

Article

Bamboo Garden Trimming Problem: Priority Schedulings [†]

Mattia D'Emidio ^{1,*} , Gabriele Di Stefano ¹ and Alfredo Navarra ² 

¹ Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila; 67100 L'Aquila, Italy; gabriele.distefano@univaq.it

² Department of Mathematics and Computer Science, University of Perugia, 06123 Perugia, Italy; alfredo.navarra@unipg.it

* Correspondence: mattia.demidio@univaq.it; Tel.: +39-0862-43-4466

† This paper is an extended version of our paper published in the 45th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Nový Smokovec, Slovakia, 27–30 January 2019.

Received: 18 January 2019; Accepted: 11 April 2019; Published: 13 April 2019



Abstract: The paper deals with the *Bamboo Garden Trimming* (BGT) problem introduced in [Gąsieniec et al., SOFSEM'17]. The problem is difficult to solved due to its close relationship to *Pinwheel scheduling*. The garden with n bamboos is an analogue of a system of n machines that have to be attended (e.g., serviced) with different frequencies. During each day, bamboo b_i grows an extra height h_i , for $i = 1, \dots, n$ and, on the conclusion of the day, at most one bamboo has its entire height cut. The goal is to design a perpetual schedule of cuts to keep the height of the tallest ever bamboo as low as possible. The contribution in this paper is twofold, and is both theoretical and experimental. In particular, the focus is on understanding what has been called *priority schedulings*, i.e., cutting strategies where priority is given to bamboos whose current height is above a threshold greater than or equal to $H = \sum_{i=1}^n h_i$. Value H represents the total daily growth of the system and it is known that one cannot keep bamboos in the garden below this threshold indefinitely. As the first result, it is proved that, for any distribution of integer growth rates h_1, \dots, h_n and any priority scheduling, the system stabilises in a fixed cycle of cuts. Then, the focus is on the so-called ReduceMax strategy, a greedy priority scheduling that each day cuts the tallest bamboo, regardless of the growth rates distribution. ReduceMax is known to provide a $O(\log n)$ -approximation, with respect to the lower bound H . One of the main results achieved is that, if ReduceMax stabilises in a round-robin type cycle, then it guarantees 2-approximation. Furthermore, preliminary results are provided relating the structure of the input instance, in terms of growth rates, and the behavior of ReduceMax when applied to such inputs. Finally, a conjecture that ReduceMax is 2-approximating for the BGT problem is claimed, hence an extended experimental evaluation was conducted to support the conjecture and to compare ReduceMax with other relevant scheduling algorithms. The obtained results show that ReduceMax: (i) provides 2-approximation in all considered inputs; and (ii) always outperforms other considered strategies, even those for which better worst case approximation guarantees have been proven.

Keywords: bamboo garden trimming; periodic scheduling; approximation algorithms; perpetual testing; experimental evaluation

1. Introduction

This paper deals with a perpetual scheduling problem in which n machines, denoted later as *bamboos*, need to be attended with possibly *known* and likely different frequencies. In other words, some machines may have to be attended more often than others. This problem was proposed and

studied by [1] under the name of *Bamboo Garden Trimming* (BGT) problem. Given is a collection (garden) of n bamboos b_1, b_2, \dots, b_n along with respective daily growth rates h_1, h_2, \dots, h_n greater than 0. The authors of [1] assumed that, initially, the height of each bamboo is set to zero, whereas in this paper bamboos are allowed to start with arbitrary heights.

The robotic gardener maintaining the garden trims one bamboo per day to height zero according to some predefined schedule. The main goal in BGT is to design perpetual schedules of cuts which keep the height of the tallest ever bamboo as low as possible. The main constraints are that the gardener is allowed to cut exactly one (of their choice) bamboo at the end of each day and is not allowed to attend the garden at any other times. Once the gardener has decided which bamboo to trim at the end of the current day (which corresponds to one round in the schedule), the action of actual trimming is instantaneous. Different variants of the problem have been defined [1], but, in its “easiest” representation, all data are available to the gardener, i.e., one knows how many bamboos form the garden and for each bamboo what is the daily growth rate. Hence, one may think about trimming strategies that consider all the data, or simply choosing the bamboo to be trimmed according with the current status of the garden.

The problem reveals a combinatorial beauty, also related to the easiness of its definition that rapidly allows to start thinking about trimming strategies. As a first thought, it usually appears surprising that that the problem comes out to be of difficult resolution. This further motivates the design of different and efficient trimming strategies, varying on the usage of the available data. Moreover, the interest to this problem comes from its close relation with fundamental scheduling problems, e.g., *Pinwheel* scheduling.

The Pinwheel problem [2] is defined as follows. Given a set $V = f_1, f_2, \dots, f_n$ of positive integers called *Pinwheel frequencies*, one is asked to create an infinite sequence S of indices drawn from the set $1, 2, \dots, n$, such that, any sub-sequence of $f_i \in V$ consecutive elements in S includes at least one index i . The density of set V is defined as $D = \sum_{i=1}^n \frac{1}{f_i}$ and it is known that instances having density $D > 1$ cannot be scheduled. An example of input instance for the Pinwheel problem is shown in Figure 1.

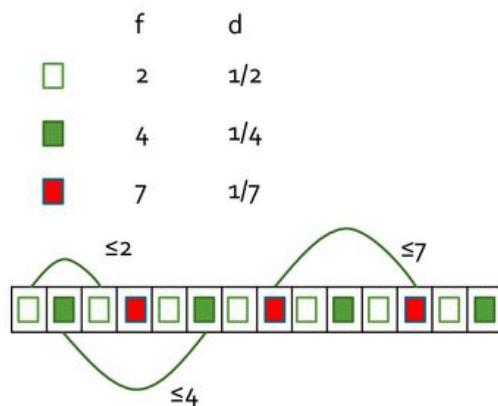


Figure 1. An example of input instance for the Pinwheel problem where $V = (2, 4, 7)$ and hence $D = 1/2 + 1/4 + 1/7 \approx 0.89$.

In [2], it has been shown that Pinwheel is NP-hard assuming succinct representation of the problem, whereas it can be efficiently solved when the Pinwheel frequencies are powers of 2 and the density is at most 1 [1,3]. In [1], a 2-approximation algorithm for BGT, based on the resolution of the Pinwheel problem in the special case where frequencies are powers of 2 and the density is at most 1, has been introduced. In more details, given an instance of BGT with rates $\{h_1, h_2, \dots, h_n\}$, one can create a parametrical instance $I(\delta)$ of Pinwheel with frequencies

$$f_i = \left\lfloor (1 + \delta) \frac{H}{h_i} \right\rfloor, \text{ for } i = 1, 2, \dots, n \text{ and for any } \delta > 0$$

and prove that a (feasible) solution to $I(\delta)$ implies a $(1 + \delta)$ -approximation schedule for the original BGT instance. In particular, one can set $\delta = 1$ and round frequencies to closest power of two to obtain an instance of Pinwheel having power of two frequencies and density $D \leq 1$, that can be scheduled efficiently by a greedy algorithm. We refer the interested reader to [1,3] for more details on the proofs.

Furthermore, a 4-approximation and an $O(\log n)$ -approximation algorithms have been provided, in the case the gardener exploits only partial knowledge of the input instance. In detail, to achieve the 4-approximation, it is sufficient to know for any couple of bamboos which one is associated with the highest growth rate, without knowing the exact rates. To achieve the $O(\log n)$ -approximation, instead, no knowledge about growth rates is assumed. Here, the interest is more related to this last setting, with the aim to improve the approximation ratio to a constant.

1.1. Related Work

The BGT problem, while of independent combinatorial interest, originates from perpetual testing of virtual machines in cloud systems [4]. In such systems, the frequency in which virtual machines are tested for undesirable symptoms varies depending on the importance of dedicated cloud operational mechanisms. The BGT problem considered here is also a close relative of several classical algorithmic problems which focus on *monitoring* including *Graph Exploration* [5–8], *Art Gallery Problem* [9] and its dynamic extension *k-Watchmen Problem* [10]. In the work on *Patrolling* [11,12], the studies focus on monitoring a set of points with the same frequency of attendance, whereas in [13], the frequency may vary.

In this paper, similarly to [1], the focus is on the case where each bamboo has its own attendance factor, which makes it related to *periodic scheduling* [14], several variants of *Pinwheel* related problems [2,15–17] including *periodic Pinwheel* problem [18,19] and *Pinwheel scheduling* [20], as well as the concept of *P-fairness* in sharing multiple copies of some resource among various tasks [21,22]. It is worth pointing out here that the intractability of BGT results from the hardness of Pinwheel scheduling proved in [23].

Of course, the relevance of a new variant of scheduling is of main interest to the algorithmic community as scheduling issues represent fundamental primitives in many contexts. To this respect, it is worth citinh the close relation of the proposed BGT problem with the work done on similar settings in [24,25]. The main difference basically resides in the objective function. In fact, in [24,25], the aim is to minimise the long-run average height of the bamboos, whereas in BGT the aim concerns the minimisation of the maximum height ever reached by the bamboos.

In related research on minimising the maximum occupancy of a buffer in a system of n buffers, the usual setting is a game between the player and the adversary [26–28]. The adversary decides how a fixed total increase of data in each round is distributed among the buffers and tries to maximise the maximum occupancy of a buffer. The player decides which buffer (or buffers) should be emptied next and tries to minimise the maximum buffer size. The upper bounds developed in this more general context can be translated into upper bounds for the BGT problem. However, here the aim is to derive tighter bounds for the case where the knowledge of growth rates is not exploited, hence adopting strategies able to deal with input data being either partially or entirely unknown.

Probably the most natural algorithm to keep the elevation of the bamboo garden low is the greedy approach of always cutting the currently tallest bamboo. This method, which is agnostic with respect to growth rates, was first coined in [4] under the name of ReduceMax and further studied in [1]. This strategy was also considered independently (and under a different name) in the adversarial setting of buffer minimisation problems [27]. Another method studied in [1] is the ReduceFastest₂ approach, in which the fastest growing bamboo is cut but only among those having height above the threshold $2H$, where

$$H = \sum_{i=1}^n h_i.$$

Value H represents the total daily growth of the system of bamboos and it is known [1] that one cannot keep bamboos below this threshold level. From [1], it is known that ReduceFastest₂

provides a constant approximation to BGT, i.e., none of the bamboos grows above height $4H$. However, its applicability depends on the knowledge of value H and at least an ordering of the bamboos with respect to their growth rates. More interestingly, `ReduceMax` does not require such a knowledge but surprisingly it is not well understood. While there are insights that it should perform better than `ReduceFastest2`, the only upper bound known for the maximum height of bamboos is $O(H \cdot \log n)$ [1].

1.2. Achieved Results

In this paper, the main contribution is twofold, and it refers to both theoretical and experimental studies on the BGT problem. In particular, the focus is on better understanding of what are called *priority schedulings* that operate on any “reasonable”, i.e., involving all bamboos, strategy of cuts in which priority is given to bamboos with the current height above a threshold greater than or equal to H . Both `ReduceMax` and `ReduceFastest2` fall into the priority category. Moreover, one further requirement is that scheduling strategies must be fully deterministic. Hence, in the case of ties, i.e., when two or more bamboos are eligible to be cut with respect to the considered strategy, the bamboo with the biggest index is selected.

As the first result, it is proved that, for any distribution of integer growth rates h_1, \dots, h_n , and any priority scheduling, the system stabilises in a fixed cycle of cuts, eventually. However, the time needed to converge depends on the initial heights of bamboos. Then, it is shown that, whenever `ReduceMax` stabilises in a round-robin type cycle, it guarantees 2-approximation for BGT. Finally, extended experiments are conducted to compare `ReduceMax` with other strategies, including `ReduceFastest2`. Note that the work in [4] contains some experiments on `ReduceMax`, focusing only on selected distributions of bamboo growth rates. In contrast, here the focus is on all possible distributions of bounded size. The experiments show `ReduceMax` being 2-approximating in all considered instances, and provide evidence that `ReduceMax` outperforms all other tested strategies. Consequently, we conjecture the following.

Conjecture 1. *Algorithm `ReduceMax` is 2-approximating for the BGT problem.*

Note that, the results presented in this paper have appeared in a preliminary form in [29].

2. Notation

Given is a collection $\{b_1, b_2, \dots, b_n\}$ of n bamboos (i.e., a *garden*) along with respective daily growth rates h_1, h_2, \dots, h_n . It is assumed that each h_i is a positive integer, for any $1 \leq i \leq n$, and call a *configuration* C^t the sequence $(\ell_1^t, \ell_2^t, \dots, \ell_n^t)$ of the bamboos' heights at a given day t . Any configuration $C^t = (\ell_1^t, \ell_2^t, \dots, \ell_n^t)$ is determined by a *growth mechanism* applied on its predecessor $C^{t-1} = (\ell_1^{t-1}, \ell_2^{t-1}, \dots, \ell_n^{t-1})$, i.e., for any $1 \leq i \leq n$, it holds $\ell_i^t = \ell_i^{t-1} + h_i$. The only exception to this behavior is what is called a *trimming* operation. In particular, a bamboo b_i is said to be *trimmed* (equivalently *cut*) at a given day t if the height ℓ_i^t is reduced to zero and hence, trivially, $\ell_i^{t+1} = h_i$. For the sake of simplicity, in what follows the term t is omitted from all notations when the number of the day is clear from the context. Finally, given a configuration C , let $V(C)$ denote the *volume* of configuration C , which is the sum of all bamboos' heights in C , i.e.,

$$V(C) = \sum_{i=1}^n \ell_i.$$

An input instance I to the BGT problem is a set of growth rates $\{h_i\}_{1 \leq i \leq n}$, complemented by the initial configuration C^0 . The interest is in the design of perpetual schedules of cuts, which allow keeping the tallest bamboo in the garden as low as possible. In particular, it is assumed that at most one trimming operation can take place every day. Therefore, the aim at designing an algorithm \mathcal{A} that, for a given input instance I , computes a *perpetual schedule* $\mathcal{A}(I) = (i_1, i_2, \dots)$, i.e., a sequence of indices

$i_j \in \{0, 1, 2, \dots, n\}$ that determines, for any day $t > 0$, the bamboo to be trimmed, unless $i_j = 0$ when no bamboo is cut. In other words, a schedule of this kind defines an ordered sequence of trimming operations on the bamboos. In what follows, an algorithm determining perpetual schedules is called a *perpetual scheduling*, or simply *scheduling* algorithm.

Given an input instance I and a perpetual scheduling \mathcal{S} , an *execution* $E(I, \mathcal{S})$ is the sequence (C^0, C^1, \dots) obtained by applying the schedule computed by \mathcal{S} on C^0 . Moreover, given an execution E and a configuration C , let $M(E)$ ($M(C)$, respectively) denote the maximum height reached by a bamboo in E (C , respectively). Finally, let $H = \sum_{i=1}^n h_i$ denote the sum of the growth rates. It is known that no algorithm can compute a schedule that keeps the heights of all the bamboos below H indefinitely (i.e., such that $M(E) < H$) [1].

3. Theoretical Results

In this section, first a formal definition of priority scheduling along with the analysis of their performance (see Section 3.1) are introduced. Then, the focus is on the strategy ReduceMax and on showing that it provides 2-approximation for BGT under specific assumptions (see Section 3.2).

3.1. Priority Schedulings

Let \mathcal{C} be the set of any configuration of n bamboos.

Definition 1. An oblivious scheduling $\sigma : \mathcal{C} \rightarrow \{0, 1, \dots, n\}$ is a function which for any configuration of heights in \mathcal{C} returns an index i of the bamboo to be cut, and $i = 0$ means that none of the bamboos is scheduled to be cut.

In other words, in oblivious scheduling the next cut is solely based on the current configuration, without exploiting any knowledge about past cuts.

Definition 2. A configuration $C = (\ell_1, \ell_2, \dots, \ell_n)$ is said to be ordered whenever $i < j$ implies $h_i \geq h_j$.

The above implies the order of the sequence in C reflects a non-increasing ordering of the growth rates, i.e., ℓ_1 is the height of the bamboo with the biggest growth rate.

Definition 3. An ordered oblivious scheduling $\sigma : \mathcal{O} \rightarrow \{0, 1, \dots, n\}$ is an oblivious scheduling where $\mathcal{O} \subset \mathcal{C}$ is the set of ordered configurations in \mathcal{C} .

Definition 4. Given a threshold $\tau \geq H$ and any (ordered) configuration $C \in \mathcal{C}$, let L be the set of indices of all bamboos whose height is strictly greater than τ . An oblivious scheduling σ_τ is a (ordered) τ -priority scheduling if and only if $L \neq \emptyset$ implies $\sigma_\tau(C) \in L$.

In the remainder of the paper, a (ordered) τ -priority scheduling is simply referred to as a *priority scheduling* when the ordering and the value of τ are either clear from the context or irrelevant. Clearly, ReduceFastest₂ is an ordered $2H$ -priority scheduling, as it cuts only bamboos above threshold $2H$ on the basis of the ordering of the growth rates. For ReduceMax, instead, it can be proved it is a priority scheduling regardless of the ordering of the bamboos' growth rates. This means it can be applied to a wider range of input configurations, not only to ordered ones, as required by ReduceFastest₂.

Fact 1. ReduceMax is a priority scheduling.

Proof. Given any threshold $\tau \geq H$ (as required by Definition 4) each day ReduceMax cuts the tallest bamboo. This includes also the case when there are bamboos taller than τ . This in turn means that ReduceMax gives priority to bamboos higher than τ , if any. \square

The next lemma gives an upper bound on the volume the garden may reach in a priority scheduling. This is useful to prove that any priority scheduling stabilises into a cycle of finite length.

Lemma 1 (Upper Bound on Volume). *Given a τ -priority scheduling σ_τ and an input I , there exists a time t , such that, for any $t' > t$, it holds $V(C^{t'}) \leq n\tau$, where $C^{t'} \in E(I, \sigma_\tau)$.*

Proof. First, assume that $M(E(I, \sigma_\tau)) > \tau$, as otherwise $V(C) \leq n\tau$, for each $C \in E(I, \sigma_\tau)$, and the lemma holds. If it is also assumed $M(C^0) > \tau$ then it is possible to prove that, within finite time, a configuration C with $M(C) \leq \tau$ is reached by applying σ_τ . In particular, note that for as long as there are bamboos having height greater than τ , the total volume decreases each day, being $\tau \geq H$, of at least 1. Thus, there must exist a time t when eventually $M(C^t) \leq \tau$, since the volume cannot decrease indefinitely. At this time, it holds $V(C^t) \leq n\tau$.

Therefore, let $t' > t$ be the first day after time t such that $C^{t'}$ has a bamboo having height greater than τ . Clearly, $V(C^{t'}) < V(C^{t'-1}) - \tau + H$, as σ_τ cuts a bamboo with height greater than τ and, at the end of the day, all bamboos grow by H in total. Moreover, by hypothesis $V(C^{t'-1}) \leq n\tau$, since $\ell_i \leq \tau$, for each $\ell_i \in C^{t'-1}$. Hence,

$$V(C^{t'}) < (n-1)\tau + H \leq n\tau$$

as $H \leq \tau$. Now, by focussing on $V(C^{t'+1})$, since in $C^{t'}$ the scheduling algorithm cuts a bamboo having height bigger than τ , and since the sum of daily growths is exactly H , it follows that $V(C^{t'+1}) < V(C^{t'})$. This is actually true for any following configuration until day t'' where $\ell_i \leq \tau$ for any $\ell_i \in C^{t''}$. Notice that in t'' , the same set of hypotheses as of day t hold, then by repeating the reasoning, the claim follows. \square

By Lemma 1, the next corollary can be obtained, which guarantees that any priority scheduling stabilises into a cycle of finite length, i.e., that the sequence of cuts becomes periodic.

Corollary 1 (Existence of a Cycle). *Given a τ -priority scheduling σ_τ and an input I , there exist two days t and t' , where $t < t'$, $C^t = C^{t'}$, and $C^t, C^{t'} \in E(I, \sigma_\tau)$.*

Proof. The claim follows from Lemma 1, since the number of configurations having volume at most $n\tau$ is finite. \square

By the above corollary, it follows that any priority scheduling stabilises in a cycle, eventually. In fact, by Definition 1, a priority scheduling (which is an oblivious scheduling) computes the same trimming operation if the same configuration shows up again. It is worth reminding that, in the case of ties, the bamboo having the biggest index is cut. In this paper, the focus is on the properties of such cycles, as they represent the perpetual trimming process that have to be executed indefinitely. Hence, the trend will be to disregard configurations preceding L_E where, given an execution E , L_E denotes its periodic part (i.e., the sequence of configurations in the cycle). In particular, to better characterise such cycles, the next notation is introduced:

- $L_E = (C_1 = C^{t'}, C_2 = C^{t'+1}, C_3, \dots, C_{\lambda_E})$, where $C^{t'}$ is the first configuration belonging to L_E , reached from C^0 , and $\lambda_E = |L_E|$ is the length of the cycle, i.e., the number of configurations in L_E .
- l^t is the height of the bamboo cut in $C_t \in L_E$. It is assumed $l^t = 0$ if no bamboo is cut.
- c_i is the number of times bamboo b_i is cut in L_E , for each $i = 1, \dots, n$, which is equal to the number of relative maximum heights reached by b_i in the cycle.
- m_i^j is the relative maximum height reached by b_i in the cycle just before the j th cut, for $i = 1, \dots, n$ and $j = 1, \dots, c_i$. Note that, by definition, during day t , $l^t = m_i^j$ for some values of i and j .
- $M_i = \sum_{j=1}^{c_i} m_i^j$ is the sum of the relative maximum heights reached by b_i .

The next lemma provides a very useful property of the cuts that are performed within a cycle of a priority scheduling. In particular, it can be shown that the average value of heights reached by bamboos in L_E , just before a cut, is always H . As shown below, the lemma is exploited to prove that within some specific circumstances ReduceMax is a 2-approximation algorithm.

Lemma 2 (On Average Height of Cuts within a Cycle). *Given an execution E of a priority scheduling, then $\frac{1}{\lambda_E} \sum_{t=1}^{\lambda_E} l^t = H$.*

Proof. Let $\Delta_V^t = V(C_t) - V(C_{t-1})$ be the change of the volume from C_{t-1} to C_t , for any $t = 1, \dots, \lambda_E$. $C_0 \equiv C_{\lambda_E}$, since cycle L_E exists by Corollary 1. Thus, it holds $\Delta_V^t = H - l^t$, because at day t the bamboo of height l^t is cut and all bamboos grow by H in total. Now, since $\sum_{t=1}^{\lambda_E} \Delta_V^t$ must be equal to zero, as configurations in L_E come periodically, then:

$$\sum_{t=1}^{\lambda_E} \Delta_V^t = \sum_{t=1}^{\lambda_E} (H - l^t) = \lambda_E \cdot H - \sum_{t=1}^{\lambda_E} l^t = 0.$$

Therefore, it is obtained that

$$\sum_{t=1}^{\lambda_E} l^t = \lambda_E \cdot H$$

and the claim holds. \square

The next lemma as well as the subsequent corollary are not exploited in the rest of the paper as the focus is on analysing the performance of ReduceMax. However, such results constitute interesting and elegant properties characterising general priority schedulings, hence they are worth being reported. In particular, in what follows, the length of L_E is characterised in terms of the maximum height reached by any bamboo having growth rate equal to h_i .

Lemma 3 (Characterisation of λ_E). *Given an execution E of a priority scheduling and an index $i \in \{1, 2, \dots, n\}$, then*

$$\lambda_E = \frac{M_i}{h_i}.$$

Proof. Let $p_i^j = m_i^j / h_i$. The value of p_i^j is an integer representing the number of days between the j th cut of b_i and the previous one. Then,

$$M_i = \sum_{j=1}^{c_j} m_i^j = \sum_{j=1}^{c_j} h_i \cdot p_i^j = h_i \cdot \sum_{j=1}^{c_j} p_i^j = h_i \cdot \lambda_E.$$

\square

An immediate consequence is the following corollary.

Corollary 2 (Properties of Rates and Maximum Heights). *Given an execution E of a priority scheduling, and two indices $i, j \in \{1, 2, \dots, n\}$, then:*

$$\frac{M_i}{M_j} = \frac{h_i}{h_j}.$$

3.2. ReduceMax Scheduling

In this section, the focus is on ReduceMax. The aim is to show that ReduceMax performs better than an $O(\log n)$ -approximation algorithm. Actually, the approximation ratio is improved only for

a specific set of input instances, by also exploiting some results achieved for priority schedulings. The results contained in this section constitute a path toward a partial achievement of the aim, leading to a 2-approximation ratio for ReduceMax when confined to some specific instances. By Fact 1, it is shown above that ReduceMax is a priority scheduling, regardless of the chosen threshold $\tau \geq H$ required by Definition 4. It follows that ReduceMax inherits all results obtained in the previous section for priority schedulings. In particular, given an execution E obtained by applying ReduceMax, let l^t be the height of the bamboo cut in $C_t \in L_E$, i.e., $l^t = M(C_t)$, and let $\overline{M}_E = \frac{1}{\lambda_E} \sum_{t=1}^{\lambda_E} l^t$ denote the average of the maximum heights reached by bamboos in L_E . Then, for ReduceMax, Lemma 2 can be reformulated as follows.

Corollary 3 (On Average Height of Cuts of ReduceMax). *Let E be an execution of ReduceMax. Then, $\overline{M}_E = H$.*

A natural intuitive property is provided in the next lemma.

Lemma 4 (On the Amounts of Cuts within a Cycle). *Let E be an execution of ReduceMax. For any two indices $i, j \in \{1, 2, \dots, n\}$ such that $h_i \geq h_j$, it holds $c_i \geq c_j$.*

Proof. By contradiction, assume that $c_i < c_j$. Then, in cycle L_E there must be at least two cuts of b_j between two consecutive cuts of b_i . However, since $h_i \geq h_j$, b_j after the first cut will grow less than b_i and it will never reach b_i before its second cut, which is a contradiction. \square

A direct consequence of Lemma 4 is that, if two bamboos exhibit the same growth rate, they also have the same number of cuts in the cycle.

Corollary 4 (Sufficient Condition for Same Number of Cuts). *If E is an execution of ReduceMax, $h_i = h_j$ implies $c_i = c_j$.*

Let $m_i = \max\{m_i^j : j = 1, 2, \dots, c_i\}$ be the maximum height reached by bamboo b_i in a cycle L_E of ReduceMax. The next result shows that $c_i = 1$ suffices to guarantee $m_i < 2H$. Basically, the next lemma guarantees that the approximation factor of ReduceMax reduces from $O(\log n)$ [1] to 2 if all bamboos are cut only once within the cycle, i.e., when the cycle of ReduceMax is equivalent to the round-robin strategy.

Lemma 5 (Sufficient Condition for Bounded Maximum). *Let E be an execution of ReduceMax. Then,*

$$c_i = 1 \text{ for some } i \in \{1, 2, \dots, n\} \implies m_i < 2H.$$

Proof. First, notice that $c_i = 1$ suffices to immediately obtain $m_i = \lambda_E \cdot h_i$. Moreover, by Corollary 3 it holds $\overline{M}_E = H$, i.e., that $\lambda_E \cdot H - \sum_{t=1}^{\lambda_E} l^t = 0$.

Now, it is known that b_i is cut only once and, as L_E is periodic, C_1 is equal to the configuration where the height of bamboo b_i is h_i . Thus, each term l^t is trivially lower bounded by the height of bamboo b_i at day t , i.e., $l^t \geq t \cdot h_i$. Hence, $\lambda_E \cdot H - \sum_{t=1}^{\lambda_E} t \cdot h_i \geq 0$. Therefore, it holds that

$$\lambda_E \cdot H \geq \sum_{t=1}^{\lambda_E} t \cdot h_i \geq \frac{\lambda_E \cdot (\lambda_E + 1)}{2} \cdot h_i$$

which implies

$$\frac{\lambda_E^2}{2} \cdot h_i + \frac{\lambda_E}{2} \cdot h_i - \lambda_E \cdot H \leq 0$$

that is

$$\lambda_E \cdot h_i + h_i - 2 \cdot H \leq 0.$$

This in turn implies

$$\lambda_E \cdot h_i < 2H$$

and hence $m_i < 2H$. \square

If $c_i = 1$, for some $i = 1, 2, \dots, n$, then similar to the proof of Lemma 5, a limit on the value of λ_E can be obtained, as summarised by the following corollary.

Corollary 5 (Sufficient Condition for Bounded Length of Cycles). *Let E be an execution of ReduceMax. Then,*

$$c_i = 1 \text{ for some } i \in \{1, 2, \dots, n\} \implies \lambda_E \leq \frac{2 \cdot H}{h_i} - 1.$$

Given the above results, and an execution E , configurations where the bamboo having the largest growth rate, e.g., b_m , such that $c_m = 1$, are now characterised. By Lemma 4, $c_m = 1$ directly implies $c_i = 1$ for all $i = 1, 2, \dots, n$.

To this aim, given the set of growth rates h_1, h_2, \dots, h_n , the next notation is used, by: (i) $h_{(k)}$, $k \in [1, n]$, its k th order statistic, that is the k th smallest value in an ordered view of the set (e.g., $h_{(1)}$ denotes the smallest rate while $h_{(n)}$ denotes the largest one); and (ii) $b_{(k)}$, $k \in [1, n]$, the bamboo associated with $h_{(k)}$. Moreover, given an execution E , $c_{(k)}$ ($m_{(k)}$, respectively), $k \in [1, n]$, denote the number of times bamboo $b_{(k)}$ is cut (the maximum height reached by $b_{(k)}$, respectively) in L_E . The next result can be now proved.

Lemma 6 (Characterising Configurations Having $c_{(n)} = 1$). *Given an input I , let E be an execution of ReduceMax. Then:*

$$c_{(n)} = 1 \iff \frac{h_{(k)}}{h_{(k+1)}} \geq 1 - \frac{1}{n} \text{ where } k \in \{1, 2, n-1\}.$$

Proof. (\implies) Since $c_{(n)} = 1$, by Lemma 4, it is known that $c_i = 1$ for any $i = 1, 2, \dots, n$ and then $\lambda_E = n$. Hence, the maximum height reached by bamboo $b_{(n)}$ is $m_{(n)} = n \cdot h_{(n)}$, at day n . This means that there exists a bamboo $b_i \neq b_{(n)}$, such that, at day $n-1$, bamboo b_i has length larger than $b_{(n)}$, i.e., $m_i = n \cdot h_i \geq (n-1) \cdot h_{(n)}$. Consequently, given $h_{(n)} \geq h_{(n-1)} \geq h_i$, it follows that $n \cdot h_{(n-1)} \geq (n-1) \cdot h_{(n)}$ and hence:

$$\frac{h_{(n-1)}}{h_{(n)}} \geq \frac{n-1}{n}$$

By the same reasoning as above, when bamboo $b_{(n-1)}$ reaches its maximum length $m_{(n-1)} = h_{(n-1)} \cdot n$, there must exist another bamboo b_j , such that $b_j \neq b_{(n)}$, $b_j \neq b_{(n-1)}$ and $m_j = n \cdot h_j \geq (n-1) \cdot h_{(n-1)}$. Given that $h_{(n)} \geq h_{(n-1)} \geq h_{(n-2)} \geq h_j$, it follows that $n \cdot h_{(n-2)} \geq (n-1) \cdot h_{(n-1)}$ and hence:

$$\frac{h_{(n-2)}}{h_{(n-1)}} \geq \frac{n-1}{n}.$$

In general, the above reasoning can be iterated up to n times to obtain:

$$\frac{h_{(k)}}{h_{(k+1)}} \geq \frac{n-1}{n} = 1 - \frac{1}{n}$$

and the first implication of the claim follows.

(\Leftarrow) By construction, it is possible to build $L_E = (C_1, C_2, \dots, C_n)$ of an execution E such that $c_i = 1$ for each $i = 1, 2, \dots, n$. Now, let

$$C_1 = (1 \cdot h_{(n)}, 2 \cdot h_{(n-1)}, \dots, (n-1) \cdot h_{(2)}, n \cdot h_{(1)}).$$

Since

$$\frac{h_{(k)}}{h_{(k+1)}} \geq 1 - \frac{1}{n}$$

for any $k \in \{1, 2, n-1\}$, it follows that

$$(k+1) \cdot h_{(k)} \geq k \cdot h_{(k+1)}$$

for each $k \in \{1, 2, n-1\}$. In fact, it holds

$$(k+1) \cdot h_{(k)} \geq (k+1) \cdot \left(1 - \frac{1}{n}\right) \cdot h_{(k+1)} = (k+1) \cdot h_{(k+1)} - \frac{k+1}{n} \cdot h_{(k+1)} \geq k \cdot h_{(k+1)}.$$

Then, in C_1 , bamboo $b_{(1)}$ is cut and the resulting configuration is

$$C_2 = (2 \cdot h_{(n)}, 3 \cdot h_{(n-1)}, \dots, n \cdot h_{(2)}, 1 \cdot h_{(1)}).$$

By a simple algebraic manipulation, it is possible to infer that in C_2 the maximal bamboo height is $n \cdot h_{(2)}$, and hence bamboo $b_{(2)}$ is cut.

In general, for $t = 1, 2, \dots, n$, it holds

$$C_t = (t \cdot h_{(n)}, (t+1) \cdot h_{(n-1)}, \dots, n \cdot h_{(t)}, 1 \cdot h_{(t-1)}, 2 \cdot h_{(t-2)}, \dots, (t-1) \cdot h_{(1)})$$

and, in C_t , the resulting bamboo that is cut is $b_{(t)}$, having height $n \cdot h_{(t)}$. In fact, in C_t , the height of bamboo $b_{(t)}$ is straightforwardly larger than that of any bamboo $b_{(i)}$ for $i < t$. For $i > t$, instead, since the height of bamboo $b_{(i)}$ is $(t+i) \cdot h_{(i)}$, it follows, by a similar argument as above, that $(t+i) \cdot h_{(i)} \geq (t+i-1) \cdot h_{(i+1)}$, hence $b_{(t)}$ is the bamboo exhibiting the maximal height in C_t .

When $t = n$, it is possible to observe that $C_t = (n \cdot h_{(n)}, 1 \cdot h_{(n-1)}, 2 \cdot h_{(n-2)}, \dots, (n-1) \cdot h_{(1)})$, where clearly $b_{(n)}$ has the maximal height and it is cut, leading to a new configuration which coincides with C_1 . \square

Reminding that, when dealing with ReduceMax, it is not required to have bamboos ordered according to their growth rates, in what follows, executions having $c_{(n)} = 1$ are considered as *minimum-cycle executions*, since $c_{(n)} = 1$ implies that $c_i = 1$ for any bamboo b_i , and then the execution exhibits a minimum-length cycle, that is $\lambda_E = n$. Note that λ_E cannot be smaller than n , as each bamboo must be cut at least once. These observations are summarised in the next corollary.

Corollary 6 (Characterisation of Minimum-cycle Executions). *Let E be an execution of ReduceMax. Then,*

$$c_{(n)} = 1 \iff \lambda_E = n.$$

Finally, the next corollary can be also obtained for minimum-cycle executions of ReduceMax.

Corollary 7 (On the Maximum Height in Minimum-cycle Executions). *Let E be an execution of ReduceMax. If E is a minimum-cycle execution, then $M(L_E) < 2H$.*

Proof. In a minimum-cycle execution, the maximum height reached by a bamboo b_i is exactly $m_i = nh_i$. Hence, the maximum height reached during an execution by any bamboo is due to $h_{(n)}$ and,

in particular, is given by $M(L_E) = m_{(n)} = n \cdot h_{(n)}$. Hence, the thesis is an immediate consequence of Lemma 5, as $c_{(n)} = 1$. \square

Corollary 7 is of particular interest, since by Conjecture 1 it has been essentially stated that $M(E) < 2H$ for all the executions E obtained via ReduceMax. Hence, Corollary 7 represents a partial proof to the conjecture, holding for the case of minimum-cycle executions. In the next section, the validity of the posed conjecture on a large set of inputs is experimentally evaluated.

Clearly, minimum-cycle executions are desirable as they guarantee $M(E) < 2H$. However, from a practical view point, round-robin schedulers are not always the most desirable solutions as different properties and requirements might be involved. In particular, the following result provides a sufficient condition for non-minimum cycle executions.

Theorem 1 (Sufficient Condition for Non-minimum-cycle Executions). *Given an input I , let E be an execution obtained via ReduceMax. Then, E is not a minimum-cycle execution if*

$$\frac{h_{(1)}}{h_{(n)}} > e$$

where e is Euler's number.

Proof. By contradiction, assume that $\frac{h_{(1)}}{h_{(n)}} > e$ holds in E and that $c_{(n)} = 1$, i.e., E is a minimum-cycle execution. Then, by Lemma 6, it holds

$$\frac{h_{(i)}}{h_{(i+1)}} \geq 1 - \frac{1}{n}, \quad i \in \{1, 2, n-1\}$$

which implies

$$\frac{h_{(i+1)}}{h_{(i)}} \leq \frac{1}{1 - \frac{1}{n}} = \left(1 + \frac{1}{n-1}\right)$$

and hence

$$\frac{h_{(n)}}{h_{(1)}} = \frac{h_{(n)}}{h_{(n-1)}} \cdot \frac{h_{(n-1)}}{h_{(n-2)}} \cdot \frac{h_{(n-2)}}{h_{(n-3)}} \cdots \frac{h_{(3)}}{h_{(2)}} \cdot \frac{h_{(2)}}{h_{(1)}} \leq \left(1 + \frac{1}{n-1}\right)^{n-1}.$$

Moreover, it holds that $\left(1 + \frac{1}{n-1}\right)^{n-1}$ is an increasing function of n . Therefore, since

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n-1}\right)^{n-1} = e$$

it follows that

$$\frac{h_{(n)}}{h_{(1)}} \leq \left(1 + \frac{1}{n-1}\right)^{n-1} \leq e$$

which is clearly a contradiction. \square

4. Experimental Results

In what follows, an extensive experimental evaluation of four priority scheduling strategies is provided. As already pointed out, a requirement for the scheduling strategies is to be fully deterministic. Hence, in the case of ties, that is when two or more bamboos are eligible to be cut with respect to the considered strategy, the bamboo having the biggest index is selected. The considered strategies are:

- **ReduceMax** (**RMax**, for short): This is the heuristic which performance is the most relevant to the experiments. In particular, in [1], based on [27], a $O(\log n)$ -approximation guarantee has been established. However, the interest is in determining whether such a bound is tight in practice, i.e., whether the logarithmic factor is an accurate estimation. The strategy works in a greedy fashion by cutting each day the tallest bamboo.
- **ReduceFastest₂** (**RFast₂**, for short): This is another greedy strategy introduced in [1]. It guarantees 4-approximation. However, this method requires ordering the input configurations according to the non-increasing order of the bamboos' growth rates. In fact, each day it cuts the fastest growing bamboo (the one having the biggest h_i) among those whose height exceeds threshold $2H$. If none of the bamboos is taller than $2H$, no cuts are performed.
- **ReduceFastest₁** (**RFast₁**, for short): This is a variant of **RFast₂**, introduced here for the first time, obtained by decreasing the threshold from $2H$ to H , and by allowing the cut of the fastest growing bamboo also below the threshold. Basically, if none of the bamboos has reached height H , the fastest growing bamboo is cut. This is a natural extension of **RFast₂**, and the aim of defining it is to check whether there are chances to obtain better performance with respect to **RFast₂** and **RMax**. Note that **RFast₁** is an ordered H -priority scheduling.
- **ReduceMin** (**RMin**, for short): This priority algorithm cuts each day the shortest bamboo, giving priorities to those above H . The aim of defining this strategy is to evaluate performance of counter-intuitive methods, i.e., to see whether even in an adversarial approach one may obtain acceptable performances. **RMin** is a H -priority scheduling.

To evaluate the behaviour of the scheduling strategies, all above mentioned scheduling algorithms have been implemented in C++. Moreover, it has been implemented a simulation framework for both generating input instances of the problem and measuring the performance, with respect to different metrics of interest, of the approaches on said instances. Different types of experiments have been performed, with varying values of n and H and different distribution of growth rates, which are described in the following sections.

All sources were compiled with GNU g++ version 7.3.0 (O3 optimisation level) under Ubuntu Linux (Kernel 4.15.0-38). All tests were executed on a workstation equipped with an Intel®Xeon®CPU E5-2643 v3 3.40GHz CPU and 128 GB of RAM.

Notice that, for a fair comparison with respect to [1], in the experiments, it was assumed that the heights of all bamboos are initially null, that is $C^0 = (0, 0, \dots, 0)$. Moreover, as the experiments involved ordered priority scheduling (namely **RFast₁** and **RFast₂**), without loss of generality, ordered configurations were considered.

4.1. Fixed H Experimentation

In the first set of experiments, parameter H was fixed and all possible input instances of n bamboos were considered whose growth rates sum up to H . Such instances were generated by considering, for a given $H \in \mathbb{N}$, all integer partitions of H . Hence, clearly, the resulting n was in $\{1, 2, \dots, H\}$, where for $n = 1$ there was only one bamboo having growth rate equal to H while for $n = H$ all bamboos had unitary growth rate (e.g., for $H = 3$, we had instances [3], [2, 1] and [1, 1, 1]). Note that, for H , values in the set $\{5, 10, 15, 20, 25, 30, 35\}$ were selected. This choice was dictated by the fact that the number of integer partitions, and therefore instances to consider, grows very quickly as H increases and hence too large values of H induce a computationally prohibitive number of simulations. In particular, it is known that the number of integer partitions $p(k)$ of a natural number k grows asymptotically as $p(k) \approx \frac{1}{4k\sqrt{3}} e^{\pi\sqrt{\frac{2k}{3}}}$ as k approaches infinity [30]. For instance, $p(100) = 190,569,292$ while $p(1000) \approx 2.41 \cdot 10^{31}$. Figure 2b provides an overview of the total number of inputs considered in this experiment. However, the trend of the obtained results does not seem to be affected for H growing.

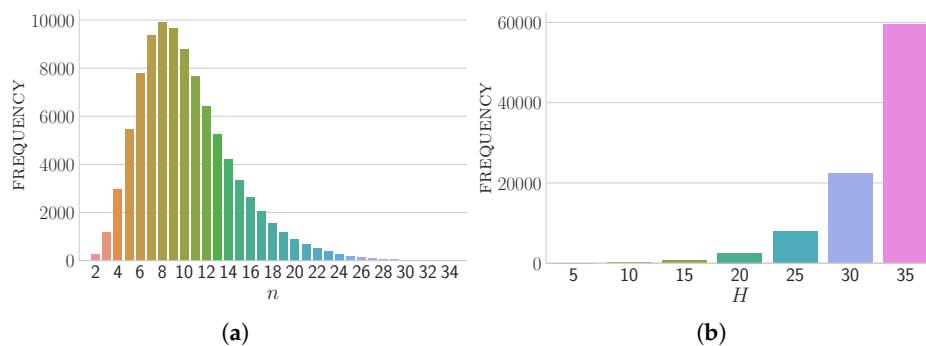


Figure 2. (a) Distribution of instances with respect to n for all partitions of integer $H \in \{5, 10, 15, 20, 25, 30, 35\}$; and (b) distribution of instances with respect to H for all partitions of integer $H \in \{5, 10, 15, 20, 25, 30, 35\}$.

The results of the application of all the considered scheduling algorithms, on instances induced by the integer partitions $H = 30$ ($H = 35$, respectively) are shown in Figure 3 (Figure 4, respectively). Panels on the left side show how the maximum $M(E)$, obtained for all configurations and strategies, varies as a function of n (see Figures 3a,c and 4a,c, respectively). Reference lines $2H$ and $4H$ are plotted to emphasise performance of the strategies in terms of $M(E)$. Panels on the right instead, instead, show how the achieved maximum value of λ_E changes as a function of n , given that all strategies are guaranteed to stabilise into cycles.

Note that the maximum measured value is reported as it is possible to have many instances having the same n . Notice also that the results for $H \in \{5, 10, 15, 20, 25\}$ lead to similar considerations with respect to those provided below, hence they have been omitted.

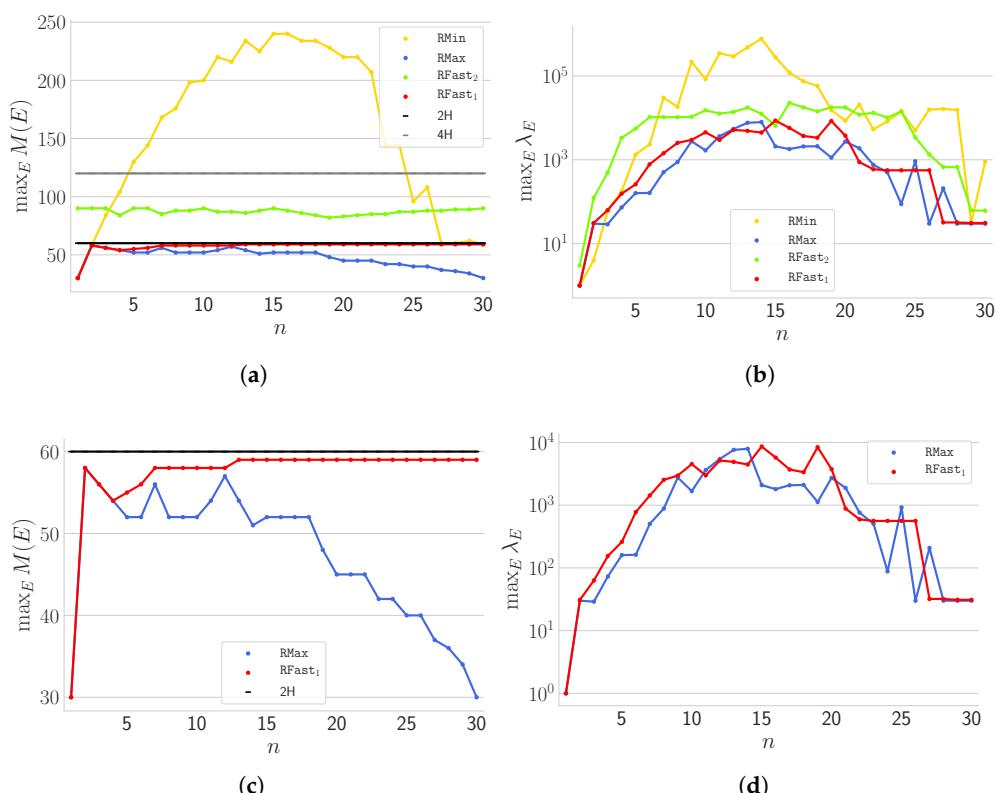


Figure 3. Experiments conducted on all possible ordered instances obtained by setting $H = 30$ and hence considering n varying in $\{1, 2, \dots, 30\}$. Panels (a,c) refer to maximum $M(E)$, whereas panels (b,d) refer to maximum λ_E . Panels (c,d) show strategies that, experimentally, exhibit 2-approximation.

An alternative view of the results of the experiments for $H = 35$ is given in Figure 5 where it is shown how the obtained values of $M(E)$ and λ_E are distributed over the considered instances. In detail, each value on the x -axis in these diagrams simply represents one instance, to which the corresponding values of $M(E)$ and λ_E has been associated on the y -axis. Instances are sorted on the x -axis in non-decreasing order according to their values on the y -axis to highlight the amount of inputs providing a same value of $M(E)$ and λ_E . This information is condensed in Figure 4 where n is on the x -axis and the maximum values are plot only for each set of instances sharing the same value of n .

On the Maximum Height. The main and the most interesting outcome of this experiment is that, notwithstanding the $O(\log n)$ approximation factor [1], RMax exhibits properties of a 2-approximation, which can be observed as $M(E) < 2H$. This supports the conjecture on the worst case approximation factor that has been proved so far being an over estimation of the true approximation upper bound. Another rather surprising evidence is that also RFast₁ is always below $2H$ in terms of $M(E)$. This suggests that, for this strategy as well there could be a way to prove the worst case 2-approximation. However, RFast₁ seems to follow an asymptotic trend toward $2H$ as n increases while a slowly decreasing trend for RMax when n increases can be observed.

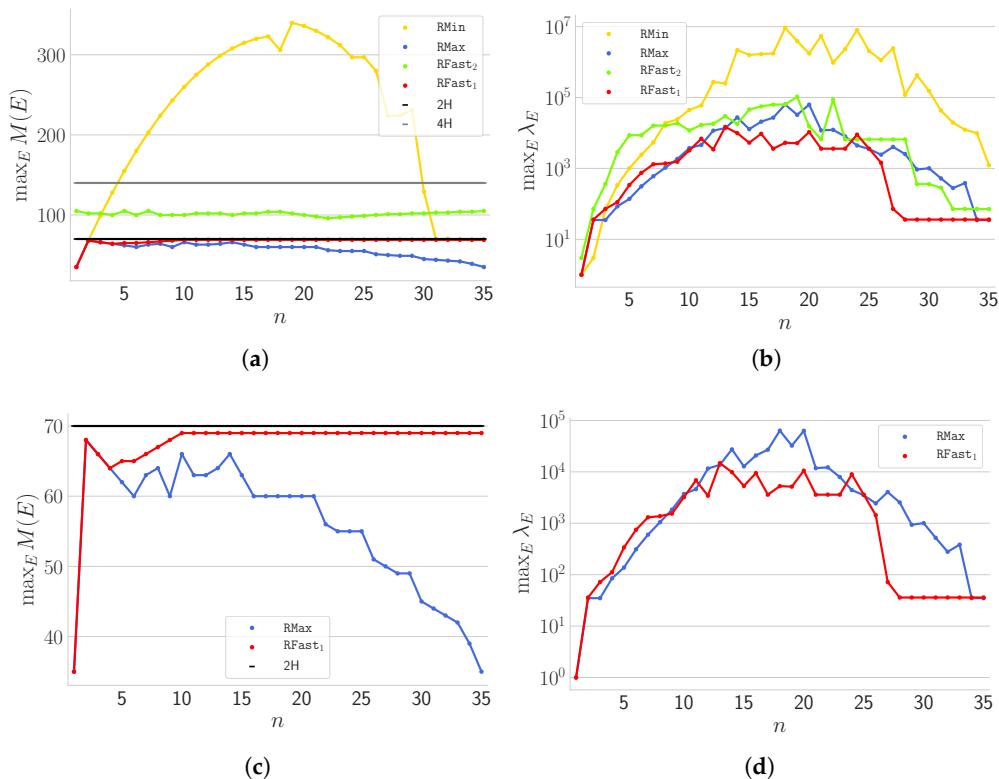


Figure 4. Experiments conducted on all possible ordered instances obtained by setting $H = 35$ and hence considering n varying in $\{1, 2, \dots, 35\}$. Panels (a,c) refer to maximum $M(E)$, whereas panels (b,d) refer to maximum λ_E . Panels (c,d) show strategies that, experimentally, exhibit 2-approximation.

Concerning RFast₂, from [1] it is known that $M(E)$ is guaranteed to be below $4H$ and this is confirmed by the performed tests. Nonetheless, it can be observed that $M(E)$ is always above $2H$, which is expected since no actions are performed by RFast₂ when there are no bamboos having height above $2H$. However, the strategy exhibits a rather uniform behaviour, with $M(E)$ stabilising toward threshold $3H$ without ever overpassing it. This suggests that perhaps also the bound of $4H$ guaranteed for RFast₂ is an overestimation of the true bound. Finally, regarding RMin, for low/high values of n (see Figures 3a and 4a), it provides smaller values of $M(E)$ with respect to RFast₂, whereas for higher values of H its performance gets worse in the opposite sense. However, there is no evidence of RMin

exhibiting a constant approximation, with values of $M(E)$ varying quite a lot with n . As a final remark, in Figure 5a, it can be observed that, for all strategies, values of maximum $M(E)$ tend to have small variance among all instances having a same H , with curves assuming rather similar (flat) trends, and values always being very close to the average. The only exception is algorithm RMin, whose values of $M(E)$ are quite different across instances having a same H . Moreover, RMax achieves values of $M(E)$ that are far better than all other strategies, including RFast₁, being below $2H$ and, in some cases, below $\frac{3}{2}H$. This is even more evident in Figures 3c and 4c where the focus is restricted on RMax and RFast₁, i.e., on experiments where $M(E)$ is observed to be always below $2H$. Finally, a global view supporting the conclusions with respect to the performance of the strategies in terms of $M(E)$ is given in the scatterplot chart of Figure 6a where all obtained values of $M(E)$ for all strategies and for all values of H are reported and compared.

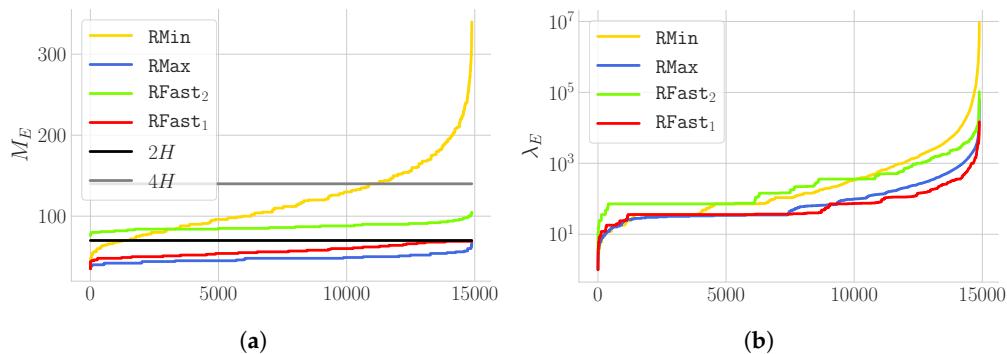


Figure 5. Distribution of values of: $M(E)$ (a); and λ_E (b) exhibited by all algorithms on instances induced by all partitions of $H = 35$. Instances are sorted by non-decreasing values of: $M(E)$ (a); and λ_E (b). To magnify the differences, the y -axis in (b) is log-scaled.

On the Length of the Cycle. In Figures 3b and 4b, the maximum λ_E exhibited by all strategies is reported. Such values can be considered as proxy of the complexity of the periodic part of each execution, as higher values of λ_E correspond to larger spaces of configurations that are explored by the strategies. This translates to higher variance in terms of height and volume, which can be seen as an undesired behaviour. Moreover, there might be also a relationship between such length and the quality of the provided factor of approximation, and it would be worth studying such relationship to define new bounds on this factor.

In more details, the data show a very big gap between the results obtained by RMax, RFast₁, and those measured for RFast₂, RMin, with the latter two tending to exhibit larger values with respect to the former two. In particular, as shown in Figure 4b, when $n = 18$, RMin cyclic part takes around 10^7 steps, whereas the worst case for RFast₂ is obtained for $n = 19$, with around 10^5 steps. Instead, RMax and RFast₁ behave rather differently with respect to other strategies, exhibiting lower values of λ_E (e.g., up to around 3 orders of magnitude lower when $n = 21$).

This may be to more accurate arguments about changes in the volume of the garden, to be exploited in proofs of constant approximation. Different considerations can be done by observing Figure 5b. In particular, the largest values of λ_E are achieved in the great majority of the cases by RFast₂ while RMax and RFast₁ result to be the best strategies also in this sense.

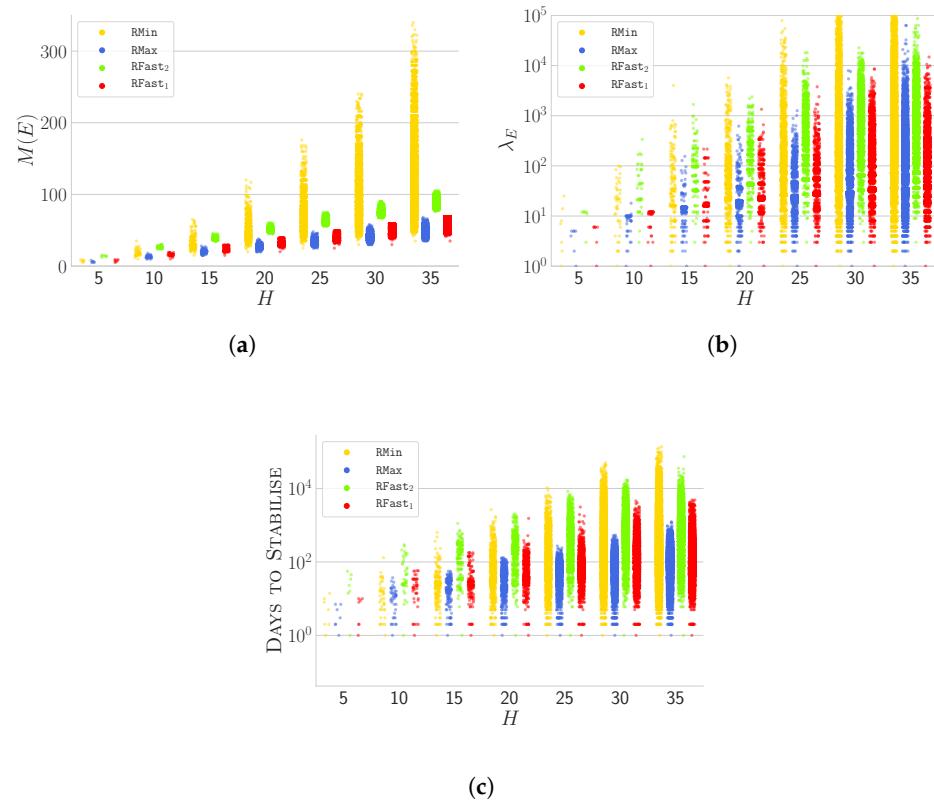


Figure 6. Distribution of values of: $M(E)$ (a); λ_E (b); and number of days before stabilisation (c) as a function of H , for all considered strategies and instances.

4.2. Fixed n Experimentation

In this section, the results of further experimentation are reported where parameter n is fixed and all possible instances with $H \geq n$ are considered, when $H \in \{5, 10, 15, 20, 25, 30, 35\}$. For example, when $n = 15$, then H can assume values 15, 20 and 25. Instances are again chosen among all the integer partitions of specific values of H . Figure 2 provides an overview of the total number of inputs considered in this experiment, as a function of n and H , respectively.

The results of this experiment are summarised in Figures 7 and 8 where the results of application of all considered algorithms on instances with $n = 10$ and $n = 15$, respectively, are shown. In particular, the maximum values of $M(E)$ and λ_E obtained by all strategies as a function of H are reported. Reference lines $2H$ and $4H$ are again used to highlight the quality in terms of $M(E)$ obtained by the different algorithms. Note that graphs for other possible values of n (e.g., $n = 20$) are omitted since they lead to similar considerations with respect to those that follow.

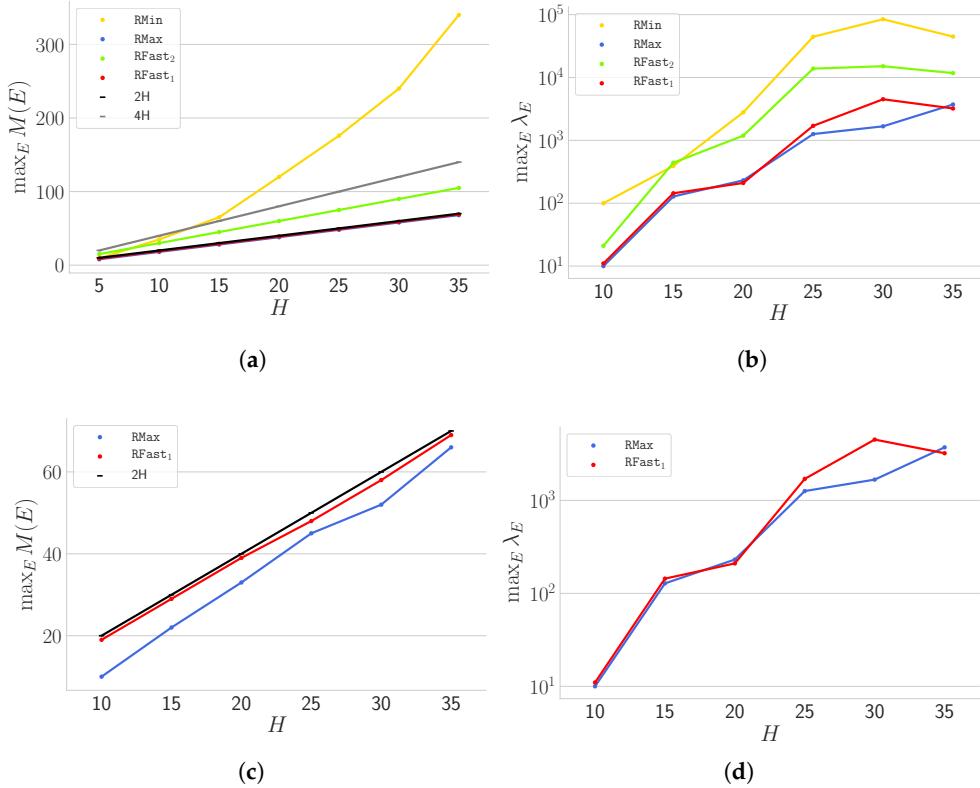


Figure 7. How maximum $M(E)$ (a,c) and maximum λ_E (b,d) change as a function of H when instances are induced by all partitions of $H \in \{10, 15, 20, 25, 30, 35\}$ having cardinality $n = 10$. Panels (c,d) show strategies that, experimentally, exhibit 2-approximation.

On the Maximum Height. As already observed above, in this experiment, it was also possible to notice that RMax and RFast₁ exhibit values of $M(E)$ that are always below $2H$ (see Figures 7 and 8). Moreover, again, RMax seems to be the best strategy, achieving values of maximum $M(E)$ that are below $2H$ and better than all other scheduling strategies, including RFast₁. This is again more evident if the attention is restricted to RMax and RFast₁ only (see Figure 7). Furthermore, the experiments confirm also the theoretical bound of RFast₂, with bamboos never exceeding $4H$. Moreover, all strategies exhibit a trend of $M(E)$ with respect to H that looks linear, with the only exception of RMin, where the curve looks slightly super-linear.

On the Length of the Cycle. In Figures 7b and 8b, the maximum λ_E obtained by all strategies in this setting is reported. The results confirm the trend observed in the case of fixed H . However, here it can be seen how RMin starts growing significantly faster for values of H that are greater than 20. Shortest cycles are obtained by RMax and ReduceFastest₁. Still, the length of the cycle seems dependent on the way ties among bamboos are resolved (see Figure 7d). For the sake of completeness, in Figures 9–12, the entire executions of the scheduling strategies are plotted on four sample instances whose size (i.e., n or H) is larger than all others considered in this paper, namely having:

- $n = 30$ bamboos with $H \in \{70, 100\}$; and
- $n = 10$ with $H \in \{40, 200\}$.

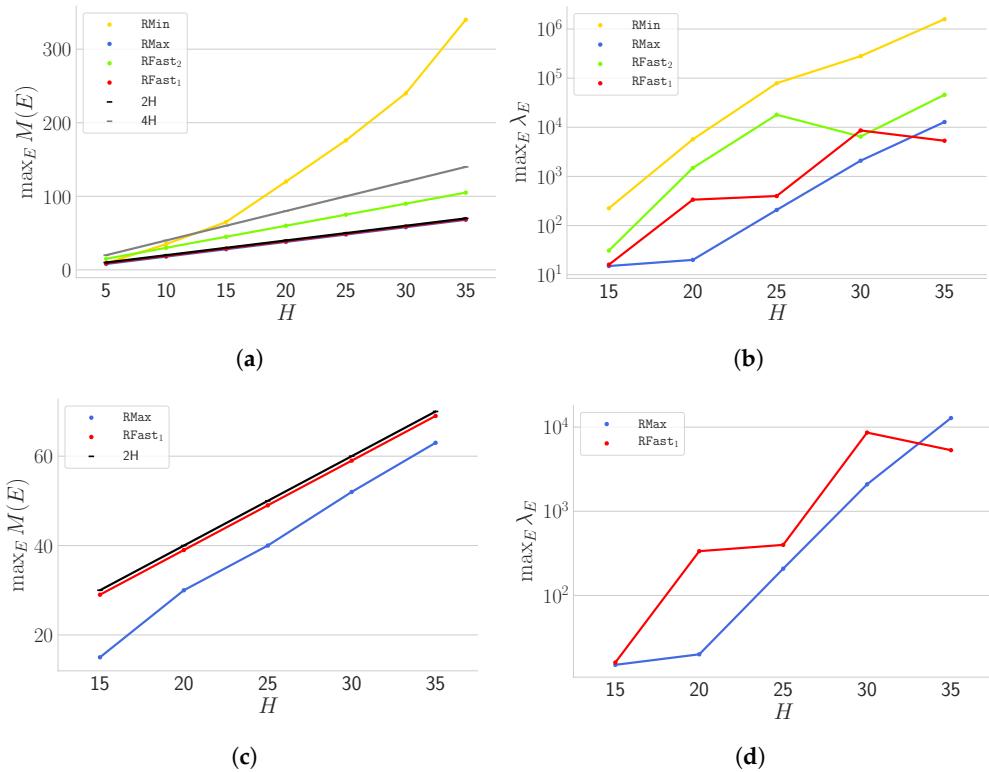


Figure 8. How maximum $M(E)$ (a) and maximum λ_E (b) change as a function of H when instances are induced by all partitions of $H \in \{15, 20, 25, 30, 35\}$ having cardinality $n = 15$. Panels (c,d) show strategies that, experimentally, exhibit 2-approximation.

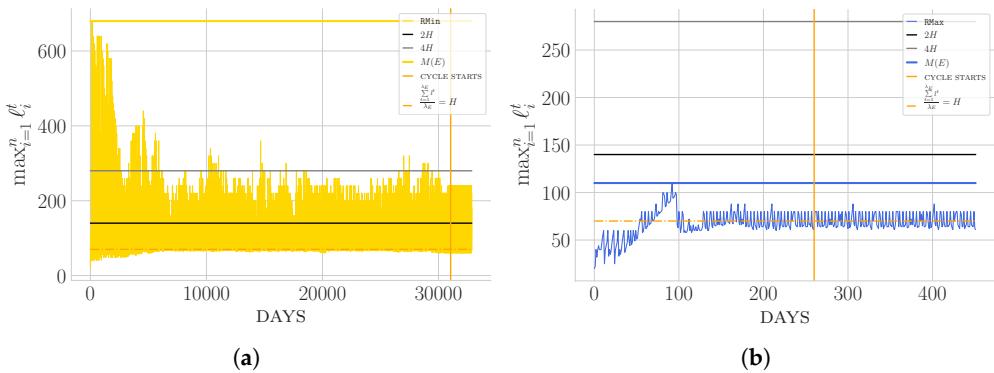


Figure 9. Cont.

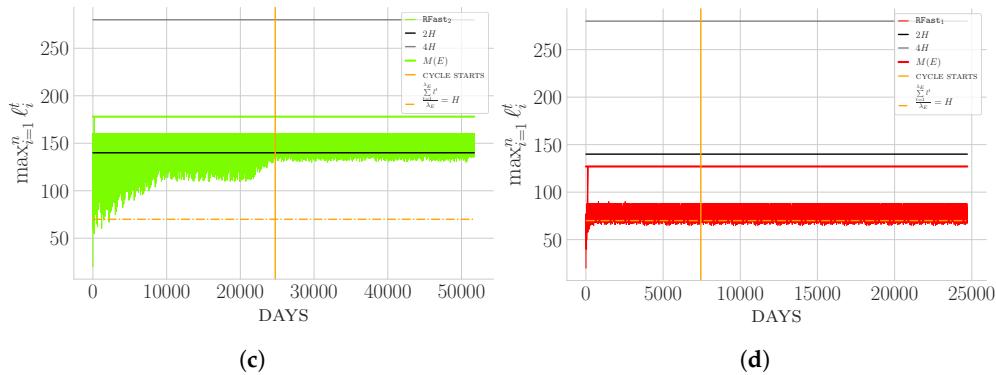
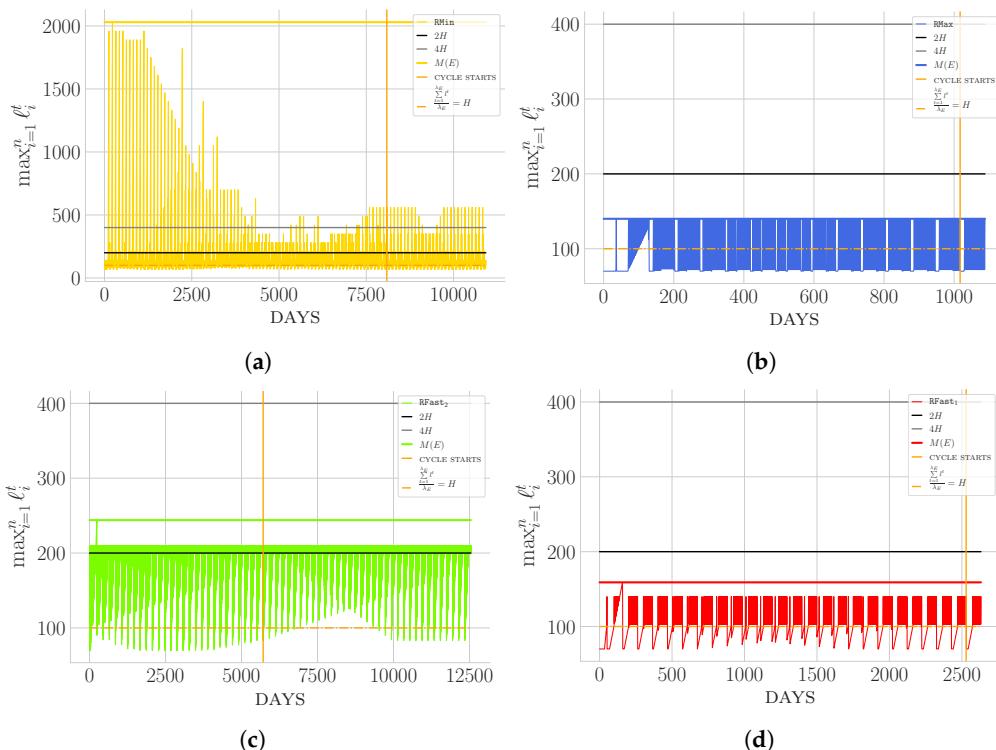


Figure 9. Snapshot of the evolution of maximum heights when strategies: RMin (**a**); RMax (**b**); RFast₂ (**c**); and RFast₁ (**d**) are applied on an instance having $n = 30$ and $H = 70$. (in details, structured as follows [20; 11; 8; 5; 1]). On the x -axis the consecutive days are reported while on the y -axis the maximum height on the corresponding day is plotted. The vertical line shows the beginning of the periodic phase while the horizontal lines show, respectively, $(\sum_{t=1}^{\lambda_E} l^t) / \lambda_E$ (i.e., H), $2H$, $4H$, and $M(E)$ over the entire execution.



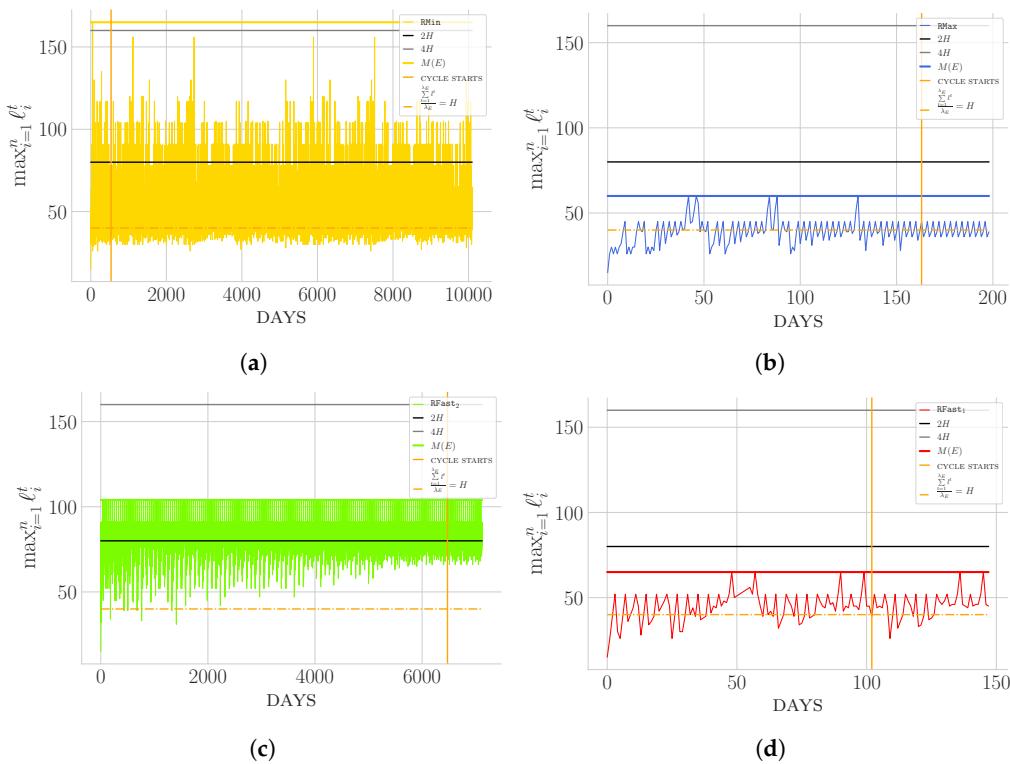


Figure 11. Snapshot of the evolution of maximum heights when strategies: RMin (a); RMax (b); RFast₂ (c); and RFast₁ (d) are applied on an instance having $n = 10$ and $H = 40$ (in details, structured as follows [15; 13; 4; 2; 1; 1; 1; 1; 1]). Data are plotted as in Figure 9.

This is done to highlight how each algorithm stabilises and how values of $M(E)$ evolve both before the cyclic part starts (i.e., during the transient phase) and during such periodic part.

In particular, the height as a function of increasing values of days is shown and both the length of the periodic part and the number of days before each algorithm stabilises are highlighted. The data lead essentially to two major observations:

- (i) Some strategies exhibit a very small variance in terms of height (e.g., RMax) with respect to others (e.g., RMin).
- (ii) Some strategies (e.g., RMax) are able to stabilise very quickly to a cycle, which is a clearly desirable behaviour in a stabilisation perspective. On top of that, they also exhibit a small λ_E .

To support these observations, an aggregated view of all the measures of $M(E)$ and λ_E are shown, for all instances and strategies, as a function of H (see Figure 6a,b). Jointly, the distribution of the length of the transient phases, i.e., the number of days before each algorithm stabilises into the periodic part is reported (see Figure 6c). Furthermore, the panels suggest that the structure of the instances affects somehow the behaviour of the scheduling algorithms and the corresponding schedules. In more details, heterogeneous distributions of rates seem to be associated with higher values of $M(E)$, λ_E and number of days before stabilisation while more evenly distributed rates tend to induce the opposite behavior. Finally, it is worth remarking on the successful verification that all the (priority) strategies investigated clearly do not violate the results of Section 3.1. Moreover, on the basis of the experimental results and in accordance to the special cases studied in Corollary 7, it is worth reporting again Conjecture 1 to emphasise and confirm the arisen intuition:

Conjecture 1. Algorithm ReduceMax is 2-approximating for the BGT problem.

Further considerations for the other heuristics might be deducted, as for instance also ReduceFastest₁ seems to guarantee an approximation ratio of 2. However, as for ReduceFastest₂, it relies on the ordering of the bamboos with respect to the growth rates.

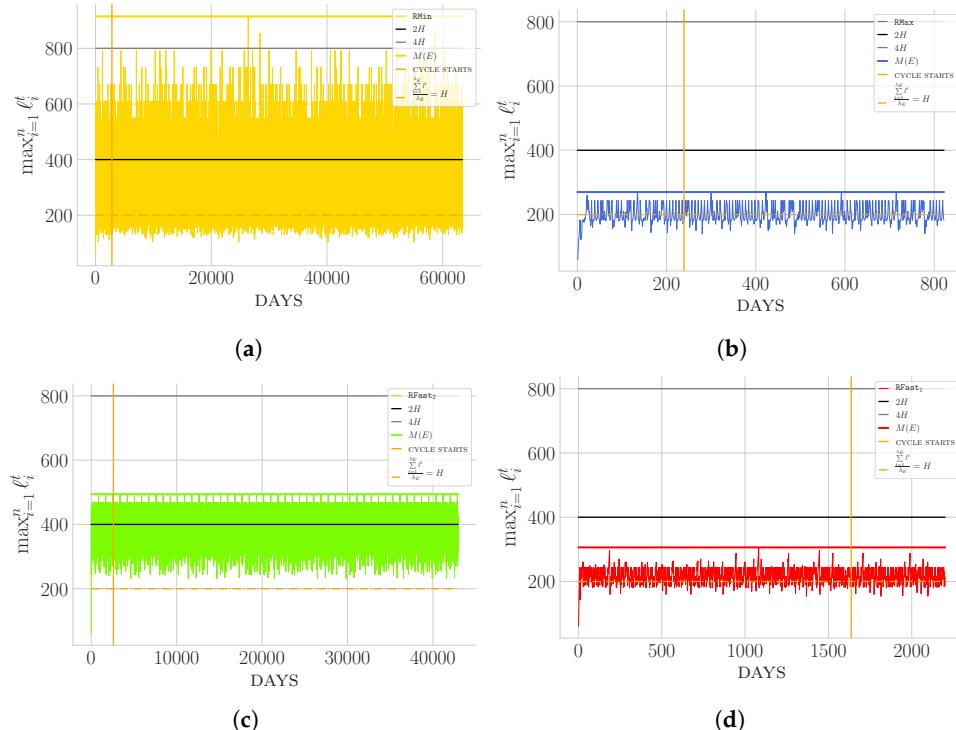


Figure 12. Snapshot of the evolution of maximum heights when strategies: RMin (a); RMax (b); RFast₂ (c); and RFast₁ (d) are applied on an instance having $n = 10$ and $H = 200$ (in details, structured as follows [61; 30; 27; 26; 13; 11; 10; 10; 9; 3]). Data are plotted as in Figure 9.

5. Conclusions

The BGT problem has been investigated to establish whether constant approximation deterministic algorithms can be designed. A new class of scheduling strategies called priority schedulings have been defined and theoretical results on such methods have been provided. In particular, it has been proved that any priority scheduling eventually brings the system to perpetually repeated sequences of configurations. In addition, ReduceMax has been deeply analysed. This is a priority scheduling for which a conjecture of 2-approximation has been claimed. Extensive experimentation was conducted to confirm the intuitions and to show that ReduceMax outperforms any other known strategy, including the 4-approximation ReduceFastest₂ which, unlike ReduceMax, relies on the knowledge of an ordering on the rates. In terms of knowledge required by the cutting strategies, a research direction that surely deserves further investigation is that of considering the more realistic scenario where the input data are not entirely known, i.e., to tackle the problem from an online algorithms perspective. With respect to problem in its formulation, instead, it would be interesting to study variants of the model where, for example: (i) the aim is optimising other metrics of interest; and (ii) the gardener is allowed to perform different types of trimming operations. Finally, it would be worth studying how the results provided in this paper can translate to the continuous version of the problem [1].

Author Contributions: All authors have equally contributed to this work.

Funding: This work has been partially supported by the European project “Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies” (GEO-SAFE), contract no. H2020-691161 and by the Italian National Group for Scientific Computation GNCS-INDAM.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gąsieniec, L.; Klasing, R.; Levcopoulos, C.; Lingas, A.; Min, J.; Radzik, T. Bamboo Garden Trimming Problem (Perpetual Maintenance of Machines with Different Attendance Urgency Factors). In *SOFSEM 2017: Theory and Practice of Computer Science, Proceedings of the 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, 16–20 January 2017*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2017; Volume 10139, pp. 229–240.
2. Holte, R.; Mok, A.; Rosier, L.; Tulchinsky, I.; Varvel, D. The pinwheel: A real-time scheduling problem. In Proceedings of the 22nd Annual Hawaii International Conference on System Sciences, Kailua-Kona, HI, USA, 3–6 January 1989; Volume 2, pp. 693–702.
3. Fishburn, P.C.; Lagarias, J.C. Pinwheel Scheduling: Achievable Densities. *Algorithmica* **2002**, *34*, 14–38. [[CrossRef](#)]
4. Alshamrani, S.; Kowalski, D.R.; Gąsieniec, L. How Reduce Max Algorithm Behaves with Symptoms Appearance on Virtual Machines in Clouds. In Proceedings of the 2015 International Conference on Cloud Computing (ICCC), Riyadh, Saudi Arabia, 26–29 April 2015; pp. 1703–1710.
5. Gąsieniec, L.; Klasing, R.; Martin, R.; Navarra, A.; Zhang, X. Fast periodic graph exploration with constant memory. *J. Comput. Syst. Sci.* **2008**, *74*, 802–822. [[CrossRef](#)]
6. Kosowski, A.; Navarra, A. Graph Decomposition for Memoryless Periodic Exploration. *Algorithmica* **2012**, *63*, 26–38. [[CrossRef](#)]
7. D’Emidio, M.; Di Stefano, G.; Frigioni, D.; Navarra, A. Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane. *Inf. Comput.* **2018**, *263*, 57–74. [[CrossRef](#)]
8. D’Emidio, M.; Frigioni, D.; Navarra, A. Explore and repair graphs with black holes using mobile entities. *Theor. Comput. Sci.* **2015**, *605*, 129–145. [[CrossRef](#)]
9. Ntafos, S. On gallery watchmen in grids. *Inf. Process. Lett.* **1986**, *23*, 9–102. [[CrossRef](#)]
10. Urrutia, J. Art gallery and illumination problems. In *Handbook of Computational Geometry*; Elsevier: Amsterdam, The Netherlands, 2000; pp. 973–1027.
11. Collins, A.; Czyzowicz, J.; Gąsieniec, L.; Kosowski, A.; Kranakis, E.; Krizanc, D.; Martin, R.; Morales Ponce, O. Optimal Patrolling of Fragmented Boundaries. In Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Montréal, QC, Canada, 23–25 July 2013; ACM: New York, NY, USA, 2013; pp. 241–250.
12. Czyzowicz, J.; Gąsieniec, L.; Kosowski, A.; Kranakis, E. Boundary Patrolling by Mobile Agents with Distinct Maximal Speeds. In *Algorithms—ESA 2011, Proceedings of the 19th Annual European Symposium, Saarbrücken, Germany, 5–9 September 2011*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2011; Volume 6942, pp. 701–712.
13. Chuangpishit, H.; Czyzowicz, J.; Gąsieniec, L.; Georgiou, K.; Jurdzinski, T.; Kranakis, E. Patrolling a Path Connecting a Set of Points with Unbalanced Frequencies of Visits. In *SOFSEM 2018: Theory and Practice of Computer Science, Proceedings of the 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, 29 January–2 February 2018*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2018; Volume 10706, pp. 367–380.
14. Serafini, P.; Ukovich, W. A Mathematical Model for Periodic Scheduling Problems. *SIAM J. Discret. Math.* **1989**, *2*, 550–581. [[CrossRef](#)]
15. Chan, M.Y.; Chin, F.Y.L. General schedulers for the pinwheel problem based on double-integer reduction. *IEEE Trans. Comput.* **1992**, *41*, 755–768. [[CrossRef](#)]
16. Chan, M.Y.; Chin, F.Y.L. Schedulers for larger classes of pinwheel instances. *Algorithmica* **1993**, *9*, 425–462. [[CrossRef](#)]
17. Hsueh, C.; Lin, K. An Optimal Pinwheel Scheduler Using the Single-number Reduction Technique. In Proceedings of the 17th IEEE Real-Time Systems Symposium, Washington, DC, USA, 4–6 December 1996; pp. 196–205.
18. Holte, R.; Rosier, L.; Tulchinsky, I.; Varvel, D. Pinwheel scheduling with two distinct numbers. *Theor. Comput. Sci.* **1992**, *100*, 105–135. [[CrossRef](#)]
19. Lin, S.S.; Lin, K.J. A Pinwheel Scheduler for Three Distinct Numbers with a Tight Schedulability Bound. *Algorithmica* **1997**, *19*, 411–426. [[CrossRef](#)]

20. Romer, T.H.; Rosier, L.E. An algorithm reminiscent of euclidean-gcd for computing a function related to pinwheel scheduling. *Algorithmica* **1997**, *17*, 1–10. [[CrossRef](#)]
21. Baruah, S.K.; Lin, S.-S. Pfair scheduling of generalized pinwheel task systems. *IEEE Trans. Comput.* **1998**, *47*, 812–816. [[CrossRef](#)]
22. Baruah, S.K.; Cohen, N.K.; Plaxton, C.G.; Varvel, D.A. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica* **1996**, *15*, 600–625. [[CrossRef](#)]
23. Mok, A.; Rosier, L.; Tulchinski, I.; Varvel, D. Algorithms and complexity of the periodic maintenance problem. *Microprocess. Microprogram.* **1989**, *27*, 657–664. [[CrossRef](#)]
24. Anily, S.; Glass, C.A.; Hassin, R. The scheduling of maintenance service. *Discret. Appl. Math.* **1998**, *82*, 27–42. [[CrossRef](#)]
25. Anily, S.; Glass, C.A.; Hassin, R. Scheduling maintenance services to three machines. *Ann. Oper. Res.* **1999**, *86*, 375–391. [[CrossRef](#)]
26. Bender, M.A.; Fekete,S.P.; Kröller, A.; Mitchell, J.S.B.; Liberatore, V.; Polishchuk, V.;Suomela, J. The Minimum Backlog Problem. *Theor. Comput. Sci.* **2015**, *605*, 51–61. [[CrossRef](#)]
27. Bodlaender, M.H.L.; Hurkens, C.A.J.; Kusters, V.J.J.; Staals, F.; Woeginger, G.J.; Zantema, H. Cinderella versus the Wicked Stepmother. In *TCS 2012: Theoretical Computer Science, Proceedings of the IFIP Theoretical Computer Science Conference, Amsterdam, The Netherlands, 26–28 September 2012*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2012, Volume 6942, pp. 57–71.
28. Chrobak, M.; Csirik, J.; Imreh, C.; Noga, J.; Sgall, J.; Woeginger, G.J. The Buffer Minimization Problem for Multiprocessor Scheduling with Conflicts. In *Automata, Languages and Programming, Proceedings of the 28th International Colloquium on Automata, Languages, and Programming, Crete, Greece, 8–12 July 2001*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2001, Volume 2076, pp. 862–874.
29. D’Emidio, M.; Di Stefano, G.; Navarra, A. Priority Scheduling in the Bamboo Garden Trimming Problem. In *SOFSEM 2019: Theory and Practice of Computer Science, Proceedings of the 45th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, 27–30 January 2019*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2019; Volume 11376; pp. 136–149.
30. Hardy, G.H.; Ramanujan, S. Asymptotic formulas in combinatorial analysis. *Proc. Lond. Math. Soc.* **1918**, *17*, 75–115. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).