Bamboo Trimming Revisited: Simple Algorithms Can Do Well Too

John Kuszmaul
Yale University
New Haven, Connecticut, USA
john.kuszmaul@gmail.com

ABSTRACT

The bamboo trimming problem considers n bamboo with growth rates $h_1, h_2, \ldots, h_n$ satisfying $\sum_i h_i = 1$. During a given unit of time, each bamboo grows by h_i, and then the bamboo-trimming algorithm gets to trim one of the bamboo back down to height zero. The goal is to minimize the height of the tallest bamboo, also known as the backlog. The bamboo trimming problem is closely related to many scheduling problems, and can be viewed as a variation of the widely-studied fixed-rate cup game, but with constant-factor resource augmentation.

Past work has given sophisticated pinwheel algorithms that achieve the optimal backlog of 2 in the bamboo trimming problem. It remained an open question, however, whether there exists a *simple* algorithm with the same guarantee—recent work has devoted considerable theoretical and experimental effort to answering this question. Two algorithms, in particular, have appeared as natural candidates: the **Reduce-Max** algorithm (which always cuts the tallest bamboo) and the **Reduce-Fastest**(x) algorithm (which cuts the fastest-growing bamboo out of those that have at least some height x). It is conjectured that **Reduce-Max** and **Reduce-Fastest**(1) both achieve backlog 2.

This paper improves the bounds for both **Reduce-Fastest** and **Reduce-Max**. Among other results, we show that the *exact optimal* backlog for **Reduce-Fastest**(x) is $x + 1$ for all $x \geq 2$ (this proves a conjecture of D'Emidio, Di Stefano, and Navarra in the case of $x = 2$), and we show that **Reduce-Fastest**(1) *does not* achieve backlog 2 (this disproves a conjecture of D'Emidio, Di Stefano, and Navarra).

Finally, we show that there is a different algorithm, which we call the **Deadline-Driven** Strategy, that is both very simple and achieves the optimal backlog of 2. This resolves the question as to whether there exists a simple worst-case optimal algorithm for the bamboo trimming problem.

CCS CONCEPTS

• **Theory of computation** → **Scheduling algorithms**; *Online algorithms*; **Design and analysis of algorithms**.

KEYWORDS

cup emptying; bamboo trimming; discretized scheduling; load balancing; parallelism

1 INTRODUCTION

A classic scheduling problem is the so-called ***cup game***, which is a two-player game that takes place on n cups. In each step of the game, the filler player distributes 1 unit of water among the cups arbitrarily; the emptier player then selects a single cup and removes up to 1 unit of water from that cup. The emptier's goal is to minimize the backlog of the system, which is defined to be the amount of water in the fullest cup.

The cup game was first introduced in the late 1960s [28, 29], and has been studied in many different forms [1, 5–7, 9, 13, 14, 18, 21, 23, 25–30]. The game has found extensive applications in areas such as processor scheduling [1, 5–7, 9, 10, 14, 18, 21, 23–30], network-switch buffer management [4, 17, 19, 32], quality of service guarantees [1, 6, 26], and data-structure deamortization [2, 3, 8, 13, 14, 16, 20, 22, 31]. See [23] for a detailed discussion of the related work.

Perhaps the most natural emptying algorithm is the **Reduce-Max** algorithm, which always empties from the fullest cup. **Reduce-Max** achieves an asymptotically optimal backlog of $O(\log n)$ [1, 13]. In fact, in addition to being asymptotically optimal, **Reduce-Max** is known to be *exactly* optimal—no other algorithm can do better, even by an additive constant [1].

An important special case of the cup game is the setting where the filler's behavior is the same on every step, also known as the fixed-rate cup game. Whereas the optimal backlog in the variable-rate cup game is $O(\log n)$, the optimal backlog in the fixed-rate cup game is $O(1)$ [5–7, 18, 21, 25, 27–30]. Perhaps surprisingly, though, the **Reduce-Max** algorithm is no longer optimal (or even asymptotically optimal!). In fact, the algorithm still allows for backlog $\Omega(\log n)$ in the fixed-rate setting [1].

Recent work has identified a potential path to redemption for the **Reduce-Max** algorithm, however. Bilò, Gualà, Leucci, Proietti, and Scornavacca [11] showed that, if the emptier is given resource augmentation over the filler, meaning that the emptier is permitted to fully empty a cup on each step rather than removing just a single unit of water, then the backlog achieved by the **Reduce-Max** algorithm becomes $O(1)$. Note that, since the backlog is constant,

the resource augmentation never results in the emptier removing more than $O(1)$ units of water at a time.

Although there is a long history of studying resource-augmented variants of the cup game [9, 13, 14, 26], it is only relatively recently that researchers have begun to study the resource-augmented fixed-rate version of the game [11, 15, 18]. These papers have dubbed the problem as the ***Bamboo Garden Trimming Problem***, based on the following (rather creative) problem interpretation. A robotic panda gardener is responsible for maintaining a bamboo garden. The garden consists of n bamboo $b_1, \ldots, b_n$ with corresponding growth-rates $h_1 \geq \ldots \geq h_n$ satisfying $\sum_{i=1}^{n} h_i = 1$. Each bamboo b_i starts at height 0 and grows at a steady rate of h_i every time unit. At the end of each time unit, the player chooses one of the bamboo and chops this bamboo down to height 0. The goal, of course, is to achieve the smallest possible backlog, which is the height of the tallest bamboo.

It is known that no bamboo trimming algorithm can guarantee a backlog less than 2, as it is possible to achieve backlog at least $2 - 2\varepsilon$ against any bamboo trimming algorithm with two bamboo that have fill rates $1 - \varepsilon$ and ε [11]. Recent work has yielded complex pinwheel algorithms [18] that achieve backlog 2, and are thus optimal in terms of the worst-case backlog; there has also been effort to extend the guarantees of these algorithms to achieve strong competitive ratios for cases where backlog less than 2 is possible [12, 33].

1.1 The quest for a simple optimal bamboo-trimming algorithm

The relative complicatedness of the known pinwheel algorithms has sparked a great deal of interest in the question as to whether there exists some *simple* algorithm that achieves the optimal backlog of 2. It would be especially interesting if **Reduce-Max** were to achieve backlog 2, since this would mean that the algorithm is optimal for both bamboo-trimming and the standard cup game. Currently, the best known bound for **Reduce-Max** is a backlog of 9 [11]. Experimental work [15] has found that **Reduce-Max** does, in fact, seem to achieve a backlog of 2, however, leading the authors to pose a backlog of 2 as a conjecture.

Another algorithm family that has been studied for its simplicity is **Reduce-Fastest(x)**. This algorithm trims down the fastest-growing bamboo out of those that have height at least x at the end of each time unit. Initial study proved that **Reduce-Fastest**(2) achieves backlog at most 4 [18], and further work demonstrated that **Reduce-Fastest(x)** achieves backlog at most

$$\max\left(x + \frac{x^2}{4(x-1)}, \frac{1}{2} + x + \frac{x^2}{4(x-1/2)}\right)$$

for all $x > 1$, which yields a bound of $19/6$ at $x = 2$ [11] (and is $\geq 1.25x$ for all $x > 1$). Extensive computer experimentation [15] suggests that this bound of $19/6$ is still not optimal, and has led researchers to conjecture that **Reduce-Fastest**(2) actually achieves a backlog of 3. Based on the same experiments, the authors further conjecture that **Reduce-Fastest**(1) achieves the optimal backlog of 2 [15]. However, as of now, no theoretical bounds on the backlog of **Reduce-Fastest**(1) are known.

1.2 Our Results

Our first result is an improved bound on the backlog of **Reduce-Max** for bamboo trimming. We prove that **Reduce-Max** achieves a backlog of 4, which narrows the gap between the upper and lower bounds from 7 to 2. At a technical level, our bound relies on a novel potential-function argument; we believe that this argument may be of independent interest as a tool that could help in future analyses of similar problems.

Our second set of results analyze **Reduce-Fastest(x)** for different values of x. We are able to prove that **Reduce-Fastest(x)** achieves backlog $x + 1$ for all $x \geq 2$, and we give a matching lower bound showing that this analysis is tight. This is the first time that a tight analysis has been achieved for **Reduce-Fastest(x)** for any value of x. For $x = 2$, the result gives a backlog of 3, which resolves a conjecture of [15]. On the other hand, we disprove the conjecture of [15] that **Reduce-Fastest**(1) achieves backlog 2. Instead, we show that **Reduce-Fastest**(1) allows for a backlog of $3 - \epsilon$ for any $\epsilon > 0$. More generally, we show that there is no x for which **Reduce-Fastest(x)** achieves a backlog of 2.01, meaning that **Reduce-Fastest(x)** is no longer a candidate in the quest for a simple optimal algorithm.

Our final result is a simple algorithm, which we call the ***Deadline-Driven Strategy***, that does in fact achieve a backlog of 2. The algorithm, which is based upon Liu and Layland's algorithm from the early 70s for a related scheduling problem [29], maintains the simplicity associated with **Reduce-Max** and **Reduce-Fastest(x)**, while also matching the backlog bounds of the more complicated pinwheel-based algorithms.

The **Deadline-Driven Strategy** selects the bamboo that will soonest achieve height 2 of the bamboo that have height at least 1. From a scheduling standpoint, we can consider the time at which a bamboo achieves height 2 to be its deadline. From this perspective, the **Deadline-Driven Strategy** is simply chopping down the bamboo with the closest deadline, while not considering very short bamboo with height less than 1. The **Deadline-Driven Strategy** shares an intriguing relationship with **Reduce-Max** and **Reduce-Fastest(x)**. The **Reduce-Max** strategy is concerned solely with the height of a bamboo, whereas the **Reduce-Fastest(x)** strategy is concerned solely with the speed of a bamboo. **Reduce-Max** can be thought of as cutting down the bamboo that is closest to achieving height 2 in terms of height (possibly selecting that bamboo that has furthest surpassed 2 if the conjecture that **Reduce-Max** achieves backlog 2 is false), and **Reduce-Fastest(x)** cuts down the quickest bamboo that has surpassed some threshold x. The **Deadline-Driven Strategy** strikes a balance between these two — instead of cutting down the bamboo that is closest to 2 in terms of distance or the bamboo with the maximum speed, it cuts down the bamboo that is closest to 2 in terms of time. It's interesting that of these three simple bamboo trimming algorithms, one is concerned with distance, one with speed, and the third with time.

1.3 The Relationship to the Multi-Processor Cup Game

We observe that many of our results in this paper extend to the multi-processor fixed-rate cup flushing game, which is the analogous multi-processor version of the bamboo garden trimming problem.

In each step of the multi-processor cup game with p processors, the filler places p units of water into the cups, with no more than 1 unit of water going to any cup. The emptier then removes water from each of p cups – the emptier corresponds to a p-processor machine. The multi-processor version of the bamboo garden trimming problem is the multi-processor fixed-rate cup flushing game, in which the player empties p cups entirely instead of removing only 1 unit of water from each of p cups.

As noted by [9], solutions to the single-processor version of the fixed-rate cup flushing game immediately yield solutions to the multi-processor version, since a time step in the multi-processor version can be modelled as a chunk of p steps of the single-processor game in which the fill rates are reduced by a factor of p. Thus, an upper bound of y on the backlog achieved by an algorithm in the single-processor version of the bamboo trimming problem immediately yields a corresponding algorithm that achieves backlog no more than $y + 1$ in the p-processor version of the bamboo trimming problem. (A gap of 1 is lost since cups emptied in the first step of a chunk of p steps in the single-processor fixed-rate cup flushing game will not be emptied until the end of the corresponding time step in the corresponding multi-processor game, resulting in a backlog as much as $(p - 1)/p$ units of water higher).

Thus we are able to show that the analogous version of **Reduce-Max** achieves backlog at most 5 in the multi-processor bamboo trimming game, that **Reduce-Fastest**(x) achieves backlog at most $x + 2$ for all $x \geq 2$, and that the **Deadline-Driven Strategy** achieves backlog at most 3 for the multi-processor version of the bamboo trimming problem.

2 REDUCE-MAX

In this section, we prove the following theorem.

THEOREM 2.1. *The **Reduce-Max** algorithm limits the backlog to strictly less than* 4.

Recall that we have n bamboo $b_1, \ldots, b_n$ with corresponding growth rates $h_1 \geq h_2 \geq \cdots \geq h_n$. We denote the height of b_i at time t, after the t-th cut, by $|b_i|_t$. For $i \in [n]$ and $t \in \mathbb{N} \cup \{0\}$, we define the ***volume function***

$$V(i, t) = \sum_{k=1}^{i} \min(2, |b_k|_t)$$

to be the function measuring the height at time t of the bamboo with growth rates at least h_i, counting tall bamboo as having height at most 2.

For $i \in [n]$ and $t \in \mathbb{N} \cup \{0\}$, we then define a potential function

$$\Phi(i, t) = \sum_{\substack{k \in [i] \\ 2(k-1) < V(i,t)}} h_k \cdot \min(2, V(i, t) - 2(k - 1)),$$

which is a weighted sum of $h_1, \ldots h_i$, where the multiplicative weights sum to $V(i, t)$ and are each at most 2. The weights are distributed

to maximize the sum by putting as much weight as possible on the earlier values of k. For example, if $V(i, t) = 7.25$, we would have

$$\Phi(i, t) = 2h_1 + 2h_2 + 2h_3 + 1.25h_4.$$

We prove the following lemma by examining the behavior of our potential function $\Phi(i, t)$ over time.

LEMMA 2.2. *For all $i \in [n]$ and $t \in \mathbb{N} \cup \{0\}$,*

$$|b_i|_t \leq 4 - \Phi(i, t) \leq 4.$$

PROOF. We proceed by induction on time t. For the base case we consider $t = 0$, in which case

$$|b_i|_t = 0 \leq 2 \leq 4 - \Phi(i, t)$$

for all $i \in [n]$. (Note that $0 \leq \Phi(i, t) \leq \sum_{k=1}^{n} 2h_k \leq 2$ by the definition of Φ.)

For the inductive step, we suppose that the theorem holds at t for all i. We will then prove that the theorem also holds at time $t + 1$ for all i.

Let $i \in [n]$. We know that

$$|b_k|_t \leq 4 - \Phi(k, t)$$

for all $k \in [n]$ by the inductive hypothesis. Between time t and $t + 1$ each b_k first grows by h_k, and then the tallest bamboo, some b_j, is cut down. We refer to the time between the bamboo growing and the tallest bamboo being cut down as the ***intermediate*** step. We assume that b_j has height at least 2 during the intermediate step between t and $t + 1$, as otherwise the lemma trivially holds for all bamboo at time $t + 1$ (none of the bamboo will even have height 2 at time $t + 1$). We will complete the proof with 3 cases.

Case 1: $j < i$. In this case, a quicker-growing bamboo was cut down in the stead of b_i.

We know that b_i has grown exactly h_i units from time t to time $t + 1$:

$$|b_i|_{t+1} = |b_i|_t + h_i. \tag{1}$$

We also know that the volume function satisfies $V(i, t + 1) \leq V(i, t) - 1$ because the growth step adds at most 1 unit of volume, and then a bamboo b_j with intermediate height at least 2 and with $j < i$ is cut down, which removes 2 units of volume. That is, the volume function with respect to i decreases by at least 1 unit from time t to $t + 1$. Thus

$$\Phi(i, t + 1) \leq \Phi(i, t) - h_i. \tag{2}$$

since this removed unit of volume would have been weighted by some growth-rate at least h_i in the weighted sum $\Phi(i, t + 1)$.

By the inductive hypothesis, we know that the lemma holds for time t, so we have

$$|b_i|_t \leq 4 - \Phi(i, t).$$

Substituting with Equations (1) and (2) we have

$$|b_i|_{t+1} - h_i \leq 4 - (\Phi(i, t + 1) + h_i)$$

and thus

$$|b_i|_{t+1} \leq 4 - \Phi(i, t + 1).$$

Case 2: $j = i$. In this case, we know b_i was just chopped down, so

$$|b_i|_{t+1} = 0 < 2$$
$$\leq 4 - \Phi(i, t+1).$$

Here we use the fact that Φ never exceeds 2 as

$$\Phi(i, t+1) \leq \sum_{k \in [n]} 2h_k \leq 2.$$

Case 3: $j > i$. In this case, a slower-growing bamboo, b_j, was cut down in the stead of b_i. We know that b_j, with height $|b_j|_t + h_j$, is the tallest bamboo during the intermediate step between t and $t+1$. So

$$|b_i|_{t+1} \leq |b_j|_t + h_j. \tag{3}$$

How does V change from $V(j, t)$ to $V(i, t+1)$? During the growth phase, $\sum_{k=1}^{i} h_k \leq 1 - h_j$ units of volume are added to the bamboo with indices $1, \ldots, i$. On the other hand, $V(j, t)$ includes at least $2 - h_j$ units of volume from bamboo b_j, which $V(i, t+1)$ does not include. Thus we have

$$V(i, t+1) \leq V(j, t) - 1. \tag{4}$$

Each unit of volume is weighted by at least h_j in both $\Phi(j, t)$ and $\Phi(i, t+1)$, so by Equation (4) we have

$$\Phi(i, t+1) \leq \Phi(j, t) - h_j. \tag{5}$$

We know by the inductive hypothesis that

$$|b_j|_t \leq 4 - \Phi(j, t).$$

Substituting by Equations (3) and (5) we have

$$(|b_i|_{t+1} - h_j) \leq 4 - (\Phi(i, t+1) + h_j)$$

and thus

$$|b_i|_{t+1} \leq 4 - \Phi(i, t+1).$$

$\square$

We conclude the section by proving Theorem 2.1.

PROOF OF THEOREM 2.1. It follows from Lemma 2.2 that no bamboo can achieve height 4 even during an intermediate step. Recall that bamboo i has height $|b_i|_t + h_i$ during the intermediate step after time t. If $|b_i|_t < 2$, then it follows immediately that $|b_i|_t + h_i < 3 < 4$. Otherwise, we know $V(i, t) \geq 2$, which implies that $\Phi(i, t) \geq 2h_1$. Thus we can apply Lemma 2.2 to find the bound

$$|b_i|_t + h_i \leq 4 - \Phi(i, t) + h_i$$
$$\leq 4 - 2h_1 + h_i$$
$$\leq 4 - h_1 < 4.$$

Note that we have, in fact, proved a slightly stronger claim than that of Theorem 2.1. Not only does **Reduce-Max** limit the backlog to 4, it actually limits the backlog to $4 - h_1$, i.e., 4 minus the largest growth rate among the bamboo. $\square$

3 REDUCE-FASTEST

Reduce-Fastest(x) is a bamboo trimming algorithm that cuts down the fastest-growing bamboo with height at least x at each time step (if no bamboo has height at least x, then no action is taken).

Reduce-Fastest(x) was first studied by Gąsieniec, Klasing, et al. in the case of $x = 2$ in [18]. They proved that **Reduce-Fastest(2)** achieves backlog 4. In [15], D'Emidio et al. performed an extensive experimental evaluation of several bamboo garden trimming algorithms including **Reduce-Fastest(1)** and **Reduce-Fastest(2)**. The authors conjectured that **Reduce-Fastest(1)** achieves backlog 2 and **Reduce-Fastest(2)** achieves backlog 3. We are able to disprove the first conjecture, and prove a more general form of the second conjecture: that **Reduce-Fastest(x)** limits the backlog to $x + 1$ for all $x \geq 2$, and that this bound is tight.

THEOREM 3.1. *For all $x \geq 2$, **Reduce-Fastest(x)** prevents any bamboo from achieving height $x + 1$. (And thus the backlog is strictly less than $x + 1$.)*

PROOF. Suppose for the sake of contradiction that we have n bamboo $b_1, \ldots, b_n$ with corresponding fill-rates $h_1, \ldots, h_n$, and that some bamboo b_i achieves height $x+1$ at time t_3 after most-recently achieving height x at time t_1. We then consider the bamboo that are cut at least once in $[t_1, t_3)$, and denote the set of such bamboo by S. For all $b_j \in S$, we denote by m_j the number of times that b_j is cut in the interval $[t_1, t_3)$.

The following claim shows that for all $b_j \in S$, $h_j \geq m_j \cdot h_i$. That is, for a bamboo to be cut down m times in the interval $[t_1, t_3)$, it must have fill-rate at least m times that of b_i.

CLAIM 1. *For all $b_j \in S$, we have $h_j \geq m_j \cdot h_i$.*

PROOF. We begin by considering the case of $m_j = 1$. In this case we have $h_j \geq h_i$, as h_j was cut down by **Reduce-Fastest(x)** at a time when b_i had height at least x, so b_j must be at least as fast-growing as b_i.

Next we consider the case of $m_j \geq 2$. In this case we know b_j is cut down m_j times in the interval $[t_1, t_3)$, so it must grow at least $x(m_j - 1)$ in that interval as b_j has to regrow to a height of at least x units between successive cuts. However, b_i fails to grow even 1 unit during the same interval $[t_1, t_3)$, as it has height at least x at time t_1, and it has not yet achieved height $x + 1$ at time $t_3 - 1$. In other words, in the time that bamboo b_j grows by at least $x(m_j-1)$, bamboo b_i grows by less than 1. Thus

$$h_j \geq x(m_j - 1)h_i$$
$$\geq 2(m_j - 1)h_i$$
$$\geq m_j h_i,$$

where the final inequality uses $m_j \geq 2$. $\square$

We now consider the length of the interval $[t_3, t_1)$. We have by Claim 1 that

$$t_3 - t_1 = \sum_{b_j \in S} m_j$$

$$\leq \sum_{b_j \in S} \frac{h_j}{h_i}$$

$$\leq \frac{1}{h_i} \sum_{b_j \in S} h_j$$

$$\leq \frac{1}{h_i}(1 - h_i)$$

$$= 1/h_i - 1.$$

Moreover $t_3 - t_1$ is integer, so

$$t_3 - t_1 \leq \lfloor 1/h_i - 1 \rfloor$$

$$\leq \lfloor 1/h_i \rfloor - 1.$$

But this means that the interval is too short for b_i to reach height $x+1$. In particular, b_i requires at least $\lfloor 1/h_i \rfloor$ time to achieve height $x + 1$ after achieving height x. To be explicit, we have that at time t_1 bamboo b_i has height strictly less than $x + h_i$, and thus that at time t_3 bamboo b_i has height strictly less than

$$x + h_i + (t_3 - t_1)h_i \leq x + h_i(1 + \lfloor 1/h_i \rfloor - 1)$$

$$\leq x + 1.$$

Therefore b_i does not achieve height $x + 1$ at time t_3, which is a contradiction.

□

The bound of $x + 1$ for all $x \geq 2$ on the backlog guaranteed by **Reduce-Fastest**(x) is tight. We believe this is the first tight bound on **Reduce-Fastest**(x) for any value of x.

THEOREM 3.2. *For any $\varepsilon, x > 0$, there exists some $n \in \mathbb{N}$ such that* ***Reduce-Fastest****(x) allows for backlog at least $x + 1 - \varepsilon$.*

PROOF. Consider n bamboo with uniform fill rates $h_i = 1/n$ for all i. No bamboo will be cut down until they all simultaneously achieve height at least x. Then over the next n time steps, all of the bamboo will be cut down, with the last bamboo reaching height at least $x + (n - 1)/n$.

Setting $n = \lceil 1/\varepsilon \rceil$, we obtain a backlog at least

$$x + 1 - 1/n \geq x + 1 - \varepsilon.$$

□

We conclude this section by providing lower bounds on the backlog achieved by **Reduce-Fastest**(x). In particular, we give a counterexample to the conjecture that **Reduce-Fastest**(1) achieves backlog 2. Interestingly, it remains an open problem as to whether **Reduce-Fastest**(1) achieves any finite backlog.

THEOREM 3.3. ***Reduce-Fastest****(1) does not achieve any backlog less than 3.*

PROOF. Suppose we have f fast bamboo with growth rates

$$1/(f + \sqrt{f})$$

and $s = \sqrt{f} + 1$ fast bamboo with growth rates

$$1/(f + 2\sqrt{f} + 2)$$

for some $f \in \mathbb{N}$ that is a perfect square.

We note that in this construction,

$$\sum_i h_i = f(f + \sqrt{f}) + (\sqrt{f} + 1)/(f + 2\sqrt{f} + 2)$$

$$< \sqrt{f}(\sqrt{f} + 1) + 1/(\sqrt{f} + 1)$$

$$= 1.$$

Thus the sum of the fill rates is less than 1, and so this is a valid construction of bamboo.

Now we examine the behavior these bamboo exhibit when the cutting player utilizes the **Reduce-Fastest**(1) strategy. Initially, all bamboo have height less than 1, so the player does not cut any bamboo down. At time $f + \sqrt{f}$, the fast bamboo all simultaneously achieve height 1. Thus at time steps

$$f + \sqrt{f}, \ldots, 2f + \sqrt{f} - 1,$$

the f fast bamboo are cut down. Then all of the fast bamboo have height less than 1, and the slow bamboo have all achieved height 1. Thus **Reduce-Fastest**(1) will cut down the slow bamboo until a fast bamboo again achieves height 1. Therefore during each of the time steps

$$2f + \sqrt{f}, \ldots, 2f + 2\sqrt{f} - 1$$

a slow bamboo will be cut down. During those time steps, $\sqrt{f} = s - 1$ slow bamboo are cut down, meaning exactly 1 slow bamboo has not yet been cut by time $2f + 2\sqrt{f}$. The **Reduce-Fastest**(1) algorithm does not have time to cut this last slow bamboo, as at time $2f + 2\sqrt{f}$ the first of the fast bamboos that was cut again achieves height 1. Since **Reduce-Fastest** prioritizes fast bamboo, it will then cut down each of the fast bamboo during time steps

$$2f + 2\sqrt{f}, \ldots, 3f + 2\sqrt{f} - 1$$

as they successively achieve height 1. Thus the final remaining uncut slow bamboo will not be cut for the first time until at least time $3f + 2\sqrt{f}$, by which time it has achieved height

$$(3f + 2\sqrt{f})/(f + 2\sqrt{f} + 2)$$

$$= 3 - (4\sqrt{f} + 6)/(f + 2\sqrt{f} + 2)$$

$$= 3 - o(1).$$

Thus this construction of bamboo achieves backlog arbitrarily close to 3 as $f \to \infty$.

□

The following theorem, while rather simple, serves to show that **Reduce-Fastest**(x) cannot achieve worst-case backlog arbitrarily close to the optimal value of 2 for any value of x.

THEOREM 3.4. *There is no value of x for which* ***Reduce-Fastest****(x) achieves backlog 2.01.*

PROOF. By theorem 3.2, we know that this holds for any value of $x > 1.01$. Also note for $0 < x < 1$, a simple construction with one bamboo of growth rate x and another of growth $1 - x$ achieves infinite backlog, as the slower of the two bamboos is never cut down. (And for $x \leq 0$, any construction with multiple bamboo with distinct growth rates achieves unbounded backlog).

We now offer a simple, concrete construction that holds for any value of $1 \le x \le 1.01$. Suppose we have 900 bamboo with growth rates $1/1000$ and 140 bamboo with growth rates $1/1400$. The sum of the growth rates is $\sum_i h_i = 1$. This construction is very similar to the construction of fast and slow bamboo offered in the preceding theorem, but it is loose enough to continue to offer a bound on the backlog of at least 2.01 for all $x < 1.01$.

The fast bamboo achieve height x somewhere between time 1000 and time 1010 depending on x. Then each of the 900 fast bamboo are cut down, and then some proper subset of the slow bamboo are cut down until the first cut of the fast bamboo again achieves height x at some time no later than 2020. Then the fast bamboo are again all cut. By the time that all of the fast bamboo have been cut exactly twice, at least 2900 time steps have elapsed, and some slow bamboo remains uncut with height at least $2900/1400 > 2.01$. □

Thus, we can eliminate **Reduce-Fastest**(x) from consideration in the search for a simple, optimal bamboo-cutting algorithm.

4 DEADLINE-DRIVEN STRATEGY

We now present a very simple algorithm, the **Deadline-Driven Strategy**, which achieves backlog 2. The algorithm, which is novel in the context of bamboo trimming, was introduced by Liu and Layland [29] in the context of a related scheduling problem in the early 70's.

We begin by translating the result of Liu and Layland [29] to be in terms of the fixed-rate cup game, which we remind the reader is defined as follows. There are n cups with fill rates $h_1, \ldots, h_n$ satisfying $\sum_{i=1}^{n} h_i \le 1$. At the beginning of each time step, each cup i receives h_i units of water. The player then selects a cup from which to remove 1 unit of water — if the cup contains less than 1 unit of water, it is emptied. The backlog for the cup game is defined as the height of the fullest cup.

The Deadline Driven Strategy examines all cups with height at least 1, and removes water from the cup that will soonest reach height 2 — it arbitrarily chooses from the cups with the closest deadline of reaching height 2.

One interpretation of Theorem 7 from Liu and Layland's paper [29] is that the **Deadline-Driven Strategy** achieves backlog 2 for the fixed-rate cup game (i.e., the non-flushing bamboo game) as long as the fill-rates are inverse integers — each h_i is equal to $1/k_i$ for some $k_i \in \mathbb{N}$. Subsequent work [26] rediscovered the **Deadline-Driven Strategy** for a related scheduling problem; one consequence of their arguments is that, *if backlog 2 is possible*, then the **Deadline-Driven Strategy** achieves it. Since [6] established that backlog 2 is, in fact possible (using results from network flow theory), it follows that one can remove the inverse-integer constraint on Liu and Layland's result [29]. That is, the **Deadline-Driven Strategy** achieves backlog 2 in the fixed-rate cup game for any set of fill rates.

We now give an alternative analysis of the **Deadline-Driven Strategy** that applies to both the fixed-rate cup game and the bamboo trimming problem—our analysis is significantly simpler than those in past work.

THEOREM 4.1. *Suppose that we have n cups $b_1, \ldots, b_n$ with corresponding fill-rates $h_1, \ldots, h_n$ satisfying $\sum_{i=1}^{n} h_i \le 1$. Then the* **Deadline-Driven Strategy** *for the cup game will limit the backlog to strictly less than 2. Furthermore, the equivalent strategy will also limit the backlog to strictly less than 2 if applied to n bamboo with fill-rates $h_1, \ldots, h_n$.*

PROOF. Following the terminology from Liu and Layland's paper, we say that cup b_i is **requested** at time t if it reaches height 1 at time t. At a time t_0, we say that a cup b_i has a **deadline** at time t if $|b_i|_{t_0} \ge 1$ and $|b_i|_{t_0} + (t - t_0)h_i \in [2, 2 + h_i)$, that is, the cup has height at least 1 and it will achieve height 2 at time t if no water is removed during the interval $[t_0, t)$. We say that cup b_i **overflows** at time t if it achieves height 2 at time t, i.e., the cup is not attended to before its deadline. Whenever a cup with fill in the range $[1, 2)$ is emptied from, we say that the request (when the cup previously reached height 1) is **completed**.

Suppose for the sake of contradiction that cup i overflows at time t_3 and that this is the first ever overflow. We then define t_1 to be the last time prior to t_3 during which either the player is idle (as all cups have height strictly less than 1) or the player chooses a cup with deadline after t_3.

We consider the time interval $(t_1, t_3) = [t_1 + 1, t_3 - 1]$, during which we know the player is busy removing water from cups with deadlines at or before t_3. Furthermore, at time t_1, the player was either idle or was busy removing water from a cup with deadline after t_3. Thus all cups with request time $\le t_1$ and deadline $\le t_3$ had already had their requests completed by time t_1. Thus in the interval (t_1, t_3), the player is continuously working on tasks with request time after t_1 and deadline at or before t_3.

Now we count the number of requests that occur at or after $t_1 + 1$ with deadline at or before t_3. We call such requests and deadlines **critical**.

Case 1: There are at least $t_3 - t_1$ critical deadlines. Let $\varepsilon_j = 1 - |b_j|_{t_1}$ for each cup j. Since every request at or before t_1 with a deadline in $(t_1, t_3]$ is completed by time t_1, we know that any cup at time t_1 that has fill 1 or larger must have a deadline after t_3 and must not contribute any critical requests/deadlines. Thus each cup j that has at least one critical deadline satisfies $\varepsilon_j > 0$. It follows that if a cup j has $r > 0$ critical deadlines, then the total amount of water placed into cup j during times $(t_1, t_3]$ is at least $r + \epsilon_j > r$. Since at least $t_3 - t_1$ total critical deadlines occur in the interval $(t_1, t_3]$, it follows that more than $t_3 - t_1$ water is placed into cups during those $t_3 - t_1$ steps, a contradiction.

Case 2: There are at most $t_3 - t_1 - 1$ critical deadlines. Since the player is non-idle during the interval (t_1, t_3), and since t_3 is the first step during which any cup overflows, the player completes $t_3 - t_1 - 1$ critical requests, one during each of the steps $t_1 + 1, \ldots, t_3 - 1$. Additionally, the final deadline for cup i (which overflows at time t_3) is not met and thus corresponds to a critical request that is not completed. Hence there are at least $t_3 - t_1 - 1$ critical requests that get completed during the interval $(t_1, t_3]$ and at least 1 critical request that does not get completed. This contradicts the assumption that there are $t_3 - t_1 - 1$ or fewer critical requests/deadlines.

Precisely the same analysis that we have used to prove the theorem for the fixed-rate cup game also applies to the bamboo-garden trimming problem. (Indeed, the bamboo-garden trimming problem can be modelled as a version of the fixed-rate cup game in which the player empties cups instead of only removing 1 unit of water.) □

Thus, the **Deadline-Driven Strategy** is a simple algorithm which achieves the optimal worst-case backlog of 2.

ACKNOWLEDGMENTS

REFERENCES

[1] M. Adler, P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. Goldberg, and M. 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), 2003, pp. 101–108.

[2] A. Amir, M. Farach, R. M. Idury, J. A. L. Poutré, and A. A. Schäffer, *Improved dynamic dictionary matching*, Inf. Comput., 119 (1995), pp. 258–282.

[3] A. Amir, G. Franceschini, R. Grossi, T. Kopelowitz, M. Lewenstein, and N. Lewenstein, *Managing unbounded-length keys in comparison-driven data structures with applications to online indexing*, SIAM J. Comput., 43 (2014), pp. 1396–1416.

[4] Y. Azar and A. Litichevskey, *Maximizing throughput in multi-queue switches*, Algorithmica, 45 (2006), pp. 69–90.

[5] A. Bar-Noy, A. Nisgav, and B. Patt-Shamir, *Nearly optimal perfectly periodic schedules*, Distributed Comput., 15 (2002), pp. 207–220.

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

[7] S. K. Baruah, J. Gehrke, and C. G. Plaxton, *Fast scheduling of periodic tasks on multiple resources*, in Proceedings of IPPS '95, The 9th International Parallel Processing Symposium, April 25-28, 1995, Santa Barbara, California, USA, IEEE Computer Society, 1995, pp. 280–288.

[8] M. A. Bender, R. Das, M. Farach-Colton, R. Johnson, and W. Kuszmaul, *Flushing without cascades*, in Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020, S. Chawla, ed., SIAM, 2020, pp. 650–669.

[9] M. A. Bender, M. Farach-Colton, and W. Kuszmaul, *Achieving optimal backlog in multi-processor cup games*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019, M. Charikar and E. Cohen, eds., ACM, 2019, pp. 1148–1157.

[10] M. A. Bender and W. Kuszmaul, *Randomized cup game algorithms against strong adversaries*, in Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021, D. Marx, ed., SIAM, 2021, pp. 2059–2077.

[11] D. Bilò, L. Gualà, S. Leucci, G. Proietti, and G. Scornavacca, *Cutting bamboo down to size*, Theoretical Computer Science, (2022).

[12] F. Della Croce, *An enhanced pinwheel algorithm for the bamboo garden trimming problem*, arXiv:2003.12460, (2020).

[13] P. Dietz and D. Sleator, *Two algorithms for maintaining order in a list*, in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC), 1987, pp. 365–372.

[14] P. F. Dietz and R. Raman, *Persistence, amortization and randomization*, in Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1991, pp. 78–88.

[15] M. D'Emidio, G. Di Stefano, and A. Navarra, *Bamboo garden trimming problem: Priority schedulings*, Algorithms, 12 (2019), p. 74.

[16] J. Fischer and P. Gawrychowski, *Alphabet-dependent string searching with wexponential search trees*, in Combinatorial Pattern Matching - 26th Annual Symposium, CPM 2015, Ischia Island, Italy, June 29 - July 1, 2015, Proceedings, F. Cicalese, E. Porat, and U. Vaccaro, eds., vol. 9133 of Lecture Notes in Computer Science, Springer, 2015, pp. 160–171.

[17] H. R. Gail, G. A. Grover, R. Guérin, S. L. Hantler, Z. Rosberg, and M. Sidi, *Buffer size requirements under longest queue first*, Perform. Evaluation, 18 (1993), pp. 133–140.

[18] L. Gąsieniec, R. Klasing, C. Levcopoulos, A. Lingas, J. Min, and T. 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, Springer, 2017, pp. 229–240.

[19] M. H. Goldwasser, *A survey of buffer management policies for packet switches*, SIGACT News, 41 (2010), pp. 100–128.

[20] M. T. Goodrich and P. Pszona, *Streamed graph drawing and the file maintenance problem*, in Graph Drawing - 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers, S. K. Wismath and A. Wolff, eds., vol. 8242 of Lecture Notes in Computer Science, Springer, 2013, pp. 256–267.

[21] N. Guan and W. Yi, *Fixed-priority multiprocessor scheduling: Critical instant, response time and utilization bound*, in 26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum, IPDPS 2012, Shanghai, China, May 21-25, 2012, IEEE Computer Society, 2012, pp. 2470–2473.

[22] T. Kopelowitz, *On-line indexing for general alphabets via predecessor queries on subsets of an ordered list*, in 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012, IEEE Computer Society, 2012, pp. 283–292.

[23] W. Kuszmaul, *Achieving optimal backlog in the vanilla multi-processor cup game*, in Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020, S. Chawla, ed., SIAM, 2020, pp. 1558–1577.

[24] W. 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, 2021, pp. 1248–1261.

[25] A. Litman and S. Moran-Schein, *On distributed smooth scheduling*, in SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures, July 18-20, 2005, Las Vegas, Nevada, USA, P. B. Gibbons and P. G. Spirakis, eds., ACM, 2005, pp. 76–85.

[26] A. Litman and S. Moran-Schein, *Smooth scheduling under variable rates or the analog-digital confinement game*, Theor. Comp. Sys., 45 (2009), pp. 325–354.

[27] A. Litman and S. Moran-Schein, *On centralized smooth scheduling*, Algorithmica, 60 (2011), pp. 464–480.

[28] C. L. Liu, *Scheduling algorithms for multiprocessors in a hard real-time environment*, JPL Space Programs Summary, 1969, (1969).

[29] C. L. Liu and J. W. Layland, *Scheduling algorithms for multiprogramming in a hard-real-time environment*, Journal of the ACM (JACM), 20 (1973), pp. 46–61.

[30] M. Moir and S. Ramamurthy, *Pfair scheduling of fixed and migrating periodic tasks on multiple resources*, in Proceedings of the 20th IEEE Real-Time Systems Symposium, Phoenix, AZ, USA, December 1-3, 1999, IEEE Computer Society, 1999, pp. 294–303.

[31] C. W. 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, January 12-14, 2003, Baltimore, Maryland, USA, ACM/SIAM, 2003, pp. 618–627.

[32] M. Rosenblum, M. X. Goemans, and V. Tarokh, *Universal bounds on buffer size for packetizing fluid policies in input queued, crossbar switches*, in Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004, IEEE, 2004, pp. 1126–1134.

[33] M. van Ee, *A 12/7-approximation algorithm for the discrete bamboo garden trimming problem*, Operations Research Letters, 49 (2021), pp. 645–649.