algorithms: one algorithm with running time $2^{O(n)}$ that achieves backlog $\Omega(n)$, and another with running time $2^{O(\log^2 n)}$ that achieves backlog $\Omega(n^{1-\epsilon})$ for any constant $\epsilon > 0$.

In Section 4 we prove a novel invariant: the average fill of the k fullest cups is at most $2n - k$. In particular this implies (setting $k = 1$) that backlog is $O(n)$. Thus, our analysis is tight.

Section **??** has similar macro-structure to Section 3: We lower bound backlog in Corollary **??**, using Proposition **??** as the base case of the inductive argument and the "Oblivious Amplification Lemma" (Lemma **??**) to facilitate the inductive step. Corollary **??** gives the lower bound $2^{\Omega(\sqrt{\log n})}$ on backlog, against a "greedy-like" emptier. In particular the corollary asserts that we can achieve this backlog in time $O(\text{poly}(n))$. Note that the restriction on runtime of the filler is the main way in which this bound differs from the adaptive case.

[2]A useful part of many filling algorithms is maintaining an "anchor" set of "anchored" cups. The filler always places 1 unit of water in each anchored cup. This ensures that the fill of an anchored cup never decreases after it is placed in the anchor set.

2 Preliminaries

The cup game consists of a sequence of rounds. On the t-th round the state starts as S_t. The filler chooses the number of processors p_t for the round. Then the filler distributes p_t units of water among the cups (with at most 1 unit of water to any particular cup). After this, the game is in an intermediate state, which we call state I_t. Then the emptier chooses p_t cups to empty at most 1 unit of water from. Note that if the fill of a cup that the emptier empties from is less than 1 the emptier reduces the fill of this cup to 0 by emptying from it; we say that the emptier ***zeros-out*** a cup if it empties from a cup with fill less than 1. Note that whenever the emptier zeros-out a cup the emptier has removed less fill than the filler has added; hence the average fill will increase. This concludes the round; the state of the game is now S_{t+1}.

Denote the fill of a cup c by $\text{fill}(c)$. Let the ***positive tilt*** of a cup c be $\text{tilt}(c) = \max(0, \text{fill}(c))$, and let the positive tilt of a set X of cups be $\text{tilt}(X) = \sum_{c\in X} \text{tilt}(c)$. Let the ***mass*** of a set of cups X be $m(X) = \sum_{c\in X} \text{fill}(c)$. Denote the average fill of a set of cups X by $\mu(X)$. Note that $\mu(X)|X| = m(X)$.

Let the ***rank*** of a cup at a given state be its position in a 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.

Many of our lower bound proofs will adopt the convention of allowing for negative fill. This is because our results are used recursively, so this negative fill is really just fill below the average fill. Note that it is strictly easier for the filler to achieve high backlog when cups can zero out, because then some of the emptiers resources are wasted. On the other hand, during the upper-bound proof we show that a greedy emptier maintains the desired invariants even if cups zero-out. This is crucial as the game is harder for the emptier when cups can zero out.

When measuring fill relative to average fill, we will reference the actual – i.e. non-relative – fills by calling them ***absolute*** fills.

3 Adaptive Filler Lower Bound

Proposition 1. *There exists an adaptive filling strategy for the variable-processor cup game on n cups that achieves backlog at least $\frac{1}{4}\ln(n/2)$, where fill is relative to the average fill of the cups, with negative fill allowed.*

Proof. Let $h = \frac{1}{4}\ln(n/2)$ be the desired fill. Once a cup with fill at least h is achieved the filler stops, the process completed. Let A consist of the $n/2$ fullest cups, and

B consist of the rest of the cups. Note that A, B are implicitly functions of the round t.

If the process is not yet complete, that is $\text{fill}(c) < h$ for all cups c, then $\text{tilt}(A \cup B) < h \cdot n$. Assume for sake of contradiction that there are more than $n/2$ cups c with $\text{fill}(c) \leq -2h$. The mass of those cups would be at most $-hn$ but the mass of the remaining cups cannot exceed $hn/2$. This implies that average fill is negative when it is in fact 0 by definition, a contradiction. Hence there are at most $n/2$ cups c with $\text{fill}(c) \leq -2h$.

The filler sets the number of processors equal to 1 and plays a single processor cup game on $n/2$ cups that each have fill at least $-2h$ (which must exist) for $n/2 - 1$ steps. Throughout this process the filler maintains a set of cups called the ***active set***: the set of cups that the filler will place fill in. The filler initializes the active set to be A. Note that that $\text{fill}(c) \geq -2h$ for all cups $c \in A$, as A consists of the $n/2$ fullest cups. The filler removes 1 cup from the active set at each step. At each step the filler distributes water equally among the cups in its active set. Then, the emptier will choose some cup to empty from. If this cup is in the active set, the filler removes it from the active set. Otherwise, the filler chooses an arbitrary cup to remove from the active set.

After $n/2 - 1$ steps, the active set will consist of a single cup. This cup's fill has increased by $1/(n/2) + 1/(n/2 - 1) + \cdots + 1/2 + 1/1 \geq \ln n/2 = 4h$. This cup's fill started as at least $-2h$. Thus this cup has fill at least $2h$ now, as desired. □

Lemma 1 (The Adaptive Amplification Lemma)**.** *Let f be an adaptive filling strategy that achieves backlog $f(n)$ in the variable-processor cup game on n cups (relative to average fill, with negative fill allowed). Let $n_0 \in \mathbb{N}$ be a constant such that a filler can achieve backlog 1 on n_0 cups (n_0 exists by Proposition 1). Let $\delta \in (0, 1)$ be a parameter, and let $L \in \mathbb{N}$ be a parameter such that $n_0 \leq (1-\delta)\delta^L n \leq n_0/\delta$.* [3]

Then, there exists an adaptive filling strategy that achieves backlog $f'(n)$ satisfying

$$f'(n) \geq (1-\delta) \sum_{\ell=0}^{L} f((1-\delta)\delta^\ell n)$$

in the variable-processor cup game on n cups.

Proof. The basic idea of this analysis is as follows:

- **Step 1:** Using f repeatedly, achieve average fill at least $(1-\delta)f(n(1-\delta))$ in a set of $n\delta$ cups.
- **Step 2:** Reduce the number of processors to $n\delta$, and recurse on the $n\delta$ cups with high average fill.

Let A, the ***anchor set***, be initialized to consist of the $n\delta$ fullest cups, and let B the ***non-anchor set*** be initialized to consist of the rest of the cups (so $|B| = (1-\delta)n$). Let $n_\ell = n\delta^{\ell-1}$, $h_\ell = (1-\delta)f(n_\ell(1-\delta))$; the filler will achieve a set of at least $n_\ell\delta$ cups with average fill at least h_ℓ on the ℓ-th level of recursion. On the ℓ-th level of recursion $|A| = \delta \cdot n_\ell, |B| = (1-\delta) \cdot n_\ell$.

We now elaborate on how to achieve Step 1. 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.

On the ℓ-th level of recursion the filler uses the following procedure, termed a ***swapping-process***, to achieve the desired average fill in A: repeatedly apply f to B, and then take the cup generated by f within B with fill at least $f(|B|)$ and swap it with a cup in A; repeat until A has the desired average fill. Note that

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

so

$$\mu(A) = -\mu(B) \cdot (1-\delta)/\delta.$$

Thus, if at any point in the process B has average fill lower than $-h_\ell \cdot \delta/(1-\delta)$, then A has average fill at least h_ℓ, so the process is finished. So long as B has average fill at least $-h_\ell \cdot \delta/(1-\delta)$ we will apply f to B.

It is somewhat complicated to apply f to B however, because we must guarantee that in the steps that the algorithm takes while applying f the amount of mass removed from B by the emptier is the same as the amount of mass added to B by the filler. This might not be the case if the emptier does not empty from each anchor cup at each step. We say that the emptier ***neglects*** the anchor set on an application of f if there is some step during the application of f in which the emptier does not empty from some anchor cup.

We will apply f to B at most $h_\ell n_\ell \delta + 1$ times, and at the end of an application of f we only swap the generated cup into A if the emptier has not neglected the anchor set during the application of f.

Note that each time the emptier neglects the anchor set the mass of the anchor set increases by 1. If the emptier neglects the anchor set $h_\ell n_\ell \delta + 1$ times, then the average fill in the anchor set increases by more than h_ℓ, so the desired average fill is achieved in the anchor set.

Otherwise, there must have been an application of f for which the emptier did not neglect the anchor set. We only swap a cup into the anchor set if this is the case. In this case we achieve fill
$-h_\ell \cdot \delta/(1-\delta) + f(n_\ell(1-\delta)) = (1-\delta)f(n_\ell(1-\delta)) = h_\ell$
in a non-anchor cup, and swap it with the smallest cup in the anchor set.

We achieve average fill h_ℓ in the anchor set for L levels of recursion. Summing h_ℓ for $0 \leq \ell \leq L$ yields the desired result.

□

Corollary 1. *There are adaptive filling strategies for the variable-processor cup game on n cups that achieve backlog $\Omega(n)$ in time $2^{O(n)}$ and backlog $\Omega(n^{1-\epsilon})$ for any constant $\epsilon > 0$ in time $2^{O(\log^2 n)}$.*

Proof.
Fix $\epsilon \in (0, 1/2)$, and let c, δ be parameters, with $c \in$

[3] Note that n must be sufficiently large, and δ, L must be chosen such that $(1-\delta)\delta^L n \in \mathbb{N}$ because it doesn't make sense to talk of "fractional cups".

$(0,1), 0 < \delta \ll 1/2$ – these will depend on ϵ, n. Say that we aim to achieve backlog at least $cn^{1-\epsilon}$. Observe that if we apply the Amplification Lemma to a function f satisfying $f(k) \geq ck^{1-\epsilon}$ for $k \leq g$ then for any k_0 with $k_0(1-\delta) \leq g$ (which enforces $k_0 \leq g/(1-\delta)$) we have the following:

$$f'(k_0) \geq$$

$$(1-\delta)\sum_{\ell=0}^{L} c(((1-\delta)\delta^\ell)k_0)^{1-\epsilon}$$

$$= ck_0^{1-\epsilon}(1-\delta)^{2-\epsilon}\sum_{\ell=0}^{L}(\delta^\ell)^{1-\epsilon},$$

where L is the greatest integer such that $(1-\delta)\delta^L n \geq n_0$ where n_0 is a constant such that a filler can achieve backlog 1 on n_0 cups (this definition is identical to the definition in the statement of Lemma 1). Note that as δ will be very small, $\sum_{\ell=0}^{L}(\delta^L)^{1-\epsilon}$ is very well approximated by $1+\delta^{1-\epsilon}$, so we will not loose much by relaxing our lower bound on $f'(k_0)$ to only use the first 2 terms of the sum. We have the looser bound

$$f'(k_0) \geq ck_0^{1-\epsilon}(1-\delta)^{2-\epsilon}(1+\delta^{1-\epsilon}).$$

Let

$$h(\delta) = (1-\delta)^{2-\epsilon}(1+\delta^{1-\epsilon}).$$

We prove the following claim:

Claim 1. *There exists an appropriate choice of δ that is small enough such that $h(\delta) > 1$ and large enough such that $L \geq 1$, when ϵ is chosen to be $4/\lg n$, or a positive constant. In particular, if ϵ is chosen to be $4/\lg n$ then we will choose $\delta \leq O(1/n)$, and if ϵ is chosen to be a positive constant then we will choose $\delta \leq O(1)$.*

Note that if $h(\delta) \geq 1$, then $f'(k_0) \geq ck_0^{1-\epsilon}$, meaning we have constructed from f a new function f' that satisfies the inequality $f'(k) \geq ck^{1-\epsilon}$ for $k \leq g/(1-\delta)$, as opposed to only for $k \leq g$ as in the case of f. [4] Then by repeatedly amplifying a function, we should be able to arbitrarily extend the support, which will allow us to attain the desired backlog.

We now prove Claim 1.

Proof. Consider the Taylor series for $(1-\delta)^{2-\epsilon}$ about $\delta = 0$:

$$(1-\delta)^{2-\epsilon} = 1 - (2-\epsilon)\delta + O(\delta^2).$$

So, to find a δ where $h(\delta) \geq 1$ it suffices – note that, again, we choose to neglect the δ^2 term as it does not strengthen the lower bound substantially because it is so small – to find a δ with

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

Rearranging we have

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

This clearly is true for sufficiently small δ, as $\delta^{1-\epsilon}$ will be much greater than δ or $\delta^{2-\epsilon}$. However it will be beneficial to have a more explicit criterion for possible choices of δ in terms of ϵ. To get this, we enforce a much stronger inequality on $\delta^{1-\epsilon}$ by vastly overestimating $\delta^{2-\epsilon}$ as δ. Surprisingly even with this overestimate we are still able to get the desired value of ϵ to work, as we will demonstrate later. We have,

$$\delta \leq \frac{1}{(2(2-\epsilon))^{1/\epsilon}}. \tag{1}$$

In addition to the constraint that δ must be small enough such that $h(\delta) \geq 1$, the only other constraint on δ is that δ must be large enough that the sum from the Amplification Lemma has at least two terms, i.e. such that $L \geq 1$. We need $L \geq 1$ because otherwise the Amplification Lemma doesn't give a larger function. The condition $L \geq 1$ enforces

$$\delta(1-\delta)n \geq n_0.$$

Recall that we choose $\delta < 1/2$, so $1-\delta > 1/2$. Thus to make δ sufficiently big it suffices to chose δ with

$$\delta \geq 2n_0/n. \tag{2}$$

Any choice of δ that is sufficiently large to make $L \geq 1$ and simultaneously small enough to make $h(\delta) \geq 1$ is a valid choice of δ. That is, δ is valid if and only if it satisfies

$$\frac{2n_0}{n} \leq \delta \leq \frac{1}{(2(2-\epsilon))^{1/\epsilon}}. \tag{3}$$

To achieve the desired backlog of $\Omega(n)$ we can use $\epsilon = \gamma/\lg n$ for appropriate constant γ, as

$$n^{1-\gamma/\lg n} = n/2^\gamma = \Omega(n).$$

We show that there is a valid choice of γ such that the following inequality is satisfied:

$$2n_0/n \leq \frac{1}{(2(2-\gamma/\lg n))^{(1/\gamma)\lg n}}. \tag{4}$$

Note that

$$(2(2-\gamma/\lg n))^{(1/\gamma)\lg n} \leq 4^{(1/\gamma)\lg n} \leq n^{2/\gamma}.$$

Thus, clearly by choosing e.g. $\gamma = 4$ we have the desired inequality. Inequality 4 implies that there is a valid choice of δ when we chose $\epsilon = \gamma/\lg n$. When proving that we can achieve backlog $\Omega(n)$ we use $\epsilon = 4/\lg n$, and $\delta = O(1/n)$ satisfying Inequality 3 for our choice of ϵ. When proving that we can achieve backlog $\Omega(n^{1-\epsilon})$ for constant $\epsilon > 0$ we choose $\delta > 0$ to be a constant satisfying Inequality 1, and δ, being constant, trivially satisfies Inequality 2. □

Now we proceed to show that with the appropriate values of δ, ϵ we can achieve a filling strategy that achieves backlog $cn^{1-\epsilon}$ on n cups. First we present a simple existential argument to show that an algorithm achieving backlog $\Omega(n^{1-\epsilon})$ for ϵ constant exists. Next we modify the existential proof to achieve an algorithm achieving the same backlog, but in bounded running time, specifically running time $2^{O(\log^2 n)}$. Finally we present an algorithm where δ is set extremally; doing so, along with other modifications of the approach, we can achieve backlog $\Omega(n)$ in running time $2^{O(n)}$.

[4] Note that although $f'(k) \geq ck^{1-\epsilon}$ holds for at least as many k as $f(k) \geq ck^{1-\epsilon}$, it does not necessarily hold for strictly more; in particular, if $\lfloor g/(1-\delta) \rfloor = g$ then the inequality on f' holds for no more k than the inequality on f, as f and f' are functions on $\mathbb{N}$. In general we have to be somewhat careful about the fact that there are an integer number of cups throughout this proof (this issue was deferred from earlier proofs to be dealt with here).

Proposition 2. *There exists a filling algorithm that achieves backlog $\Omega(n^{1-\epsilon})$ for any constant $\epsilon > 0$.*

Proof. Let $\epsilon > 0$ be constant. By Claim 1, there is a valid constant setting of δ; let $\delta \ll 1/2$ be an appropriate constant. Let $f^*(n)$ be the supremum over all filling strategies of the backlog achievable on n cups. Clearly f^* must be greater than or equal to the amplification of f^*. Assume for contradiction that there is some least n_* such that

$$\begin{cases} f^*(k) < ck^{1-\epsilon}, & k > n_* \\ f^*(k) > ck^{1-\epsilon}, & k \le n_* \end{cases}$$

Note that $n_*(1-\delta)\delta \ge n_0/(1-\delta)$ by appropriate choice of constant c, and Proposition 1, which states that we can get backlog $O(\log n_*)$ on n_* cups[5]. Because f^* satisfies the Amplification Lemma we have:

$$\begin{aligned} &f^*(n_*) \\ &\quad \ge (1-\delta)\sum_{\ell=0}^{L} f^*((1-\delta)\delta^\ell n_*) \\ &\quad \ge cn_*^{1-\epsilon} h(\delta) \\ &\quad \ge cn_*^{1-\epsilon} \end{aligned}$$

which is a contradiction. Hence f^* achieves backlog $cn^{1-\epsilon}$. □

Proposition 3. *We can construct a filling strategy that achieves backlog $\Omega(n^{1-\epsilon})$ for constant $\epsilon > 0$ in time $2^{O(\log^2 n)}$.*

Proof. It is desirable to have an algorithm for achieving this backlog with bounded running time; we now modify the existential argument to make it constructive, which yields an algorithm for achieving backlog $cn^{1-\epsilon}$ on n cups in finite running time. We again use constant $\epsilon > 0$ and appropriate constant δ.

We start with the algorithm given by Proposition 1 for achieving backlog

$$f_0(k) = \begin{cases} \lg k, & k \ge 1, \\ 0 & \text{else.} \end{cases}$$

Then we construct an algorithm that achieves better backlog using the Amplification Lemma (Lemma 1): we construct f_{i+1} as the amplification of f_i.

Define a sequence $g_i \in \mathbb{N}^\infty$ with

$$g_i = \begin{cases} \lceil 1/\delta \rceil \gg 1, & i = 0, \\ \lceil g_{i-1}/(1-\delta) \rceil - 1 & i \ge 1 \end{cases}$$

That is, g_{i+1} is the greatest integer strictly less than $g_i/(1-\delta)$. Note that $(1/\delta)/(1-\delta) > (1+\delta)/\delta = 1/\delta + 1$. Thus $g_1 = 1 + g_0$, and in general, $g_{i+1} > g_i$, because the difference $g_{i+1} - g_i$ can only grow as i grows.

We claim the following regarding this construction:

Claim 2.

$$f_i(k) \ge ck^{1-\epsilon} \text{ for all } k < g_i. \tag{5}$$

Proof. We prove Claim 2 by induction on i. Claim 2 is true in the base case of f_0 by taking c sufficiently small, in particular small enough that $f_0(k) \ge ck^{1-\epsilon}$ holds for $k < g_0$.[6] As our inductive hypothesis we assume Claim 2 for f_i; we aim to show that Claim 2 holds for f_{i+1}. Note the key property of g_i, that $g_{i+1} \cdot (1-\delta) < g_i$. Also note that (at least without loss of generality) the f_i are monotonically increasing functions: given more cups we can always achieve higher fill than with fewer cups. Thus we have, for any $k < g_{i+1}$,

$$\begin{aligned} &f_{i+1}(k) \\ &\quad \ge (1-\delta)\sum_{\ell=0}^{L} f_i((1-\delta)\delta^\ell k) \\ &\quad \ge ck^{1-\epsilon} h(\delta) \\ &\quad \ge ck^{1-\epsilon}, \end{aligned}$$

as desired. □

Note that $g_{i+1} \ge g_i + 1$ so by continuing this process we eventually reach some f_{i_*} such that $f_{i_*}(n) \ge cn^{1-\epsilon}$. Note that $i_* \le n$. Let the running time $f_{i_*}(n)$ be $T(n)$. Note that $f_{i_*}(n)$ must call $f_{i_*-1}(n(1-\delta)\delta^\ell)$ as many as $n(1-\delta)\delta^\ell$ times, for all $0 \le \ell \le L$. However, we only need the terms of the sum where $\ell = 0, 1$, so we can take $L = 1$. Thus we have the following (loose) recurrence bounding $T(n)$:

$$T(n) \le \delta n \cdot T(n(1-\delta)) + T(\delta n).$$

We can upper bound this by

$$n^{\frac{\log n}{\log(1/(1-\delta))}}.$$

Continuing for $O(\log n)$ levels of recursion should be sufficient to achieve the desired backlog. This gives running time

$$T(n) \le ((1+\delta)n)^{O(\log n)} \le 2^{O(\lg^2 n)}$$

as desired. ok, technically I'm ignoring integer problems in saying lets only do $O(\lg n)$ levels of recursion, it'll be enough. I should prove it. But for δ constant it seems pretty obvious.

□

Proposition 4. *We can construct a filling algorithm that achieves backlog $\Omega(n)$ in time $2^{O(n)}$.*

Proof. We describe a simple filling strategy that gives the desired backlog. Let $n_0 \le O(1)$ be a constant such that we can achieve backlog 1 on n_0 cups, and note that this is possible by Proposition 1. We construct a function that achieves large backlog on n cups. To achieve large backlog on n cups we first recursively apply our function to $(1-\delta)n$ cups repeatedly (for each of the δn cups that we are attempting to get high fill in), as described in the proof of the Amplification Lemma, and transfer over the cups that we get. Then we achieve backlog 1 on the δn

[5] Note: this is where it is crucial that ϵ, δ are constants.

[6] Note: this is where it is crucial that ϵ, δ are constants.

cups whose average fill has been increased. The backlog we achieve satisfies the following recurrence:

$$f(n) \geq \begin{cases} (1-\delta)f((1-\delta)n)+1, & \text{if } n\delta(1-\delta) > n_0 \\ 0, \text{ else.} \end{cases}$$

Let $(1-\delta)^c = \delta$, let $\delta^2 n < n_0 < (1-\delta)^{2c-1}n$ by our choice of $\delta = O(1/n)$. We can get backlog

$$\sum_{i=1}^{c}(1-\delta)^i.$$

To see this, consider a binary tree representing our algorithm. At every branch we both proceed to recurse on a $1-\delta$ fraction of the cups, and achieve backlog 1 on a δ fraction of the cups.

The sum evaluates to

$$\frac{(1-\delta)^2}{\delta}$$

which, if we chose $\delta = 1/n$, becomes $\Omega(n)$.

The running time satisfies the recurrence

$$T(n) = \delta n T((1-\delta)n) + O(1)$$

because to achieve backlog $f(n)$ we must achieve backlog $f((1-\delta)n)$ δn times, and then achieve backlog 1 on the remaining cups. Solving this recurrence yields that the running time is

$$\frac{(\delta n)^c - 1}{\delta n - 1}.$$

Recalling that $\delta = O(1/n)$ this becomes

$$2^{O(n)}.$$

□

□

4 Adaptive Filler Upper Bound

Let $\mu_S(X)$ and $m_S(X)$ denote the average fill and mass of a set of cups X at state S (e.g. $S = S_t$ or $S = I_t$).[7] Let $S_t(\{r_1, \ldots, r_m\})$ and $I_t(\{r_1, \ldots, r_m\})$ denote the cups of ranks $r_1, r_2, \ldots, r_m$ at states S_t and I_t respectively. Let $[n] = \{1, 2, \ldots, n\}$, let $i + [n] = \{i+1, i+2, \ldots, i+n\}$. We will use concatenation of sets to denote unions, i.e. $AB = A \cup B$.

We establish the following Lemma:

Lemma 2. *The greedy emptier maintains the invariant*

$$\mu_{S_t}(S_t([k])) \leq 2n - k \text{ for all } t \geq 1, k \in [n].$$

In particular, for $k = 1$, this implies that the greedy emptier never lets backlog exceed $O(n)$.

Proof. We prove the invariant by induction on t. The invariant holds 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 $k \in [n]$.

[7] Note that in the lower bound proofs (e.g. Section 3) when we used the notation m (for mass) and μ (for average fill), we omitted the subscript indicating the state at which the properties were measured. In those proofs it was sufficiently clear to leave the state implicit. However, in this section the state is crucial, and needs to be explicit in the notation.

Fix a round $t \geq 1$, and any $k \in [n]$. We assume 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, so we actually need all of them) 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 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 aren't among the k fullest cups in S_{t+1}. Let $C = I_t(p_t + [k-a])$, with $c = k - a = |C|$.

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. Further, note that $S_{t+1}([k]) = AC$ because once the cups in B are emptied from the cups in B are not among the k fullest cups, so the cups in C take their places among the k fullest cups.

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

Claim 3. *Without loss of generality*

$$S_t([r]) = I_t([r])$$

for any rank $r \in [n]$.

Proof. 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 $f_x > f_y$. Let the amount of fill that the emptier adds to these cups be $\Delta_x, \Delta_y \leq 1$; note that $f_x + \Delta_x < f_y + \Delta_y$.

Define a new state S'_t where cup x has fill f_y and cup y has fill f_x. Let the amount of water that the filler places in these cups from the new state be $f_x - f_y + \Delta_x$ and $f_y - f_x + \Delta_y$ for cups x, y respectively. This is valid as both fill amounts are at most 1: $f_x - f_y + \Delta_x < \Delta_y \leq 1$ and $f_y - f_x + \Delta_x < \Delta_x \leq 1$.

We can repeatedly apply this process to swap each cup in $I_t([r]) \setminus S_t([r])$ into being one of the $a+b$ fullest cups in the new state S'_t. At the end of this process we will have some "fake" state S^f_t. Note that S^f_t must satisfy the invariants if S_t satisfied the invariants, because our process can be thought of as just relabelling the cups; in particular $\text{fill}(S^f_t(r)) = \text{fill}(S_t(r))$ for all ranks $r \in [n]$.

It is without loss of generality that we start in state S^f_t because from state I_t we could equally well have come from state S_t or state S^f_t. Thus we consider state I_t to have come from state S^f_t. □

Now we proceed with the proof of the Lemma.

Claim 4. *If some cup in A zeros out then invariant holds.*

Proof. Say a cup in A zeroes out. Then
$$m_{S_{t+1}}(A) \le (a-1)(2n-(a-1))$$
because the $a-1$ fullest cups must have satisfied the invariant on round S_t, and $\text{fill}_{S_{t+1}}(I_{t+1}(a)) = 0$. Furthermore, the fill of all cups in C must be at most 1 in I_t to be less than the fill of the cup in A that zeroed out.

Thus,
$$m_{S_{t+1}}(S_{t+1}([k])) \le (a-1)(2n-(a-1)) + k - a.$$
We claim that this is less than $a(2n-a)$. This follows from simple manipulation of the above expression:
$$\begin{aligned}&(a-1)(2n-(a-1)) + k - a\\ &= a(2n-a) + a - 2n + a - 1 + k - a\\ &= a(2n-a) + (k-n) + (a-n) - 1\\ &< a(2n-a)\end{aligned}$$
as desired.

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

And we have
$$m_{S_{t+1}}(S_{t+1}([k])) \le k(2n-k).$$
□

Claim 5. *If no cups in A zero out and $b = 0$ the invariant holds.*

Proof. First we consider the case $b = 0$. If $b = 0$, then $S_{t+1}([k]) = S_t([k])$.

The emptier has removed 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:
$$m_{S_{t+1}}(S_{t+1}([k])) \le m_{S_t}(S_t([k])) + a - a \le k(2n-k).$$
□

Claim 6. *If no cups in A zero out and $b > 0$ the invariant holds.*

Proof. Because $b > 0$ and $a + b \le k$ 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 from 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 from 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:
$$m_{S_{t+1}}(S_{t+1}([k])) \le m_{S_t}(A) + m_{S_t}(C) + b.$$
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 quantifying exactly how much higher fill in A pushes down fill in C we can achieve the desired inequality. We can upper bound $m_{S_t}(C)$ by
$$m_{S_t}(C) \le \frac{c}{b+c} m_{S_t}(BC) = (m_{S_t}(ABC) - m_{S_t}(A)) \frac{c}{b+c}$$
because $\mu_{S_t}(C) \le \mu_{S_t}(B)$ without loss of generality by the interchangeability of cups. Thus we have
$$m_{S_t}(AC) \le m_{S_t}(A) + \frac{c}{b+c} m_{S_t}(BC) \tag{6}$$
where
$$\begin{aligned}&m_{S_t}(A) + \frac{c}{b+c} m_{S_t}(BC)\\ &= \frac{c}{b+c} m_{S_t}(ABC) + \frac{b}{b+c} m_{S_t}(A).\end{aligned} \tag{7}$$
Note that the expression in Equation 7 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 $(2n - |ABC|, 2n - |A|)$ we upper bound $m_{S_t}(A) + m_{S_t}(C)$. At this point the inequality can be verified by straightforward algebra, however this is not elegant; instead, we combinatorially interpret the sum.

We define a new "fake" state F, which may not represent a valid configuration of cups (i.e. might not satisfy the invariants), where $\mu_F(A) = 2n - |A|$ and $\mu_F(ABC) = 2n - |ABC|$, in particular with all the cups in A having identical fill, and all the cups in BC having identical fill. We can think of F as having come from a state where every cup has fill $\mu_F(ABC) = 2n - |ABC|$. To reach F from this state where every cup has identical fill we must increase the fill of each cup in A by some amount, and decrease the fill of each cup in BC by an amount such that the mass added to A is taken away from BC. To reach fill $\mu_F(A) = 2n - |A|$, the cups in A must have been increased by $|BC|$ from their previous fill of $2n - |ABC|$. To equalize an increase in $\mu_F(A)$ of $|BC|$, we need a corresponding decrease in $\mu_F(BC)$ by $|A|$. That is,
$$\mu_F(BC) = 2n - |ABC| - |A|.$$
Thus we have the following bound:
$$\begin{aligned}&m_{S_t}(A) + m_{S_t}(C)\\ &\le m_F(A) + c\mu_F(BC) &(*)\\ &\le a(2n-a) + c(2n - |ABC| - a)\\ &\le (a+c)(2n-a) - c(a+c+b)\\ &\le (a+c)(2n-a-c) - cb,\end{aligned}$$
where (*) follows from Equation 7.

Consider a new configuration of fills F achieved by starting with state S_t, and moving water from BC into A until $\mu_F(A) = 2n - |A|$. [8] This transformation increases (strictly increases if and only if we move a non-zero amount

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

of water) the mass in AC because water in BC counts less towards mass in AC than water in A by Inequality 6. In particular, if mass $\Delta \geq 0$ fill is moved from BC to A, then the mass of AC increases by $\frac{b}{b+c}\Delta \geq 0$.

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

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

Hence we must have

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

Thus

$$\mu_F(BC) \leq \mu_F(ABC) - |A| \leq 2n - |ABC| - |A|.$$

Thus we have the following bound:

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

Recall that we were considering $b > 0$, and since $b > 0$ we have that $c = k - a \geq b > 0$, i.e. $c \geq 1$. Hence we have

$$m_{S_t}(A) + m_{S_t}(C) \leq k(2n-k) - b$$

So

$$m_{S_t}(A) + m_{S_t}(C) + b \leq k(2n-k).$$

As shown previously the left hand side of the above expression is an upper bound for $m_{S_{t+1}}([k])$. Hence the invariant holds.

□

We have shown the inductive hypothesis 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 $t \in \mathbb{N}$. □

5 Conclusion

Many important open questions remain open.

References

[1] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, page 28, 1962.

[2] William Kuszmaul. Achieving optimal backlog in the vanilla multi-processor cup game. *SIAM*, 2020.