Optimal Time-Backlog Tradeoffs for the Variable-Processor Cup Game

William Kuszmaul ✉
Massachusetts Institute of Technology, Cambridge, MA, USA

Shyam Narayanan ✉
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

The *p-processor cup game* is a classic and widely studied scheduling problem that captures the setting in which a p-processor machine must assign tasks to processors over time in order to ensure that no individual task ever falls too far behind. The problem is formalized as a multi-round game in which two players, a filler (who assigns work to tasks) and an emptier (who schedules tasks) compete. The emptier's goal is to minimize backlog, which is the maximum amount of outstanding work for any task.

Recently, Kuszmaul and Westover (ITCS, 2021) proposed the *variable-processor cup game*, which considers the same problem, except that the amount of resources available to the players (i.e., the number p of processors) fluctuates between rounds of the game. They showed that this seemingly small modification fundamentally changes the dynamics of the game: whereas the optimal backlog in the fixed p-processor game is $\Theta(\log n)$, independent of p, the optimal backlog in the variable-processor game is $\Theta(n)$. The latter result was only known to apply to games with *exponentially many* rounds, however, and it has remained an open question what the optimal tradeoff between time and backlog is for shorter games.

This paper establishes a tight trade-off curve between time and backlog in the variable-processor cup game. We show that, for a game consisting of t rounds, the optimal backlog is $\Theta(b(t))$ where

$$b(t) = \begin{cases} t & \text{if } t \leq \log n \\ t^{1/3} \log^{2/3}\left(\frac{n^3}{t} + 1\right) & \text{if } \log n < t \leq n^3 \\ n & \text{if } n^3 < t. \end{cases}$$

An important consequence is that the optimal backlog is $\Theta(n)$ if and only if $t \geq \Omega(n^3)$. Our techniques also allow for us to resolve several other open questions concerning how the variable-processor cup game behaves in beyond-worst-case-analysis settings.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Parallel algorithms; Theory of computation → Scheduling algorithms

Keywords and phrases Cup Games, Potential Functions, Greedy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.85

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2205.01722

Funding *William Kuszmaul*: Funded by a Fannie & John Hertz Foundation Fellowship; by an NSF GRFP Fellowship; and by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.the United States Air Force Research Laboratory under Cooperative Agreement Number

49th International Colloquium on Automata, Languages, and Programming (ICALP 2022).
Editors: Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff;
Article No. 85; pp. 85:1–85:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.
Shyam Narayanan: Funded by an NSF GRFP Fellowship and a Simons Investigator Award.

1 Introduction

The classical p-processor cup game. The p processor cup game captures the general problem in which there are some number n of tasks competing for a smaller number p of processors [7, 21, 8, 33, 31, 37, 6, 24, 34, 35, 17, 10, 27, 1, 16, 32, 25]. A scheduler must assign tasks to processors over time in order to ensure that no individual task ever falls too far behind.

Formally, this is captured as a game with n cups, each capable of holding an arbitrarily large amount of water, and two competing players, a filler and an emptier. In each round of the game, the filler distributes p new units of water into the cups, placing at most 1 unit of water into any particular cup. The emptier then selects p distinct cups and removes up to 1 unit of water from each of them. Note that, whereas the filler may place their p units of water in fractional amounts across arbitrarily many cups, the emptier can only choose p cups per step to empty from. The emptier's goal is to minimize the backlog of the system, which is the amount of water in the fullest cup.

If one views the cup game as a scheduling problem, then the cups represent tasks, the water represents work, the filler represents an adversary that determines when work arrives, and the emptier represents a scheduler that can select p tasks to run on a given time step (we will use the terms "round" and "time step" interchangeably). Although we will primarily be interested in the cup game as a scheduling problem [7, 21, 8, 33, 31, 37, 6, 24, 34, 35, 1, 32, 17], it has also found applications to many other problems (e.g. deamortization of data structures [2, 17, 16, 3, 38, 23, 18, 26, 9], network-switch buffer management [22, 4, 39, 20], quality-of-service guarantees [7, 1, 32], etc.).

Beginning in the late 1960s, much of the early work on the p-processor cup game focused on the fixed-rate version of the game, in which the filler's behavior is the same at every round [7, 21, 8, 33, 31, 37, 6, 24, 34, 35]. In this version of the game, it is possible for the emptier to achieve a backlog of $O(1)$, both in the single-processor cup game (i.e., $p = 1$) [34, 35] and in the multi-processor cup game (i.e., $p > 1$) [7]. In recent decades, much of the research has shifted to focus on the non-fixed-rate version of the game, in which the filler is an adaptive adversary that can change their behavior from step to step [1, 16, 27, 32, 5, 19, 15]. In this setting, it is possible for the emptier to achieve backlog $O(\log n)$ [1, 16, 27], and this is known to be asymptotically optimal for all $p \leq n - \sqrt{n}$ [27]. There is also a long history of researchers applying techniques from beyond-worst-case analysis to cup games, e.g., resource augmentation [10, 27, 32, 17], smoothed analysis [27, 10], adversary restrictions [10, 27, 14, 28, 17], semi-clairvoyance [32], etc.

A repeating theme in these directions of work has been the relative difficulty of analyzing the multi-processor case in comparison to the single-processor case. As Liu discussed in his seminal 1969 paper [34], and as many later authors have subsequently reiterated [32, 7, 24, 10, 27], the difficulty of the multi-processor case stems from the fact that the emptier must remove water from p *distinct* cups, even if the vast majority of the water is in a smaller number of cups. For both the fixed-rate and the non-fixed-rate games, the optimal backlog in the multi-processor version of the game [7, 27] was proven decades after the corresponding result for the single-processor game was first shown [34, 16].

The variable-processor cup game. Recent work by Kuszmaul and Westover [29] has considered the question of what happens if the parameter p is permitted to change over time, with the filler adaptively determining both what value of p will be used at each round and where the p new units of water are placed. The resulting game, which is known as the variable-processor cup game, captures settings in which the amounts of resources available to the players fluctuate over time.[1]

The problem of what to do when computing resources fluctuate has received increasing attention in recent years due to the proliferation of shared-computing systems in which multiple users and virtual operating systems simultaneously run on a single physical multi-core machine; the fact that the machine is shared means that the amount of resources (eg., cache, processors, memory bandwidth, etc.) available to each user is constantly changing, depending on the current demands of other users. This phenomenon has led researchers to revisit problems in which computing resources have traditionally been viewed as static [36, 11, 12, 13, 30, 29].

Intuitively, the variable-processor cup game would seem to be relatively similar to its classical p-processor counterpart. Indeed, the backlog in the p-processor cup game is $O(\log n)$ regardless of the value of the parameter p, suggesting that the same should be true if p is permitted to vary. The central result of [29] is that this intuition turns out to be completely wrong: given sufficiently many time steps, the filler can actually force a backlog of $\Omega(n)$ in the variable-processor cup game, and this backlog is asymptotically optimal.

In order to achieve the backlog of $\Omega(n)$, the authors [29] construct a strategy for the filler in which the number of processors p follows a recursive "fractal-like" pattern. The recursive structure requires a relatively large number of time steps to complete – to achieve the full backlog of $\Omega(n)$, the construction requires exponentially many time steps.

The unexpectedly large backlog prompts several questions. The main open question is the problem of determining the optimal trade-off between backlog and time in the variable-processor cup game, and, in particular, what the optimal backlog is in games of polynomial lengths. In this setting, it is not even known whether the filler can achieve polynomial backlog – the best known filler strategy in this case [29] achieves backlog $2^{O(\sqrt{\log n})}$. Understanding the optimal trade-off between time and backlog in polynomial-length games is especially important since, for instance, one may have a bound on the number of rounds in a given scheduling application, which may allow for a much better guarantee on the backlog.

The authors of [29] also raise the question of whether smaller bounds on backlog can be achieved via beyond-worst-case analysis. Based on their results, they propose two directions, in particular, that seem promising. One is to place an additional restriction on the filler that p can only change at a certain rate; this would thwart the recursive structure of their lower bound construction which changes p dramatically between levels of recursion. The other is to consider the use of resource augmentation, meaning that the emptier is allowed to remove slightly more water in each time step than the filler is permitted to place into the cups. This direction seems promising due to the large amount of time required by the filling strategy of [29], since over such a large amount of time, the resource augmentation would potentially offer a large advantage to the emptier.

[1] As discussed in [29], there is no fundamental reason why the amount p of resources should fluctuate in the same way for the filler as it does for the emptier over time. However, by assuming that p is always the same for both players, one ensures that there is a fair playing field: neither player has an advantage over the other in terms of their resources.

Our results. The main result of this paper is a tight trade-off curve between time and backlog in the variable-processor cup game. We show that, for a game consisting of t rounds, the optimal backlog is $\Theta(b(t))$ where

$$b(t) = \begin{cases} t & \text{if } t \leq \log n \\ t^{1/3} \log^{2/3} \left(\frac{n^3}{t} + 1\right) & \text{if } \log n < t \leq n^3 \\ n & \text{if } n^3 < t. \end{cases} \tag{1}$$

By optimal, we mean that there exists an emptying strategy such that no filler can achieve backlog greater than $\Omega(b(t))$ after t rounds, and there exists a filling strategy such that no emptier can achieve backlog less than $O(b(t))$ after t rounds. In the case of the emptying strategy, we show that this tradeoff curve is achieved by the greedy emptying algorithm (i.e., always empty from the fullest cups).

Equation (1) comes with several interesting takeaways. The first is that in short games, of length $o(n^3)$, the emptier *can* achieve sub-linear backlog – prior to this result, it remained open whether the emptier could even ensure $o(n)$ backlog for only n rounds. The second is that in games of size $\Omega(n^3)$, the optimal backlog is $\Theta(n)$ – this resolves the open question of [29] as to whether linear backlog can be achieved in polynomial time. The third is that the optimal tradeoff between backlog and time has a somewhat unexpected polylogarithmic low-order term, which disappears only when t grows to be $\Omega(n^3)$.

By examining the inverse of the function $b(t)$, another way to think about (1) is that, for any quantity $b \leq n$, the amount of time $t(b)$ needed for an optimal filler to force a backlog of $\Omega(b)$ against an optimal emptier is

$$t(b) = \Theta\left(b + \frac{b^3}{\log^2 \frac{n}{b}}\right), \tag{2}$$

and that backlogs $b = \omega(n)$ are not achievable by the filler (the latter fact, of course, is already known due to [29]).

The second contribution of this paper is to analyze the variable-processor cup game under two forms of beyond-worst-case analysis, each of which resolves an open question posed by [29]. We begin by considering the setting in which the rate at which the filler can change p is severely limited: p is permitted to change by most ± 1 per time step, and the filler can only change p every $\text{poly}(n)$ time steps. Remarkably, this has no effect on the optimal backlog, and the filler can still force a backlog of $n/2$ in polynomial time. Next, we consider the setting in which the emptier has $\varepsilon > 0$ resource augmentation, meaning that, in each time step, the emptier is permitted to remove up to $1 + \varepsilon$ (rather than 1) units of water from each of p cups. This has a dramatic effect on the optimal backlog, reducing it to $O(\frac{1}{\varepsilon} \log n)$, which asymptotically matches the optimal backlog of the standard p-processor cup game when $\varepsilon = \Omega(1)$.

Techniques and paper outline. We formally describe the necessary preliminaries in Section 2, and provide a detailed overview of the technical ideas in the paper in Section 3.

In Section 4, we prove a result that ends up being useful in many of the later sections, namely that the greedy emptying algorithm is actually the *exact* optimal online algorithm, and that this holds no matter what the starting state of the cups is. Interestingly, our proof of greedy emptying being the optimal emptying strategy also applies to the fixed p-processor cup game, for which the result was also not previously known: in this setting, the greedy emptying algorithm was previously only known to be *asymptotically* optimal [27], and this

was only known for the starting state with all empty fills. We remark that, although the fact that greedy emptying is optimal is certainly intuitive, it actually isn't true for every variant of the cup game; notably, the greedy emptying algorithm isn't even asymptotically optimal for the fixed-rate version of the game [1].

In Section 5, we construct an asymptotically optimal strategy for the filler. The strategy achieves the bound in (1) no matter what strategy the emptier follows, and the strategy also easily generalizes the case where the filler is restricted to change p at a slow rate. This implies that the optimal backlog $b(t)$ is *at least* that in (1). While the filling algorithm works against any emptier, we focus on the filler working against the greedy emptier, which we know is optimal. We start with a warm-up (subsection 5.1) proving that one can establish $\Omega(n)$ backlog in $O(n^3)$ time, and we then generalize our techniques to apply to games of arbitrary lengths.

The remainder of the paper has been deferred to the extended version of this paper[2]. In Section 6 in the extended version, we turn our attention to proving a tight upper bound for the maximum backlog against a greedy emptier, that is, we show that $b(t)$ is *at most* that in (1). We begin by creating a variation of the variable-processor cup game, which we call the ***stone game***, in which the filler's behavior is limited in a certain combinatorially natural way. We analyze the maximum backlog of any filling strategy for the stone game by devising two potential functions and comparing their growth rates; this allows us to establish that $\Omega(n^3)$ time steps are needed to achieve backlog $\Omega(n)$ in the stone game. We then tighten the bound on backlog when there are fewer time steps by partitioning the cups into levels and arguing that a constant fraction of the levels interact especially nicely with the potential functions; this yields (1) for the stone game. Finally, we show that, if the emptier behaves greedily, then this stone game encapsulates the main problem, that is, is always advantageous to the filler in the variable-processor cup game to act as though they are in the stone game. Thus we can transfer out bounds on the stone game into bounds on the variable-processor cup game.

Section 6 is, in our opinion, the most technically involved section in our work. Perhaps the most interesting mathematical contribution in this section is to analyze the furthest backlog in the stone game by constructing two different potential functions and comparing their relative growth rates. While individual potential functions have been used to analyze cup games [10], this is the first time that comparing two potential functions has been applied to cup games.

Finally, in Section 7 in the extended version, we give a very simple analysis of the variable-processor cup game in the presence of resource augmentation. Our argument is non-constructive, employing the probabilistic method in order to show that there *exists* an emptying algorithm that achieves backlog $O(\frac{1}{\varepsilon}\log n)$ (interestingly, the same argument also gives a nontrivial bound of $O(\sqrt{t\log n})$ in the resource-augmentation-free setting). Since the greedy emptying algorithm is optimal, it must also achieve the same bound. To the best of our knowledge, this is the first example of the probabilistic method being used to analyze cup games.

2 Preliminaries

We first formally define the ***variable-processor cup game***. In this game, there are n cups of real-valued fills $x_1, \ldots, x_n$, all starting at 0, and two *adaptive* players, a filler and an emptier. At each round, the filler chooses an integer $1 \le p \le n$, chooses real numbers

[2] Available at `https://arxiv.org/abs/2205.01722`

$a_1, \ldots, a_n$ such that $0 \le a_i \le 1$ for all i and $\sum_{i=1}^{n} a_i = p$, and replaces x_i with $x_i' = x_i + a_i$ for all $1 \le i \le n$. The emptier then chooses a set $S \subset [n]$ of size p, and for each $i \in S$, replaces x_i' with $\max(0, x_i' - 1)$ but does not change x_i' for $i \notin S$. A single ***round*** (which we also call ***time step***) consists of both the filler's and emptier's moves. We define the ***state*** of the cups at a fixed round to be the current sequence $\{x_1, x_2, \ldots, x_n\}$ of the values of cups. Since the filler and emptier are adaptive, we note that two states are equivalent if the sequences are equal up to permutation. We will also say that the ***fills*** are $x_1, x_2, \ldots, x_n$ (we think of x_i as the fill of the i^{th} cup). Finally, for any state $X = \{x_1, \ldots, x_n\}$, we define the ***backlog*** of X as the maximum fill, or $\max_{1 \le i \le n} x_i$. The goal of the filler is to maximize the backlog after t rounds for some fixed t, whereas the goal of the emptier is to minimize it. We also define the ***variable-processor cup game with ε resource augmentation*** as the same as the variable-processor cup game, except that the emptier, for each $i \in S$, replaces x_i' with $\max(0, x_i' - (1 + \varepsilon))$.

Next, we define the ***negative-fill variable-processor cup game*** as the same as the variable-processor cup game, except that the emptier, for each $i \in S$, replaces x_i' with $x_i' - 1$. This may mean that some of the fills in a state become negative, which is allowed. We analogously define round, state, fills, and backlog (note that the backlog is $\max_{1 \le i \le n} x_i$, not $\max_{1 \le i \le n} |x_i|$).

Unless explicitly stated otherwise, the *standard* game refers to the variable-processor cup game, and the *negative-fill* game refers to the negative-fill variable-processor cup game. We will explicitly state whenever we talk about the fixed p-processor cup game (where $p = \sum_{i=1}^{n} a_i$ is fixed for every round), or the variable-processor cup game with $\varepsilon > 0$ resource augmentation.

We conclude this section by briefly commenting on the relationship between the standard game and the negative-fill game. It is easy to see that the optimal backlog in the standard game is at least as large as the optimal backlog in the negative-fill game. What is less obvious, but worth noting, is that the optimal backlogs in the two games are actually asymptotically *equal*. We provide a proof of this in Appendix A in the extended version. Thus, in general, either version of the game is equally valid (although the only place where we will take advantage of this in any nontrivial way will be Section 6 in the extended version).

3 Technical Overview

In this section, we provide a technical overview for both the lower and upper bound for the variable-processor cup game, as well as the upper bound for the variable-processor cup game with resource augmentation.

3.1 Overview of the Lower Bound on Backlog

In Section 5, we provide a lower bound on the backlog that the filler can achieve over t rounds. For now, let us assume that we are playing the negative-fill game and that the emptier is greedy, i.e., at each time step, if the filler fills p units of water, then the emptier empties 1 unit from the p fullest cups. We shall remove these assumptions at the end of the subsection.

Achieving backlog $\Omega(n)$ in n^3 steps. Why is this type of move good for the filler? Consider the potential function measuring the sum of squares of the fills, i.e., $\Phi(x_1, \ldots, x_n) := \sum_{i=1}^{n} x_i^2$. If we let $q = \frac{j-i+1}{2}$, so that q of the fills go up by $1/2$ and q of the fills go down by $1/2$, then it is not hard to show that Φ increases by $\frac{q}{2} \ge \frac{1}{2}$. If the filler can force this to happen for n^3 consecutive time steps, then Φ will increase to $\Omega(n^3)$, which means that at least one of the

$|x_i|$'s must be $\Omega(n)$. If the filler is careful, then it turns out they can further ensure that the cups are symmetric (i.e., for every cup with fill s there is another cup with fill $-s$). Thus if $|x_i| \geq \Omega(n)$ for some n, then a backlog of $\Omega(n)$ has been achieved.

The only way that the filler might be prevented from performing this type of move for n^3 consecutive time steps is if, at some time step, no two cups have the same fills as each other. Note, however, that the filler always adds half-integer values to cups and the emptier always subtracts integer values, which means that $x_1, \ldots, x_n$ are always half-integers. So, if the x_i's are all distinct, then $\max_{1 \leq i \leq n} |x_i| \geq \frac{n-1}{4} = \Omega(n)$. Recalling that the filler can ensure symmetry of the cups, it follows that in this case the filler has also achieved an $\Omega(n)$ backlog in only n^3 time steps.

Considering smaller backlogs. What if we only want to reach some backlog $o(n)$? We will now describe how the filler can achieve backlog $\Omega(t \log(n/t))$ in $O(t^3 \log(n/t))$ time steps. Combining this with some edge cases (which we defer to Section 5) results an optimal filling strategy for any backlog.

First, we claim that within t^3 steps, the filler can cause $\Theta(n)$ cups to have fills $\Omega(t)$. To see this, note that if at least half of the cups have fills less than t, then by the pigeonhole principle, there must exist some half-integer s with $|s| \leq t$ and $\Omega(n/t)$ cups of fill exactly s. If the filler causes half of these cups to go up in fill by $1/2$ and half of them to go down in fill by $1/2$, then the net effect on Φ will be that it increases by $\Omega(n/t)$. The filler can repeatedly force Φ to increase by $\Omega(n/t)$, while keeping the maximum backlog at t, until either t^3 steps have passed and $\Phi = \Omega(n \cdot t^2)$ or until at least half of the cups have reached backlog $\pm t$. Either way, at least $\Theta(n)$ of the cups must have fills more than $\Omega(t)$ or less than $-\Omega(t)$, and recalling again that the filler can also ensure a symmetry of the cups, it follows that a constant fraction of the cups have fills $\Omega(t)$.

Once we have cn cups of fill $\Omega(t)$, for some constant $c > 0$, the filler can focus on these cups, and force $c^2 n$ of these cups to fill $2 \cdot \Omega(t)$, then $c^3 \cdot n$ to fill $3 \cdot \Omega(t)$, and so on for a logarithmic number of phases. Overall, one can achieve backlog $\Omega(t \log(n/t))$ in $O(t^3 \log(n/t))$ time steps.[3]

The final piece: establishing that greedy emptying is optimal. To conclude our overview of the lowerbound, let us revisit the assumptions that (a) we are playing the negative-fill game, and (b) the emptier is playing greedily. The first assumption is trivial to remove, since the filler in the standard game is strictly better off than the filler in the negative-fill game. The second assumption, that the emptier is playing greedily, requires us to prove that the greedy emptying algorithm is always optimal (this ends up being useful to have for later results as well).

To prove the greedy emptying is optimal, in Section 4, we construct a specially designed poset on the possible states X of the system. Say that a state $X = \{x_1, \ldots, x_n\}$ ***weakly monopolizes*** a state $Y = \{y_1, \ldots, y_n\}$ if it is possible to order the cups such that either:

- $x_i \geq y_i$ for all i;
- or we have (a) that $x_i = y_i$ for all $i > 2$, that (b) $x_1 > y_2$, and that (b) we can get from X to Y by removing exactly 1 unit of water from cup 1 and placing some quantity $0 \leq c \leq 1$ of water into cup 2.

[3] Note that one can only do $\log(n/t)$ phases, rather than the more intuitive $\log n$ phases, since we need at least t cups to reach fill $a \cdot t$ in order for anything to reach fill $(a+1) \cdot t$.

The transitive closure of weak monopolization induces a partial ordering on the set of all possible system states, where $A \geq B$ if there is a sequence $A = A_1, A_2, \ldots, A_j = B$ such that each A_i weakly monopolizes each A_{i+1}.

We prove that, given the choice between two states A, B with $A \geq B$, the emptier should always prefer state B. Furthermore, starting from any state X, greedy emptying always results in a state B that satisfies $B \leq A$ for every other state A that the emptier could have reached from X. Thus the greedy emptying algorithm is optimal.

3.2 Overview of the Upper Bound on Backlog

In Section 6 in the extended version, we show that the greedy emptying algorithm achieves an upper bound on backlog matching the lower bound of Section 5. We first consider a filler that is restricted to only making moves of the form described in Subsection 3.1, that is, so that the net effect of each round is that some number $2q$ of cups at some height k are replaced with q cups at height $k + 1/2$ and q cups at height $k - 1/2$. We will later show that these types of moves are (essentially) always optimal for the filler, which means this restriction is actually without loss of generality. Considering this restricted filler along with a greedy emptier leads to the following simple combinatorial problem, which we call the ***stone game*** (we call it the ***stone-variant cup game*** in Section 6):

> *Suppose you have n stones on a number line, all starting at 0. At each time step t, you may pick any point k on the number line with 2 or more stones, choose any integer $q \geq 1$ such that there are at least $2 \cdot q$ stones at k, and move q of the stones at position k to position $k - 1$ and q of the stones at position k to position $k + 1$. If you repeat this for T time steps, what is the furthest that any of the n stones may be from the origin?*

Analyzing the time to get a stone to position $\Omega(n)$. To analyze the stone game, we start by considering how long it takes for some stone to reach $b := n/10$ in absolute value. Our goal is to show that one needs at least $T = \Omega(n^3)$ time steps. To highlight the relationship between the cup game and the stone game, we shall sometimes refer to the distance of the furthest stone from the origin simply as the backlog.

Let the positions of the stones be $x_1, \ldots, x_n$. We start by recalling the potential function $\Phi = \sum_{i=1}^{n} x_i^2$. One would naturally hope that Φ could help us establish a bound of $T = \Omega(n^3)$ (just as it helped us with the $T = O(n^3)$ bound in Subsection 3.1). For example, one way to prove the $\Omega(n^3)$ time bound would be to show that Φ increases by at most $O(1)$ in each step, and that Φ takes a value of at least $\Omega(n^3)$ after the final step (i.e., when backlog $b = n/10$ is achieved). Unfortunately, we run into two problems. The first problem is that even if $\max |x_i| = b = n/10$, we may still have that $\sum_{i=1}^{n} x_i^2$ is just $O(n^2)$ after the final step (as opposed to the desired $\Omega(n^3)$). The second, more difficult problem is that the number of stones we move in each direction could be large at each time step. If we move q stones right and q stones left, Φ increases by $2q$ (rather than the desired $O(1)$), which could be as large as n. Together, these two problems mean that, a priori, we can only get a trivial $T = \Omega(n)$ bound, since the change in the potential function Φ is up to n at each time step and the final potential may be as small as $\Theta(n^2)$.

The first problem can be resolved with a more careful analysis: in particular, one can show that a backlog of $\Omega(n)$ actually does imply $\Phi = \Omega(n^3)$. The key is to prove that there can never be large gaps between consecutive stones. Namely, one can show that if there exists a stone at some position $k > 0$, there must be at least one stone at position $k - 1$

or $k-2$, and likewise, if there exists a stone at $k<0$, there must be at least one stone at position $k+1$ or $k+2$. As a result, if there is a stone at $b=n/10$, there must be a stone at position either $b-1$ or $b-2$, at either $b-3$ or $b-4$, and so on, so one can show that there are $\Omega(b)$ stones at positions $b/2$ or greater. Thus, if there is a stone at position b, then $\Phi \geq \Omega(b)\cdot(b/2)^2 \geq \Omega(n^3)$, as desired.

The more difficult piece of the analysis is to show that, even though Φ can grow significantly in a single step, Φ only increases *on average* by $O(1)$ per step. An important insight here is to create a second potential function, Ψ, and compare the growth rates of Ψ and Φ. We define this new potential function $\Psi := \sum_{i<j} |x_i - x_j|$, where $x_1, \ldots, x_n$ are the locations of the n stones. In a given step of the stone game, if we move q stones up from k to $k+1$ and q stones down from k to $k-1$, the first potential function Φ increases by $2q$. However, one can show that Ψ must increase by at least $2q^2$ – the core reason for this is that the q stones that moved up now each have distance 2 from the q stones that moved down, whereas before they had distance 0. Now suppose for contradiction that Φ grows on average by some amount $\overline{q} = \omega(1)$ per step. This would mean that the average growth of Ψ per step is at least $\Omega(\overline{q}^2)$, so the final values of Ψ and Φ must satisfy $\Psi = \omega(\Phi)$. However, we know that $\Phi = \Omega(n^3)$ at the end of the game, and it is not hard to see that Ψ must be $O(n^3)$, since all of the x_i's are bounded in the range $[-O(n), O(n)]$. Thus we have a contradiction, and the average growth of Φ per step is actually $O(1)$. The fact that Φ is $\Omega(n^3)$ at the end of the game and grows by $O(1)$ on average per step is sufficient to show that the time T needed to achieve backlog $b=n/10$ in the stone game is at least $\Omega(n^3)$ time steps.

Considering smaller backlogs in the stone game. But what happens if we wish to analyze the time needed to achieve backlog k in the stone game, for $k \ll n$? If, at the end of the game, we knew that at least a constant fraction of the stones were in positions $\Omega(k)$, then we could use the same argument as the one above to show that the average growth rate of Φ is $O(1)$. Unfortunately, the number of stones at positions $\Omega(k)$ might be as small as $O(k)$. In this case, Φ could be as small as $O(k^3)$, whereas Ψ must be at least $\Omega(k^2 n)$. If the average growth of Φ were $\Theta(n/k)$ and the average growth of Ψ were $\Theta((n/k)^2)$, these values of Φ and Ψ could be obtained in $O(k^4/n)$ rounds, which is much smaller than our desired bound of $\Omega(k^3/\log^2(n/k))$.

To obtain the optimal bound of $\Omega(k^3/\log^2(n/k))$, we introduce another variant of the stone game that has what we call checkpoints: namely, we choose an integer ℓ and play the same game but now, once a stone has reached a position $a\cdot\ell$ for any $a \geq 0$, it can never go below it. In other words, if we move q stones from $a\cdot\ell$ to $a\cdot\ell+1$, rather than moving q stones from $a\cdot\ell$ down to $a\cdot\ell-1$, we keep them as is. A key insight is that, if a player wishes to get a stone to position $10\log n\cdot\ell$, the player must have the property that, for the majority of the checkpoints, the player gets at least half of the stones that reach checkpoint $a\cdot\ell$ all the way up to checkpoint $(a+1)\cdot\ell$.

We can think of the steps of the stone game (with checkpoints) as being split into subgames, where each subgame takes place between two checkpoints $a\cdot\ell$ and $(a+1)\cdot\ell$. We analyze each subgame individually by creating potential functions Φ_a, Ψ_a between each pair of checkpoints at $a\cdot\ell$ and $(a+1)\cdot\ell$. At least half of the subgames have the property that at least half of their stones make it to the next checkpoint, and this property makes each such subgame amenable to being analyzed using Φ_a and Ψ_a. By analyzing the subgames individually, we can show that the total length of the full stone game (which is the sum of the lengths of the subgames) is at least $\Omega(k^3/\log^2(n/k))$.

Finally, one can show that adding the checkpoints to the original stone game only makes it easier for the player to achieve a large backlog. This involves proving that if states X and Y have $x_i \geq y_i$ for all i, then given any stone-game operation on Y to obtain a state Y', we can generate a corresponding (but perhaps different) stone-game operation on X to obtain X', and preserve the ordering $x'_i \geq y'_i$ for all i. Thus, the $T = \Omega(k^3/\log^2(n/k))$ bound also applies to the original stone game.

Transferring bounds from the stone game to the cup game. Overall, the potential-function arguments above get optimal bounds for the *stone game* but they do not directly give bounds for the more general variable-processor cup game. The main issue is that in certain time steps in the cup game, it might be possible for the filler to make "backward" moves where both Φ and Ψ decrease considerably: in the worst case, Ψ can even decrease by up to n^2. Our analysis of the stone game relied heavily on the fact that the change in Ψ was comparable to the square of the change in Φ (in a given step), but if Ψ can decrease significantly in a single step, then our comparison of growth rates no longer works. Our fix for this is not to modify the potential functions, but rather to show that these backward moves, or any other "non-stone game" moves, are never advantageous to the filler, even in the long run.

To show that non-stone-game moves are never advantageous for the filler, we analyze the relationship between the stone game and the variable-processor cup game. Say that a sequence $X = \{x_1, \ldots, x_n\}$ of real numbers ***majorizes*** another sequence $Y = \{y_1, \ldots, y_n\}$ (where $x_1 \geq \cdots \geq x_n$ and $y_1 \geq \cdots \geq y_n$) if $x_1 + \cdots + x_i \geq y_1 + \cdots + y_i$ for all $1 \leq i \leq n$, and if $\sum_i x_i = \sum_i y_i$. We prove that, for any sequence of cup game rounds, one can create a corresponding sequence of stone game rounds such that after every round, the sequence of stone positions majorizes the sequence of cup fills. Since X majorizing Y implies that $x_1 \geq y_1$, we get that the maximum backlog after T steps of the cup game is at most the maximum backlog after T steps of the stone game. Therefore, even if we cannot utilize the potential functions on the general cup game, it suffices to look at the stone game as it will have a greater maximum backlog.

The main technical claim needed to show this majorization result is to show that if a sequence X majorizes a sequence Y, and if Y can be converted to a sequence Y' in one round of filling/greedy-emptying in the variable-processor cup game, then it is possible to convert X into some X' (also with a single round of filling/greedy emptying) such that X' majorizes Y'. This claim is quite casework-heavy and crucially uses the fact that we are in the *variable-processor* cup game. Indeed, the choice of p may have to differ between the round performed on X and the round performed on Y, and perhaps surprisingly, the claim is actually *false* in the fixed p-processor cup game.

3.3 Overview of Resource-Augmentation Analysis

In Section 7 in the extended version, we analyze the variable-processor cup game with ε resource augmentation (and non-negative fills), meaning that in each time step, the emptier is permitted to remove up to $1 + \varepsilon$ units of water from each of p cups (rather than just 1 unit of water).

We prove that even a very small amount of resource augmentation significantly decreases backlog of the game: the greedy emptying algorithm achieves backlog $O(\varepsilon^{-1} \log n)$.

The proof uses the probabilistic method. Rather than analyzing the greedy emptying algorithm directly, we instead construct a randomized emptying algorithm that, at any given moment, achieves backlog $O(\varepsilon^{-1} \log n)$ with *non-zero* probability. (Importantly, the randomized algorithm is against an *adaptive* filler, not an oblivious one.) The existence

of such a randomized algorithm non-constructively implies the existence of a deterministic emptying algorithm with the same guarantee. But we already know that the best deterministic emptying algorithm is the greedy one. Thus greedy emptying achieves backlog $O(\varepsilon^{-1} \log n)$ (deterministically).

To simplify our discussion here, let us consider only games of polynomial length (in Section 7 in the extended version we will consider arbitrary game lengths). In this case, our randomized emptying algorithm can simply take an approach that we call ***proportional emptying***: in each time step of the game, if the filler places some amount q_j of water into cup j, then the emptier empties from cup j with probability exactly q_j.

To analyze proportional emptying, we show that, at any given moment, each cup has fill $O(\varepsilon^{-1} \log n)$. Roughly speaking, the amount of water in each cup can then be modeled as a biased random walk: in each time step, the expected amount of water that the emptier removes is a factor of $1 + \varepsilon$ larger than the amount of water that the filler inserts. The bias in the random walk prevents it from ever reaching a large fill. The result is that a simple Chernoff-style analysis (modified using a variation on Azuma's martingale inequality to handle the fact that the filler is an adaptive adversary) can be used to bound the fill in each cup by $O(\varepsilon^{-1} \log n)$ (with high probability).

Interestingly, the above argument also immediately yields a nontrivial bound in the resource-augmentation-free setting. Now the amount of water in each cup follows an *unbiased* random walk. At any given time step t, one can bound the height of such random walk by $O(\sqrt{t \log n})$ with high probability. Using the fact that greedy emptying is as good as any randomized emptying strategy, it follows that greedy emptying achieves backlog $O(\sqrt{t \log n})$ in a t time step game.

4 The Greedy Emptier is Always Optimal

Intuitively, the greedy emptying algorithm (i.e., always empty from the fullest cups) should be the optimal deterministic emptying algorithm (for both the p-processor cup game and the variable-processor cup game). This intuition is known to be true for the single-processor cup game starting in a state with all empty cups (in particular, the lower and upper bounds on backlog match in this case [1]), but whether or not the intuition is correct in general has remained an open question. (We remark that there are variants of the game, for example the fixed-rate cup game, where greedy emptying is *not* optimal, even asymptotically [1].) In this section, we prove that greedy emptying is, in fact, optimal for both of the p-processor cup game and the variable-processor cup game.. That is, for any starting state of the cups, and for any game length, greedy emptying is the best possible algorithm for minimizing backlog against an adaptive filler.

For any state S and any length t, define $\mathrm{OPT}(S, t)$ to be the supremum backlog that a filler can achieve at the end of a t-step game starting at state S assuming the emptier plays optimally. That is, $\mathrm{OPT}(S, 0)$ is just the amount of water in the fullest cup of S, and $\mathrm{OPT}(S, t)$ for $t > 0$ is defined by induction as

$$\sup_{S' \text{ reachable from } S \text{ by filler}} \quad \min_{S'' \text{ reachable from } S' \text{ by emptier}} \mathrm{OPT}(S'', t-1).$$

Note that this also allows for us to talk about the ***optimal emptier***, which is the emptier that achieves backlog at most $\mathrm{OPT}(S, t)$ in any t-step game starting at any state S.

Define GREEDY(S, t) to be the supremum backlog that a filler can achieve at the end of a t-step game starting at state S, assuming the emptier plays greedily. We wish to prove that OPT$(S, t) =$ GREEDY(S, t) for all S, t. Throughout the section we shall prove all of our results for both the versions of the games with non-negative fills and the versions of the games with negative fills.

We say that a state A ***monopolizes*** a state B if it is possible to assign labels $1, 2, \ldots, n$ to the cups in A and B such that:

- cups $3, 4, \ldots, n$ contain the same amounts of water in both states;
- cup 1 in A contains more water than cup 2 in B;
- cup 1 in A contains one more unit of water than cup 1 in B;
- cup 2 in B contains c more units of water than cup 2 in A for some $0 \leq c \leq 1$.

In other words, you can get from A to B by removing 1 unit of water from cup 1 and and placing $c \leq 1$ units into cup 2, and cup 1 in A contains more water than cup 2 in B.

We say that A ***dominates*** B if it is possible to label the cups such that each cup i in A contains at least as much water as cup i in B. Finally, we say that A ***weakly monopolizes*** B if either A monopolizes B, or A dominates B.

We now prove several properties of weak monopolization in the p-processor cup game (for any p). Our first lemma says that, if A weakly monopolizes B, then for any filler move on B, there is some filler move on A that preserves the weak monopolization of A over B.

▶ **Lemma 1.** *Suppose A weakly monopolizes B. Suppose that there is a p-processor filler move that transforms B into some B'. Then there exists a p-processor filler move that transforms A into some A' such that A' weakly monopolizes B'.*

Proof. If A dominates B, then the lemma is trivial. Thus we can assume that A monopolizes B. Let q be the amount by which cup 1 in A contains more water than cup 2 in B.

Define X to be a state that we reach from A if we perform the same filler move that transforms B into B'. If cup 1 in X contains more water than cup 2 in B' then we can set $A' = X$ and be done.

Suppose cup 1 in X does not contain more water than cup 2 in B'. Then, in the transformation from B to B', the filler must have placed at least q more water into cup 2 than into cup 1. Let r be the amount of water that the filler placed into cup 1, and let $r + q + s$ (for some $s \geq 0$) the amount of water that the filler placed into cup 2.

Now define A' to be the state that one would reach from A by performing the following p-processor filler move: place $r + s$ units of water into cup 1 and $q + r$ units of water into cup 2 (and then place water into cups $3, 4, \ldots, n$ in the same way as to transform B to B'). Cup 1 in A' contains the same amount of water as cup 2 in B', and cup 2 and A' contains the same amount of water as cup 1 in B'. Thus A' and B' are equivalent states, meaning that A' weakly monopolizes B'. ◀

Our next lemma says that, if A weakly monopolizes B, and if A is then greedily emptied from, then it is possible to empty from B in such a way that the monopolization relationship is preserved.

▶ **Lemma 2.** *Suppose A weakly monopolizes B. Let A' be the state reached if a p-processor emptier greedily empties from A. Then there exists a valid p-processor emptier move on B that achieves some state B' such that A' weakly monopolizes B'.*

Proof. If A dominates B, then the lemma is trivial. Thus we can assume that A monopolizes B.

Let a_1, a_2 denote the fills of cups 1 and 2 in A, and let b_1, b_2 denote the fills of cups 1 and 2 in B. So $a_1 = b_1 + 1$, $b_2 = a_2 + c$ for some $0 \leq c \leq 1$, and $a_1 > b_2$. Furthermore, let a_1', a_2' the fills of cups 1 and 2 in A', and let b_1', b_2' denote the fills of cups 1 and 2 in B' (once we've defined it).

If the greedy emptier on A empties from neither cup 1 nor cup 2, then the same set of empties on B results in a B' that is weakly monopolized by A'.

Next consider the case where the greedy emptier on A empties from both cups 1 and 2. Then we empty from the same set of cups in B to arrive at a state B'. In the negative-fill game, is immediate to see that A' monopolizes B'. In the standard game, we must be careful about the fact that one of b_1 or a_2 might be less than 1. If $b_1 < 1$, then $b_1' = 0 \leq a_2'$, and since $a_1 > b_2$, we also have $a_1' \geq b_2'$; thus A' dominates B' in this case. If $a_2 < 1$ but $b_1 \geq 1$, then we have $a_1' = b_1' + 1$ and $b_2' = a_2' + c'$ for some $0 \leq c \leq 1$; we also have $a_1' \geq b_2'$ (since $a_1 \geq b_2$), so A' monopolizes B' in this case.

Finally, consider the case where the greedy emptier on A empties from only one of the cups 1 or 2. Since $a_1 > b_2 \geq a_2$, the emptier must empty from cup 1. Suppose we construct B' by performing the same set of empties on B as were performed on A, but we remove water from cup 2 instead of cup 1. Then $a_1' = b_1'$ and $a_2' \geq b_2'$, so A' dominates B'. This completes the proof. ◀

By combining the previous two lemmas, we can arrive at the following.

▶ **Lemma 3.** *Consider either the p-processor cup game, for some p, or the variable processor cup game (and consider either the negative-fill case or the non-negative-fill case). Suppose A weakly monopolizes B. Then for any t,*

$$\text{GREEDY}(A, t) \geq \text{OPT}(B, t).$$

Proof. We prove the lemma by induction on t. In the base case of $t = 0$, the lemma reduces to showing that A has backlog at least as large as B; this follows from the fact that A weakly monopolizes B.

Now consider some $t > 0$, and suppose the result holds for $t - 1$. For each state B' that the filler can reach from B, Lemma 1 tells us that there must exist a state A' that the filler can reach from A in such that A' weakly monopolizes B'. Moreover, Lemma 2 tells us that, if greedy emptying from A' results in some state A'', then there must be an emptying move on B' that results in a state B'' such that A'' weakly monopolizes B''.

Note that, in the previous paragraph, the filler's move from B to B' (i.e., the choice of B') fully determines A' (by the construction in Lemma 1), A'' (by greedy emptying on A'), and B'' (by the construction in Lemma 2). Thus, we will think of A', A'', B'' as being functions of B'.

Expanding out $\text{GREEDY}(A, t)$, we have

$$\text{GREEDY}(A, t) \geq \sup_{B'} \text{GREEDY}(A'', t - 1).$$

By the inductive hypothesis, since each A'' weakly monopolizes B'', we have $\text{GREEDY}(A'', t - 1) \geq \text{OPT}(B'', t - 1)$. Thus

$$\text{GREEDY}(A, t) \geq \sup_{B'} \text{OPT}(B'', t - 1).$$

Since $\sup_{B'} \text{OPT}(B'', t - 1) \geq \text{OPT}(B, t)$, the lemma follows. ◀

Recall that our goal is to upperbound $\text{GREEDY}(A,t)$ by $\text{OPT}(A,t)$ for all A,t. Thus Lemma 3 may at first seem to be backward progress, since the lemma establishes a *lower bound* on $\text{GREEDY}(A,t)$. The trick to using Lemma 3 is that we will only ever apply the lemma to states A and game lengths t for which we have already proven by induction that $\text{OPT}(A,t) = \text{GREEDY}(A,t)$; in this setting, the lemma establishes that $\text{OPT}(A,t) \geq \text{OPT}(B,t)$ which, as we shall see, ends up being critical to the proof.

▶ **Theorem 4.** *Consider either the p-processor cup game, for some p, or the variable-processor cup game (and consider either the negative-fill case or the non-negative-fill case). For all states A and game lengths t,*

$$\text{GREEDY}(A,t) = \text{OPT}(A,t).$$

Proof. We prove the theorem by induction on t with a trivial base case of $t = 0$. Consider $t > 0$, and suppose the theorem holds for $t-1$.

Consider any state A' that the filler can reach from A. Let A'' be the state reached from A' if the emptier empties greedily, and let B be the state reached from A' if the emptier empties according to OPT. Since the transformation from A' to A'' empties from cups with at least as much water as the transformation from A' to B, there must be a sequence of states $X_0, X_1, \ldots, X_k$, with $X_0 = B$ and $X_k = A''$, such that each X_i weakly monopolizes X_{i-1}. In particular, one can define the X_i's such the only difference between each X_i and X_{i+1} is that, to get from A' to X_{i+1} instead of from A' to X_i, the emptier removes water from a cup j instead of a cup k, where cup j contains less water than cup k in state A'. It is straightforward to verify that this results in each X_{i+1} weakly monopolizing each X_i: if cup k (in state A') has fill at least 1, then X_{i+1} monopolizes X_i; and otherwise, both cups k and j (in state A') have fills less than 1, and thus X_{i+1} dominates X_i.

By the inductive hypothesis, we know that $\text{OPT}(X_i, t-1) = \text{GREEDY}(X_i, t-1)$ for each X_i. Thus Lemma 3 tells us that $\text{OPT}(X_i, t-1) \geq \text{OPT}(X_{i-1}, t-1)$. By transitivity, it follows that $\text{OPT}(X_k, t-1) \geq \text{OPT}(X_0, t-1)$, and thus $\text{OPT}(B, t-1) \geq \text{OPT}(A'', t-1)$. Again applying the inductive hypothesis to deduce that $\text{OPT}(A'', t-1) = \text{GREEDY}(A'', t-1)$, we have that

$$\text{OPT}(B, t-1) \geq \text{GREEDY}(A'', t-1).$$

Thus

$$\text{GREEDY}(A,t) = \sup_{A''} \text{GREEDY}(A'', t-1) \leq \sup_{B} \text{OPT}(B, t-1) = \text{OPT}(A,t).$$

Finally, as $\text{GREEDY}(A,t) \geq \text{OPT}(A,t)$ trivially, we have $\text{GREEDY}(A,t) = \text{OPT}(A,t)$. ◀

▶ **Corollary 5.** *Theorem 4 continues to hold for the game with non-negative fills even if the emptier is given $1+\varepsilon$ resource augmentation.*

Proof. The proof of Theorem 4 continues to hold without modification. The only difference is that, now, we say that a state A monopolizes a state B if there is a ordering of the cups for which two properties hold: (1) we can take $1+\varepsilon$ unit of water from some cup 1 in A, place some amount $0 \leq c \leq 1+\varepsilon$ of water into some other cup 2 in A, and in doing so arrive at B; and (2) cup 1 in A contains more water than cup 2 in B. ◀

5 Lower Bounding Backlog

In this section, we prove the optimal lower bound on the backlog. In other words, we show that for any integer $t \geq 1$, there exists a filling strategy that can guarantee a backlog of $\Omega(b(t))$ against *any* emptying strategy, after t rounds of the variable-processor cup game, where $b(t)$ is defined as in Equation (1) in the introduction (Section 1). This result can be captured compactly in the following theorem:

▶ **Theorem 6.** *For some absolute constant $c > 0$ and any $k \leq c \cdot n$, the filler in the variable-processor cup game can force a maximum backlog of $\Omega(k)$ using only $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps.*

We remark that due to the optimality of greedy emptying, which we proved in the previous section, we focus in this section on designing filling strategies that are effective against the greedy emptier.

In Subsection 5.1, we give a simple proof for obtaining $\Omega(n)$ backlog in $O(n^3)$ rounds. In Subsection 5.2, we generalize Subsection 5.1 and develop the full proof of Theorem 6.

5.1 Warmup: Achieving a $\Theta(n)$ Backlog Quickly Against a Greedy Emptier

In this subsection, we show that in only n^3 rounds, a filler can achieve a backlog of $\frac{n-1}{2}$ against a greedy emptier. While the following argument is simpler than the more refined lower bound in Subsection 5.2, it conveys much of the intuition behind the general lower bound. Here, we only consider the standard variable-processor cup game, though in subsection 5.2 we will consider both the standard and negative-fill games.

▶ **Theorem 7.** *Against a greedy emptier in the standard variable-processor cup game, the filler can achieve a backlog of $\frac{n-1}{2}$ in only n^3 rounds.*

Proof. The filler follows the following simple strategy, which will guarantee that the fills at the end of each round are half integers (i.e., integer multiples of $\frac{1}{2}$). Suppose the cups after some round, in sorted order, are $x_1 \geq x_2 \geq \cdots \geq x_n$, which are all half-integers.

If $x_i > x_{i+1}$ for all $1 \leq i \leq n-1$, then we must have that $x_i \geq x_{i+1} + \frac{1}{2}$ for all $i \leq n-1$, so $x_1 \geq \frac{n-1}{2}$. In this case, we have already achieved a backlog of $\frac{n-1}{2}$.

Otherwise, we consider the potential function $\Phi = \sum_{i=1}^{n} x_i^2$, i.e., the sum of the squares of the backlogs. Consider the smallest p such that $x_p = x_{p+1}$. We set the number of processors for the round to be p, we place 1 unit of water into each of cups $x_1, \ldots, x_{p-1}$ and we place $1/2$ unit of water into each of cups x_p, x_{p+1}. Then, the greedy emptier will empty 1 unit from each of the first $p-1$ cups and will choose to empty 1 unit from either the p^{th} cup or the $(p+1)^{\text{th}}$ cup (WLOG, they choose the $(p+1)$-th cup). This means that, over the course of the entire round, the first $p-1$ cups are unchanged, the p^{th} cup goes up by $\frac{1}{2}$, and $(p+1)^{\text{th}}$ cup goes down by $\frac{1}{2}$. The only exception is if $x_p = x_{p+1} = 0$ at the beginning, in which case one of these two cups will go up to $1/2$ and the other remains at 0. Clearly, all of the fills remain half-integers at the end of each round. If we replace x_p and $x_{p+1} = x_p$ with $x_p + \frac{1}{2}$ and $x_p - \frac{1}{2}$, then the sum of the squares of the backlogs increases by $1/2$. Otherwise, if we replace $x_p = x_{p+1} = 0$ with 0 and $1/2$ in some order, then the sum of the squares of the backlogs increases by $1/4$. Thus, at the end of each round, unless one of the backlogs was already $\frac{n-1}{2}$ or greater, Φ increases by at least $\frac{1}{4}$.

Therefore, after n^3 rounds, either at some point we will have achieved $x_1 > x_2 > \cdots > x_n$ and thus a backlog of $\frac{n-1}{2}$, or we have that $\sum_{i=1}^{n} x_i^2 \geq \frac{n^3}{4}$, which would mean that $\max x_i \geq \frac{n}{2}$, since all of the x_i's are nonnegative. As a result, there exists a filling strategy that can guarantee a $\frac{n-1}{2}$ backlog against a greedy emptier in at most n^3 rounds. ◀

5.2 The General Lower Bound

In this subsection, we prove the lower bound on the backlog for the variable-processor cup game after t rounds, showing that the optimal backlog is $\Omega(b(t))$ for $b(t)$ defined as in Equation (1). Equivalently, we show that for any integer $k \leq n$, a filler can achieve backlog $\Omega(k)$ in $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ rounds against any emptying strategy. We remark that if we are playing t rounds of the game and the filler has achieved backlog b by round $t' < t$, the filler can ensure the backlog stays at b by setting $p = n$ and filling every cup for steps $t'+1, \ldots, t$. Thus, we just need to show the filler can obtain this backlog *within* this many rounds.

Throughout this subsection, we assume that we are playing the negative-fill game, and that the emptier is always greedy. We remove both of these assumptions at the end in Theorem 6.

▶ **Lemma 8.** *Assume that the emptier is always greedy, and that at the beginning of some round, the fills are all (possibly negative) half-integers. For some integer $k \in \mathbb{Z}$ and positive integer $q \in \mathbb{N}$, suppose there are at least $2q$ cups having fill exactly $k/2$. Then, the filler can ensure that at the end of the round (i.e., after both the filler and emptier move), exactly q of the cups of fill exactly $k/2$ have increased to $(k+1)/2$, exactly q of the cups of fill exactly $k/2$ have decreased to $(k-1)/2$, and all remaining cups are unchanged.*

Proof. Suppose the cups are sorted in order $x_1 \geq x_2 \geq \cdots \geq x_n$, such that $x_i = x_{i+1} = \cdots = x_{i+2q-1} = k/2$. Also, suppose i is the smallest integer such that $x_i = k/2$, so either $x_{i-1} > k/2$ or $i = 1$. Then, the filler will set $p = i - 1 + q$ and fill the first $i-1$ cups with 1 unit of water and the cups $x_i, \ldots, x_{i+2q-1}$ each with $1/2$ unit of water. Then, if $i > 1$, the emptier is forced to empty 1 unit of water from each of the first $i-1$ cups. In addition, we have that the cups $i, i+1, \ldots, i+2q-1$ all have fills $(k+1)/2$, which is at least $1/2$ unit of water more than all later cups, which means that the emptier will remove 1 unit of water from exactly q of the cups $i, i+1, \ldots, i+2q-1$. Therefore, all cups $1, \ldots, i-1$ are unchanged (since 1 unit of water is added and then removed) and all cups $i+2q, \ldots, n$ are unchanged (since water is never added nor removed). Finally, among the cups $i, i+1, \ldots, i+2q-1$, exactly q of them will end up at $(k-1)/2$ and exactly q of them will end up at $(k+1)/2$. ◀

▶ **Lemma 9.** *Suppose that k, m are positive integers such that $4k|m$, and suppose there are at least m cups that currently have fill 0 (where $n \geq m$ is the total number of cups). Then, against a greedy emptier, the filler can ensure that at least $m/4$ cups will have fill exactly $k/2$ after $O(k^3)$ rounds.*

Proof. WLOG assume that the first m cups have fills $x_1, x_2, \ldots, x_m$, and at the start, $x_1 = x_2 = \cdots = x_m = 0$. (We do not assume the cups are sorted in order of fill.) We only ever modify the first m cups, and after each step, we will maintain the following invariants:

1. All of the fills are half-integers.
2. For each integer j, the number of cups of fill $j/2$ equals the number of cups of fill $-j/2$.
3. For each integer j, the number of cups of fill $j/2$ is always a multiple of $\frac{m}{4k}$.
4. No cup has fill greater than $k/2$ or less than $-k/2$.

Trivially, all 4 of these invariants are true at the beginning, since all the fills are 0 and $\frac{m}{4k}|m$. Our procedure is the following. Suppose that there exists some integer j such that $-k < j < k$ and the number of cups of fill $j/2$ is at least $\frac{m}{2k}$. Then, we use Lemma 8 to move $\frac{m}{4k}$ of these cups to fill $(j+1)/2$ and move $\frac{m}{4k}$ of these cups to fill $(j-1)/2$. In addition, if $j \neq 0$, then we know the number of cups of fill $-j/2$ is at least $\frac{m}{2k}$. So, in the next step, we move $\frac{m}{4k}$ of these cups to fill $(-j+1)/2$ and move $\frac{m}{4k}$ of these cups to fill $(-j-1)/2$.

When $j \neq 0$, we perform these two steps consecutively as a pair. We keep repeating these types of steps (and pairs of steps) until we can no longer do so. It is clear that the first three invariants are preserved, and the last is preserved since we only modify cups that have fill between $-(k-1)/2$ and $(k-1)/2$, and we change their fills by at most $1/2$ per round.

Now, we note that at each step, the potential function $\Phi = \sum_{i=1}^{m} x_i^2$ increases by $\frac{m}{8k}$, since

$$\frac{m}{4k} \cdot \left(\frac{j+1}{2}\right)^2 + \frac{m}{4k} \cdot \left(\frac{j-1}{2}\right)^2 - \frac{m}{2k} \cdot \left(\frac{j}{2}\right)^2 = \frac{m}{8k}.$$

This also means that if we do a pair of steps (i.e., modifying cups of fill $j/2$ and $-j/2$), the potential function Φ increases by $\frac{m}{4k}$.

Now, after $4k^3$ steps (and pairs of steps), Φ is at least $4k^3 \cdot \frac{m}{8k} = \frac{1}{2} \cdot mk^2$. But this is impossible, since all of the fills of cups 1 through m are between $k/2$ and $-k/2$, so the maximum possible value of Φ is $\frac{1}{4} \cdot mk^2$. This means that before reaching $4k^3$ of steps or pairs of steps, we must not be able to do any more steps. That is, for all j such that $-k < j < k$, there are less than $\frac{m}{2k}$ cups of fill exactly $j/2$, and because of our third invariant, this means there are at most $\frac{m}{4k}$ cups of fill exactly $j/2$. Therefore, the number of cups of fill between $-k+1$ and $k-1$ is at most $(2k-1) \cdot \frac{m}{4k} \leq \frac{m}{2}$. So, at least $\frac{m}{2}$ cups must have fill $\pm\frac{k}{2}$, and by the second invariant, this means at least $\frac{m}{4}$ cups must have fill $\frac{k}{2}$. Reaching this stage required at most $4k^3$ steps and pairs of steps, which means the filler can succeed in at most $8k^3$ steps. ◀

▶ **Corollary 10.** *Suppose k, m are positive integers such that $4k|m$, and suppose there are at least m cups that currently have fill exactly t for some real number t. Then, after $O(k^3)$ time steps, the filler can force at least $m/4$ of these cups to have fill exactly $t + \frac{k}{2}$, against a greedy emptier.*

Proof. We can apply same argument as in Lemma 9, where all of the fills are shifted up by the same number t. So, after at most $8k^3$ time steps, at least $m/4$ of the cups will have fill exactly $t + \frac{k}{2}$. ◀

From here, we are able to conclude with our main result of this section.

▶ **Theorem 6.** *For some absolute constant $c > 0$ and any $k \leq c \cdot n$, the filler in the variable-processor cup game can force a maximum backlog of $\Omega(k)$ using only $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps.*

Proof. For now we will assume that the emptier is greedy, and that we are playing the negative-fill cup game. We will remove both assumptions at the end of the proof.

First, suppose $k \geq c \log n$. In this case, let $k' := \left\lceil \frac{k}{c \log(n/k)} \right\rceil \geq 2$. Also, let n' be such that $n \geq n' > \frac{n}{4}$ and $n' = k' \cdot 4^r$ for some integer $r \geq 1$.

Since there are at least $n' = k' \cdot 4^r$ cups of fill 0 at the beginning of the game, we can apply Corollary 10 with $t = 0$ and $m = n'$ to make sure there are at least $k' \cdot 4^{r-1}$ cups of fill k' after $O(k'^3)$ time steps, since $4k'|n'$. Then, we can again apply Corollary 10 with $t = k'$ and $m = n'/4$ to make sure there are at least $k' \cdot 4^{r-2}$ cups of fill $2k'$ after an additional $O(k'^3)$ time steps. We can repeat this r times until we have at least k' cups of fill $r \cdot k'$ after a total of $O(r \cdot k'^3)$ time steps. But since $r = \log_4(n'/k') = \Theta(\log(n/k \cdot \log(n/k))) = \Theta(\log(n/k))$, this means that the filler can achieve a maximum backlog of $r \cdot k' = \Theta(k)$ using only $O(r \cdot k'^3) = O(k^3/\log^2(n/k))$ time steps.

Next, suppose $k < c \log n$. In this case, set $n' = 2^k$: if $c < 1$ then $n' < n$. By Lemma 8, we can send 2^{k-1} cups to fill 1 during the first round, then 2^{k-2} of those cups to fill 2 during the second round, and so on until we have 1 cup at fill k after k rounds.

Up until now we have assumed that the filler was operating against a greedy emptying algorithm in the negative-fill cup game. However, by Theorem 4, if the emptier uses a different emptying algorithm, the filler could always modify their strategy to get equal or greater backlog in the same amount of time. So, regardless of the emptying strategy, the filler can force a maximum backlog of $\Theta(k)$ using $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps, assuming we are playing the negative-fill cup game. Finally, recall that the maximum backlog that the filler can ensure in the negative-fill game is at most the maximum backlog that the filler can ensure in the standard game for any fixed number of time steps. Thus, if the filler can force a maximum backlog of $\Theta(k)$ using $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps in the negative-fill game, then the filler can force the same backlog using the same number of time steps in the standard game, which completes the theorem. ◀

Finally, to conclude the section, we use the optimality of greedy emptying to resolve an open question of [29] concerning whether a filler who is limited in the speed at which they can change p can still force a large backlog.

▶ **Proposition 11.** *Consider the variable-processor cup game, starting with all empty cups (and with negative fills either allowed for disallowed). Suppose the filler is restricted to only change p once every n^c steps, for some constant $c \geq 0$, and to only change p by ± 1 at a time. Then the filler can still force a backlog of $\Omega(n)$ in polynomial time.*

Proof. Call this version of the game the *change-limited variable-processor cup game*. Notice that the proof of optimality for greedy emptying never changes the value of p used in any given step. Thus the same proof implies that greedy emptying is optimal in the change-limited variable-processor cup game. Henceforth we will assume that the emptier is greedy.

Next, observe that, no matter the value of p, the filler can always "skip" a step in the following way: the filler simply places 1 unit of water into each of the p fullest cups, forcing the greedy emptier to remove water from those cups, so that the step has no net effect.

We have already proven that, in the standard variable-processor cup game, the filler can cause backlog $\Omega(n)$ in polynomial time. The filler can use the same strategy here, but with the following modification: between any two consecutive steps that use p and q processors, respectively, the filler spends $|p - q|n^c$ steps changing the number of processors from p to q. That is, for each $i \in [p, q]$, the filler spends n^c steps with i processors, and renders each of those steps to have no net effect. The result is that, in polynomial time, the filler can force backlog $\Omega(n)$, 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. doi:10.1145/777412.777430.

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. doi:10.1006/inco.1995.1090.

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. URL: `http://dl.acm.org/citation.cfm?id=545381.545452`.

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, June 1996. `doi:10.1007/BF01940883`.

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 multi-processor 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 (SPAA)*, 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 (SPAA)*, 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 (SODA)*, pages 958–971. SIAM, 2014.

14 Michael A Bender and William Kuszmaul. Randomized cup game algorithms against strong adversaries. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2059–2077. SIAM, 2021.

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

16 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. `doi:10.1145/28395.28434`.

17 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. URL: `http://dl.acm.org/citation.cfm?id=127787.127809`.

18 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.

19 Rudolf Fleischer and Hisashi Koga. Balanced scheduling toward loss-free packet queuing and delay fairness. *Algorithmica*, 38(2):363–376, February 2004. `doi:10.1007/s00453-003-1064-z`.

20 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.

21 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.

22 Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010. doi:10.1145/1753171.1753195.

23 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.

24 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.

25 Sungjin Im, Benjamin Moseley, and Rudy Zhou. The matroid cup game. *Operations Research Letters*, 49(3):405–411, 2021.

26 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.

27 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.

28 William Kuszmaul. How asymmetry helps buffer management: achieving optimal tail size in cup games. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1248–1261, 2021.

29 William Kuszmaul and Alek Westover. The variable-processor cup game. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

30 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 (SPAA)*, pages 213–222, 2018.

31 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.

32 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. doi:10.1007/s00224-008-9134-x.

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

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

35 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.

36 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.

37 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.

38 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.

39 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.