

Perpetual maintenance of machines with different urgency requirements^{*}

Leszek Gąsieniec¹, Ralf Klasing², Christos Levcopoulos³
Andrzej Lingas³, Jie Min¹, and Tomasz Radzik⁴

¹ Department of Computer Science, University of Liverpool, Liverpool, UK,
{l.a.gasieniec, J.Min2}@liverpool.ac.uk.

² CNRS, LaBRI, Université de Bordeaux, France, ralf.klasing@labri.fr.

³ Department of Computer Science, Lund University, Lund, Sweden,
{christos.levcopoulos, andrzej.lingas}@cs.lth.se.

⁴ Department of Informatics, King's College London, London, UK,
tomasz.radzik@kcl.ac.uk.

Abstract. A garden G is populated by $n \geq 1$ bamboos b_1, b_2, \dots, b_n with the respective daily growth rates $h_1 \geq h_2 \geq \dots \geq h_n$. It is assumed that the initial heights of bamboos are zero. The robotic gardener maintaining the garden regularly attends bamboos and trims them to height zero according to some schedule. The *Bamboo Garden Trimming Problem* (BGT) is to design a perpetual schedule of cuts to maintain the elevation of the bamboo garden as low as possible. The bamboo garden is a metaphor for a collection of machines which have to be serviced, with different frequencies, by a robot which can service only one machine at a time. The objective is to design a perpetual schedule of servicing which minimizes the maximum (weighted) waiting time for servicing.

We consider two variants of BGT. In *discrete* BGT the robot trims only one bamboo at the end of each day. In *continuous* BGT the bamboos can be cut at any time, however, the robot needs time to move from one bamboo to the next.

For discrete BGT, we show a simple 4-approximation algorithm and, by exploiting relationship between BGT and the classical Pinwheel scheduling problem, we derive a 2-approximation algorithm for the general case and a tighter approximation when the growth rates are balanced. A by-product of this last approximation algorithm is that it settles one of the conjectures about the Pinwheel problem. For continuous BGT, we propose approximation algorithms which achieve approximation ratios $O(\log(h_1/h_n))$ and $O(\log n)$.

^{*} L. Gąsieniec and J. Min's work was partially supported by Network Sciences and Technologies (NeST) at University of Liverpool. This study has been carried out in the frame of "the Investments for the future" Programme IdEx Bordeaux – SysNum (ANR-10-IDEX-03-02). T. Radzik's work was supported in part by EPSRC grant EP/M005038/1, "Randomized algorithms for computer networks." Part of this work was done while T. Radzik was visiting the LaBRI as a guest professor of the University of Bordeaux. An extended abstract of this work appeared in Proc. 43rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2017), LNCS 10139, pp. 229–240, Springer (2017) [19].

1 Introduction

We consider a perpetual scheduling problem in which n (possibly virtual) machines need to be attended (serviced) with *known* but possibly different frequencies, i.e. some machines need to be attended more often than others. The frequencies of attending individual machines are specified as positive weights h_1, h_2, \dots, h_n and the objective is to design a perpetual schedule of attending the machines which minimizes the maximum weighted time any individual machine waits for the next service. The same optimization problem arises when a data stream keeps filling a collection of n buffers according to a known distribution: buffer i receives h_i units of data in each unit of time. The objective is to design a perpetual schedule of emptying the buffers which minimizes the maximum occupancy of any individual buffer.

We model such perpetual scheduling problems using the following metaphor of the *Bamboo Garden Trimming (BGT) Problem*. A collection (garden) of n bamboos b_1, b_2, \dots, b_n with known respective daily growth rates h_1, h_2, \dots, h_n . We assume that these growth rates are already arranged into a non-increasing sequence: $h_1 \geq h_2 \geq \dots \geq h_n > 0$. Initially the height of each bamboo is set to zero. The robotic gardener maintaining the garden trims bamboos to height zero according to some schedule. The height of a bamboo b_i at time $t \geq 0$ is equal to $(t - t')h_i$, where t' is the last time when this bamboo was trimmed, or $t' = 0$, if it has never been trimmed by time t . The main task of the BGT problem is to design a perpetual schedule of cuts to keep the highest bamboo in the garden as low as possible, while complying with some specified constraints on the timing of cutting. The basic constraints considered in this paper are that the gardener can cut only one (arbitrary) 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 in the current round (at the end of the current day), then the action of actual trimming is instantaneous.

Referring back to the two scheduling problems mentioned earlier, the heights of the growing bamboos would represent the weighted times the machines wait for the next service, or the current occupancy of the data buffers. The action of cutting a bamboo b_i at the end of the current day represents attending machine i or emptying buffer i in the current time slot. Other problems which can be modeled by BGT include the perpetual testing of virtual machines in cloud systems [1]. In such systems frequency in which virtual machines are tested for undesirable symptoms vary depending on the importance of dedicated cloud operational mechanisms.

BGT is also a natural extension of several classical algorithmic problems with the focus on *monitoring* and *mobility*, including the *Art Gallery Problem* [11] and its dynamic extension called the *k-Watchmen Problem* [30]. In a more recent work on *fence patrolling* [14, 15] the studies focus on monitoring vital (possibly disconnected) parts of a linear environment where each point is expected to be attended with the same frequency. The authors of [16] study monitoring linear environments by robots prone to faults. Our paper focuses on the case where each vital part of the environment has its own, possibly unique urgency

factor. This makes it related to *periodic scheduling* [29], a series of papers on the *Pinwheel* problems [9, 10, 20] including the *periodic Pinwheel* problem [21, 25] and the *Pinwheel scheduling* problem [28], as well as the concept of *P-fairness* in sharing multiple copies of some resource among various tasks [3, 4].

We consider two variants of the BGT problem. The constraint that only one bamboo is cut at the end of each day (round) defines *discrete* BGT. In the second variant, *continuous* BGT, we assume that for any two bamboos b_i and b_j , we know the time $t_{i,j} > 0$ (which may be fractional) that the robot needs to relocate from b_i to b_j . In this variant the time when the next bamboo is trimmed depends on how far that bamboo is from the bamboo which has just been trimmed. As in discrete BGT, when the robot arrives at the bamboo to trim it, the actual action of trimming is instantaneous. In this paper we consider symmetric travel times (that is, $t_{i,j} = t_{j,i}$) and assume that the robot travels always along the fastest route, so the travel times satisfy the triangle inequality. We also assume that the robot is initially at the location of b_1 . Discrete BGT is the special case of continuous BGT when all travel times $t_{i,j}$, for $i \neq j$, are the same, while metric TSP is the special case of continuous BGT when all growth rates h_i are the same. Previous work on problems of similar nature as the continuous BGT includes recent work on patrolling [14–16, 24].

In both cases, discrete and continuous, we consider algorithms \mathcal{A} which for an input instance I (of the form $\langle h_i : 1 \leq i \leq n \rangle$ in the discrete case and $[\langle h_i : 1 \leq i \leq n \rangle, \langle t_{i,j} : 1 \leq i, j \leq n \rangle]$ in the continuous case) produce a perpetual (trimming) schedule $\mathcal{A}(I)$, that is, a sequence of indices of bamboos (i_1, i_2, \dots) which defines the order in which the bamboos are trimmed. We are mainly interested in the *approximation ratios* of such algorithms, which are defined in the usual way. For an input instance I and a trimming schedule \mathcal{S} for I , let $MH(\mathcal{S})$ denote the supremum of the heights of bamboos over all times $t \geq 0$ when the trimming proceeds according to schedule \mathcal{S} , and let $OPT(I)$ denote the infimum of $MH(\mathcal{S})$ over all schedules \mathcal{S} for I . The approximation ratio of a schedule \mathcal{S} is defined as $MH(\mathcal{S})/OPT(I)$ and the approximation ratio of an algorithm \mathcal{A} is the supremum of $MH(\mathcal{A}(I))/OPT(I)$ over all input instances I .

Regarding the time complexity of BGT algorithms, we aim at polynomial preprocessing time followed by computation of the consecutive indices of the schedule in poly-logarithmic time per one index. We will call algorithms with such performance simply polynomial-time (BGT) algorithms. The computational complexity of continuous BGT is related to the complexity of TSP, as the latter is a special case of the former. To discuss computational complexity of discrete BGT, we introduce first a lower bound on the height of schedules.

For each instance I of discrete BGT with the sum of the growth rates $H = h_1 + h_2 + \dots + h_n$, a simple and natural lower bound on the maximum height of a bamboo in any schedule is $OPT(I) \geq H$. Indeed, while the heights of all bamboos are at most $H' < H$, then during each day the *total height* of all bamboos, that is, the sum of the current heights of all bamboos, increases at least by $H - H' > 0$ (the total growth over all bamboos is H but only one bamboo, of height at most H' , is cut). Thus on some day within the first $\lfloor nH'/(H - H') \rfloor + 1$ days the total

height of the bamboos must exceed nH' , so the height of one of the bamboos must exceed H' . Observe also that it cannot happen that the maximum height of a bamboo approaches H but never reaches H , because there are only finitely many possible heights of bamboos which are less than H .

There are instances with $OPT(I) = H$. For example, for the input instance $I = (1/2, 1/4, 1/4)$, all bamboos are kept within the $H = 1$ height by the schedule with period (b_1, b_2, b_1, b_3) . An example of an input instance with $OPT(I) > H$ is $I = (7/15, 1/3, 1/5)$, for which $H = 1$ but $OPT(I) = 4/3$. For this instance, the schedule with period $(b_1, b_2, b_1, b_2, b_1, b_3)$ does not let any bamboo grow above the height $4/3$. On the other hand, a schedule which keeps the heights of b_1 and b_2 strictly lower than $4/3$ must cut b_1 every other day, implying that b_2 must also be cut every other day (after the initial couple of days). Thus, after the initial couple of days, there are no further days available to cut b_3 , so its height grows to infinity.

The computational complexity of discrete BGT is related to the computational complexity of the Pinwheel problem [20], which can be viewed as a special case of discrete BGT. The Pinwheel complexity results presented in [20] imply that for a given $K \geq H$, if there is a schedule with height at most K , then there is a cyclic schedule with height at most K , but the shortest such schedule can have exponential length.¹ This implies that the decision version of discrete BGT can be solved by considering all cyclic schedules of up to exponential length, and this can be implemented in PSPACE. Further from [20], while the cyclic schedules of height H can also have exponential length, they have concise polynomial-size representations, and there is a polynomial-time algorithm for checking if a given concise representation of a cyclic schedule of height H is valid. This implies that the restricted decision version of discrete BGT which asks if there is a schedule of height H is in NP. There have been some further complexity results for the Pinwheel problem, with direct implications for the complexity of discrete BGT (see [2]), but all in all, the complexity of the Pinwheel problem remains a long-standing open question.

In related research on minimizing the maximum occupancy of a buffer in a system of n buffers, the usual setting is a game between the player and the adversary [5, 7, 12]. The adversary decides how the fixed total increase of data in each round is distributed among the buffers and tries to maximize the maximum occupancy of a buffer. The player decides which buffer (or buffers, depending on the variant of the problem) should be emptied next and tries to minimize the maximum buffer size. The upper bounds developed in this more general context can be translated into upper bounds for our BGT problems, but our aim is to derive tighter bounds for the case when the rates of growth of the occupancy of buffers, or the rates of growth of bamboos in our language, are fixed and known.

Probably the most natural strategy to keep the elevation of the bamboo garden low is the greedy approach of always moving next to, and cutting, the currently highest bamboo. This approach, called *Reduce-Max*, was considered

¹ Exponential in the size of the input, assuming that the growth rates are rational numbers given as pairs of integers.

recently in the context of periodic testing of virtual machines in cloud systems [1], and was also studied in the adversarial setting of the buffer minimization problems mentioned above. The results presented in [7] imply a tight upper bound of $H \cdot (H_{n-1} + 1) = \Theta(H \log n)$ on $MH(\mathcal{S})$ for schedules \mathcal{S} produced by Reduce-Max for a variant of the discrete BGT with the adversary which in each round arbitrarily distributes the total daily growth of H among the bamboos. Here $H_k = \sum_{i=1}^k \frac{1}{i} = \Theta(\log k)$ is the k -th harmonic number. While this $O(H \log n)$ upper bound applies obviously also to our non-adversarial discrete BGT, when the growth rates are fixed, it was a long standing open question whether there were instances for which Reduce-Max lets some bamboos grow to heights $\Omega(H \log n)$, or even to heights $\omega(H)$. The experimental work presented in [1] pointed towards a conjecture that Reduce-Max keeps the maximum bamboo height within $O(H)$, and the question has been finally recently answered in [6], where a bound of $9H$ on the maximum bamboo height under the Reduce-Max algorithm was proven.

Our contributions and related work. In the preliminary version [19] of this paper, we introduced a modification of Reduce-Max, which we called Reduce-Fastest, to show the first simple greedy algorithm achieving constant approximation ratio. Reduce-Fastest works in the following way. Keep track of the “tall” bamboos, defined as having the current height at least $x \cdot H$ (where x is a parameter of the algorithm), and cut in each step the tall bamboo with the highest growth rate (no cutting, if there is no tall bamboo). In Section 3, we present a detailed proof that the approximation ratio of Reduce-Fastest for $x = 2$ is at most 4. Subsequently to our work, it has been proven in [6] that using the Reduce-Fastest strategy an approximation ratio of 2.62 can be achieved, and that, as mentioned above, the Reduce-Max strategy also achieves constant approximation ratio. [6] also presents efficient implementations of these two heuristics.

We refer informally to BGT algorithms like Reduce-Max and Reduce-Fastest as *online scheduling*. These algorithms are based on simple greedy strategies, the trimming schedule is revealed while the cutting progresses, and the whole cutting process would naturally adapt to changing growth rates. An alternative *offline scheduling* pre-computes the whole (cyclic) schedule. This approach would sacrifice the flexibility offered by simple greedy strategies but hopefully would give better approximation ratios. Indeed, using the Pinwheel results given in [20], one can easily obtain a 2-approximation (offline) scheduling algorithm for discrete BGT. An efficient implementation of this algorithm was developed in [6]. Recently, [31] has shown an offline scheduling algorithm, also based on Pinwheel results, which guarantees 12/7-approximation.

In Section 4 we present our main approximation algorithm for discrete BGT, which is an offline algorithm derived by further exploration of the relation between discrete BGT and the Pinwheel problem and has approximation ratio $(1 + O(\sqrt{h_1/H}))$. The benefits of the relation between the discrete BGT and Pinwheel problems extend both ways. On the one hand, our approximation algorithm uses properties of the Pinwheel problem. On the other hand, the approxi-

mation ratio which we achieve settles one of the conjectures about the Pinwheel problem as explained in Section 2.

In Section 5, we show algorithms for continuous BGT with approximation ratios $O(\log(h_1/h_n))$ and $O(\log n)$.² We also discuss how tight these approximation ratios are. We show instances of continuous BGT such that for any schedule the maximum bamboo height is greater by a $\Theta(\log n)$ factor than the lower bounds which we use in the analysis of approximation ratios. Thus for these input instances our $O(\log n)$ -approximation algorithm computes in fact schedules with constant approximation ratios. We also show instances for which this algorithm computes $\Theta(\log n)$ -approximate schedules.

2 Discrete BGT problem and Pinwheel

The input of the Pinwheel problem is a sequence $V = (f_1, f_2, \dots, f_n)$ of integers $2 \leq f_1 \leq f_2 \leq \dots \leq f_n$ called (*pinwheel*) *frequencies*. The objective is to specify an infinite sequence S of indices drawn from the set $1, 2, \dots, n$ such that for each index i , any sub-sequence of f_i consecutive elements in S includes at least one index i , or to establish that such a sequence does not exist. A sequence S with this property is called a schedule of V , and if such a sequence does not exist, then we say that the sequence of frequencies V is not feasible or that it cannot be scheduled. The Pinwheel problem is a special case of discrete BGT: an input instance (f_1, f_2, \dots, f_n) of Pinwheel is feasible if, and only if, the optimal (minimum) height for the input instance $(1/f_1, 1/f_2, \dots, 1/f_n)$ of discrete BGT is at most 1.

The Pinwheel problem was introduced in [20], where some complexity results were presented and some classes of feasible sequences of frequencies were established. It is easy to see that it is not possible to schedule any instance V whose *density* $D(V) \equiv \sum_{i=1}^n 1/f_i$ is greater than 1, since in any feasible schedule each frequency f_i takes at least $1/f_i$ fraction of the slots.³ (This upper bound of 1 on the density of a feasible instance of the Pinwheel problem is a special case of the lower bound of H on the maximum height for an instance of discrete BGT.) The sequence of frequencies $(2, 4, 4)$ is an example of a feasible instance of the Pinwheel problem with density 1. On the other hand, the input instances $(2, 3, M)$, where M is an arbitrarily large integer, show that there are instances with densities arbitrarily close to $5/6$ which are not feasible. One of the conjectures for the Pinwheel problem, which remains open, is that $5/6$ is the universal threshold guaranteeing feasibility of input instances. That is, it has been conjectured (ever since the pinwheel problem was introduced) that any instance with density at most $5/6$ can be scheduled. The current best proven bound is $3/4$ [18].

² As Metric TSP is a subproblem of continuous BGT, it is NP-hard to approximate continuous BGT with a factor better than $123/122$ [23].

³ More precisely, in each prefix of length $T - f_n$ of a feasible schedule, where T is arbitrarily large, each frequency f_i must take at least $(1/f_i)(T - f_i)$ slots, implying $\sum_{i=1}^n 1/f_i \leq 1$.

Our work is related to another conjecture for the Pinwheel problem, made by Chan and Chin [9], that when the first frequency f_1 keeps increasing, then the density threshold guaranteeing feasibility keeps increasing to 1. To be more precise, let \mathcal{V} , $\mathcal{V}_{yes} \subseteq \mathcal{V}$ and $\mathcal{V}(f, \Delta) \subseteq \mathcal{V}$ denote the set of all instances V of the Pinwheel problem with density $D(V) \leq 1$, the set of all feasible instances and the set of instances with $f_1 = f$ and density $D(V) \leq \Delta$, respectively. Define $d(f) \equiv \sup\{\Delta : \mathcal{V}(f, \Delta) \subseteq \mathcal{V}_{yes}\}$ as the density threshold guaranteeing feasibility of instances with the first frequency equal to f . That is, each input instance $(f_1 = f, f_2, f_3, \dots)$ with density less than $d(f)$ is feasible, while for each $\epsilon > 0$, there is an infeasible instance with density less than $d(f) + \epsilon$. Chan and Chin [9, 10] conjecture that $\lim_{f \rightarrow \infty} d(f) = 1$, consider a number of heuristics for the Pinwheel problem (referred to as *schedulers*) and analyze the guarantee density threshold

$$d_{\mathcal{A}}(f) \equiv \sup\{D : \text{heuristic } \mathcal{A} \text{ schedules each instance in } \mathcal{V}(f, D)\}$$

for each considered heuristic \mathcal{A} . They derive lower bounds $\ell_{\mathcal{A}}(f)$ on the values $d_{\mathcal{A}}(f)$, but for each of their lower bounds, $\lim_{f \rightarrow \infty} \ell_{\mathcal{A}}(f)$ is strictly less than 1, which leaves possibility that $\lim_{f \rightarrow \infty} d_{\mathcal{A}}(f)$ is also strictly less than 1.⁴ Thus [9, 10] left open the question of designing an algorithm \mathcal{A} for which $\lim_{f \rightarrow \infty} d_{\mathcal{A}}(f) = 1$, and there have not been other results in this direction prior to our work. Such an algorithm would immediately imply that $\lim_{f \rightarrow \infty} d(f) = 1$.

Our $(1 + O(\sqrt{h_1/H}))$ -approximate polynomial-time algorithm for discrete BGT applied to input instances $(1/f_1, 1/f_2, \dots, 1/f_n)$ is a polynomial-time scheduler for Pinwheel input instances (f_1, f_2, \dots, f_n) with the guarantee density threshold $1 - O(\sqrt{1/f_1})$. This threshold tends to 1 with increasing f_1 , proving the conjecture that $\lim_{f \rightarrow \infty} d(f) = 1$. To see that this Pinwheel scheduler has indeed the guarantee density threshold $1 - O(\sqrt{1/f_1})$, let C be a constant such that the approximation ratio of our algorithm for discrete BGT is at most $1 + C\sqrt{h_1/H}$. If the density $D(V)$ of a Pinwheel instance $V = (f_1, f_2, \dots, f_n)$ is at most $1 - C/\sqrt{f_1}$, then the BGT schedule computed for the input $(1/f_1, 1/f_2, \dots, 1/f_n)$ does not let the height of any bamboo go above (note that $H = D(V)$):

$$\begin{aligned} D(V) \left((1 + C\sqrt{(1/f_1)/D(V)}) \right) &= D(V) + C\sqrt{D(V)/f_1} \\ &\leq 1 - C/\sqrt{f_1} + C\sqrt{D(V)/f_1} \leq 1. \end{aligned}$$

Thus the computed schedule is a feasible schedule for the Pinwheel instance.

3 Discrete BGT by online scheduling

We consider two types of algorithms for the discrete variant of BGT. An *online* algorithm keeps track of the current heights of the bamboos and decides which

⁴ Chan and Chin [9, 10] showed that for some algorithms which they considered $\lim_{f \rightarrow \infty} d_{\mathcal{A}}(f)$ is actually strictly less than 1. For the other algorithms, they left unanswered the question whether $\lim_{f \rightarrow \infty} d_{\mathcal{A}}(f) = 1$.

bamboo should be cut next on the basis of simple queries like “what is the tallest bamboo?” (as in Reduce-Max algorithm), or “what is the fastest growing bamboo with the height above some threshold?” (as below in Reduce-Fastest algorithm). Such queries can be answered without knowing the whole distribution of growth rates. An *offline* algorithm determines which bamboo is to be trimmed during a particular round by first preprocessing the input (the sequence of the daily growth rates) into a representation of the full perpetual schedule. After the preprocessing, the schedule is produced with some small computational effort per each step (that is, per each cutting decision).

Online scheduling is more flexible since it can adapt, if the growth rates change. On the downside, the performance analysis of online scheduling is more complex and the approximation bounds tend to be weaker. Offline scheduling is less flexible since changes of the growth rates require re-doing the preprocessing, but leads to stronger approximation bounds. We note that our online versus offline characterization of BGT algorithms does not refer to the availability of input (in both cases the whole input is known in advance) but only indicates the general nature of algorithms.

We present in this section a simple constant-ratio online scheduling algorithm, which is obtained by the following modification of Reduce-Max. We cut next the fastest growing ‘tall’ bamboo, where tall bamboos are the bamboos with the current heights at least $x \cdot H$, for a given parameter $x > 1$. We call this algorithm Reduce-Fastest(x) and show the following approximation bound.

Theorem 1. *Reduce-Fastest(2) is a 4-approximation algorithm for discrete BGT.*

Proof. We consider the execution of Reduce-Fastest(x) for a fixed value $x > 1$. Without loss of generality, we assume that if there are two or more bamboos with the same fastest growth rate among the bamboos with the current height at least $x \cdot H$, then Reduce-Fastest chooses for trimming the bamboo with the smallest index. Thus the height of bamboo b_1 is always less than $xH + h_1 \leq (x + 1)H$.

We consider now a bamboo b_i , for an arbitrary $2 \leq i \leq n$, and assume that it reaches the height at least $C \cdot H$ for some constant $C \geq x + 1$. At any time the heights of bamboos belong to two disjoint regions: the lower region $[0, xH)$ and the upper region $[xH, \infty)$. At some point bamboo b_i must stay in the upper region for at least $t = \lfloor (C - x)H/h_i \rfloor$ consecutive rounds to reach the height $C \cdot H$. That is, during each of these t rounds, b_i has height at least xH but is not cut.

We consider a period of t consecutive rounds when bamboo b_i remains in the upper region. At each of these rounds, trimming of bamboo b_i is blocked by trimming of another bamboo b_j for some $j < i$, by the definition of Reduce-Fastest(2). The number of times when the same bamboo b_j can block bamboo b_i during this period is at most $t_j = 1 + \lfloor t/f_j \rfloor$, where $f_j = \lceil xH/h_j \rceil$ is the number of rounds

bamboo b_j needs to climb back to the upper region after trimming. Thus

$$t \leq \sum_{j=1}^{i-1} t_j = \sum_{j=1}^{i-1} \left(1 + \left\lfloor \frac{t}{f_j} \right\rfloor\right) \leq t \left(\frac{i-1}{t} + \sum_{j=1}^{i-1} \frac{1}{f_j} \right),$$

hence

$$\frac{i-1}{t} + \sum_{j=1}^{i-1} \frac{1}{f_j} \geq 1. \quad (1)$$

Since $h_i \leq H/i$ and $C - x \geq 1$, we have

$$t = \left\lfloor \frac{(C-x)H}{h_i} \right\rfloor > (C-x)i - 1 \geq (C-x)(i-1). \quad (2)$$

Inequalities (1) and (2) imply

$$1 \leq \frac{i-1}{t} + \sum_{j=1}^{i-1} \frac{1}{f_j} < \frac{1}{C-x} + \sum_{j=1}^{i-1} \frac{h_j}{xH} < \frac{1}{C-x} + \frac{1}{x}.$$

We rewrite $1 < 1/(C-x) + 1/x$ to

$$C < 2 + \left((x-1) + \frac{1}{x-1} \right),$$

and observe that the right-hand side above is minimized for $x = 2$ and for this value gives $C < 4$. Thus during the execution of Reduce-Fastest(2), no bamboo can reach the height $4H$, so the approximation ratio of this algorithm is at most 4. \square

In addition to approximation guarantees, the time complexity of BGT heuristics is also important. Efficient implementations of Reduce-Max and Reduce-Fastest were presented in [6]. Below we briefly indicate the main ideas.

For Reduce-Max, use a parametric priority queue, where keys are linear functions $at + b$ of a single parameter t . Maintain such a queue containing the value-key entries $(i, h_i t - h_i t_i)$, where t_i is the last step when bamboo b_i was cut. Thus $h_i(t - t_i)$ is the height of b_i at step $t \geq t_i$. In the current step T , delete from the queue the entry with the maximum key for $t = T$ (using the delete-max operation of the parametric priority queue) obtaining the current tallest bamboo b_i . Cut this bamboo and insert to the queue the entry $(i, h_i T - h_i T)$. The running time of one step is $O(\log^2 n)$, using, for example, the implementations of parametric priority queues proposed in [22, 26].

For Reduce-Fastest(x), let Z be the set of the current tall bamboos, that is, the bamboos which have reached height x by the current step T , and let Y_t , for $t \geq T$, be the set of the bamboos which will reach height x at step t . Maintain set Z and all non-empty sets Y_t as priority queues with value-key entries (i, h_i) of bamboo indices and heights. In addition, keep the pairs (t, Y_t) , for non-empty

sets Y_t , in a dictionary (t is the key). During the current step T : remove Y_T from the dictionary and merge it into Z (if Y_T exists), remove from Z the pair (i, h_i) with the greatest h_i (the fastest growing tall bamboo), cut bamboo b_i and insert (i, h_i) to the set Y_t for $t = T + \lceil x/h_i \rceil$ (this t is the next step when b_i reaches height x). The running time of one step is $O(\log n)$, using a mergeable priority queue with $O(\log n)$ worst-case running time per each operation (e.g. [8]).

4 Discrete BGT by offline scheduling

In this section we focus on offline scheduling which permits tighter approximation results. These results are achieved by exploring the relationship between BGT and the Pinwheel scheduling problem. Some known facts about Pinwheel scheduling give immediately a 2-approximation BGT algorithm. Our main result in this section is a $(1 + O(\sqrt{h_1/H}))$ -approximation algorithm. One of the consequences of this approximation algorithm is that it settles the conjecture made for the Pinwheel problem that if the first (smallest) frequency keeps increasing, the density threshold which guarantees feasibility increases to 1 (cf. Section 2).

4.1 Reducing discrete BGT to Pinwheel scheduling

We use the notation for the Pinwheel problem introduced in Section 2. Each feasible input sequence of frequencies $f_1 \leq f_2 \leq \dots \leq f_n$ has the density $D = \sum_{i=1}^n 1/f_i$ at most one. We also know that any instance with density at most $3/4$ is feasible [18], and it is conjectured that in fact any instance with density at most $5/6$ is feasible. To see the relationship between the BGT and the Pinwheel problem, define for a BGT input instance $I = (h_1 \geq h_2 \geq \dots \geq h_n > 0)$ the sequence of frequencies $f'_i = H/h_i$, $i = 1, 2, \dots, n$. This sequence $(f'_1, f'_2, \dots, f'_n)$ is a *pseudo-instance* of Pinwheel (pseudo, since these frequencies are rational numbers rather than integers) with density:

$$D' = \sum_{i=1}^n \frac{1}{f'_i} = \sum_{i=1}^n \frac{h_i}{H} = 1.$$

We multiply the frequencies f'_i by $1 + \delta$ to obtain another pseudo-instance $f''_i = f'_i(1 + \delta)$, $i = 1, 2, \dots, n$, with the density reduced to $1/(1 + \delta) < 1$, where $\delta > 0$ is a suitable parameter. Finally, we obtain a (proper) instance $V(I, \delta) = (f_1, f_2, \dots, f_n)$ of the Pinwheel problem by reducing each frequency f''_i to an integer $f_i \leq f''_i$. Thus we require that the integer frequencies (f_i) are not greater than the rational frequencies (f''_i) , but we do not specify at this point their exact values, leaving this to concrete algorithms. Reducing frequencies (f''_i) to (f_i) increases the density of the sequence. The room for this increase of the density was made by the initial decrease of the density to $1/(1 + \delta)$.

Lemma 1. *If I is an instance of BGT, $\delta > 0$ and an instance $V(I, \delta)$ of the Pinwheel problem is feasible, then a feasible schedule for this Pinwheel instance $V(I, \delta)$ is a $(1 + \delta)$ -approximation schedule for the BGT instance I .*

Proof. In a feasible schedule for $V(I, \delta)$, two consecutive occurrences of an index i are at most $f_i \leq H(1 + \delta)/h_i$ slots apart. This means that if this schedule is used for the BGT instance I , then the height of b_i is never greater than $h_i \cdot f_i \leq H(1 + \delta)$. \square

In view of Lemma 1, the goal is to get a *feasible* instance $V(I, \delta)$ of Pinwheel for as small value of δ as possible. We also want to be able to compute efficiently a feasible schedule for $V(I, \delta)$, if one exists. By decreasing the rational frequencies f_i'' to integer frequencies f_i , we increase the density of the Pinwheel instance, making it possibly harder to schedule. Thus we should aim at decreasing the frequencies f_i'' only as much as necessary. However, simply rounding down the frequencies f_i'' to the nearest integers might not be the best way since the integer frequencies obtained that way might not be sufficiently “regular” to imply a feasible schedule.

A 2-approximation algorithm. To give a simple illustration how Lemma 1 can be used, we refer to the result from [20] which says that any instance of Pinwheel with frequencies being powers of 2 and the density at most 1 can be scheduled and a feasible schedule can be easily computed. For an instance I of BGT, we take the instance $V(I, 1)$ where the frequencies f_i'' are rounded down to the powers of 2. Multiplying first the frequencies by 2 decreases the density to $1/2$. The subsequent rounding down to the powers of 2 decreases each frequency less than by half, so the density of the instance increases less than twice. Thus the final instance $V(I, 1)$ of Pinwheel has the density less than 1 and all frequencies are powers of 2, so it can be scheduled and, by Lemma 1, its feasible schedule is a 2-approximate schedule for the original BGT instance I .

It was shown in [18] that every instance of Pinwheel with density not greater than $3/4$ is feasible and, tracing the proof given in [18], one can obtain an efficient scheduling algorithm for such instances. For a BTG instance I , the Pinwheel instance $V(I, \delta) = (f_1, f_2, \dots, f_n)$, where $\delta = 1/3 + h_1/H$ and $f_i = \lfloor (1 + \delta)H/h_i \rfloor$ has density less than $3/4$ since

$$\frac{1}{f_i} < \frac{1}{(1 + \delta)H/h_i - 1} = \frac{h_i}{H} \cdot \frac{1}{1 + \delta - h_i/H} \leq \frac{h_i}{H} \cdot \frac{1}{1 + \delta - h_1/H} = \frac{3}{4} \cdot \frac{h_i}{H}.$$

Thus Lemma 1 and Pinwheel schedules for instances with density at most $3/4$ give $(4/3 + h_1/H)$ -approximate schedules for the BGT problem. When h_1/H decreases to 0, then this approach gives approximation ratios decreasing to $4/3$, improving on the 2-approximation. We are, however, looking for an algorithm which would compute BGT schedules with approximation ratio decreasing to 1 when h_1/H decreases to 0.

4.2 A $(1 + O(\sqrt{h_1/H}))$ -approximation algorithm

Our approach to a better BGT approximation is again through powers-of-two Pinwheel instances, as in the 2-approximation algorithm, but now we derive an

appropriate powers-of-two instance through a process of gradual transformations. We start with greater granularity of frequencies than powers of two by reducing the rational frequencies $f_i'' = H(1 + \delta)/h_i$ to the closest values of the form $2^k(1 + j/C)$, where k is an integer, $C = 2^q$ for some integer $q \geq 0$, and j is an integer in $[0, C)$. We set the parameter q in such a way that we always have $q \leq k$, so the frequencies $2^k(1 + j/C)$ are integral, forming a proper instance of Pinwheel. To show that the obtained instance $V(I, \delta)$ of Pinwheel is feasible, for a suitable choice of δ , and to construct a schedule for this instance, we use the following two observations.

Observation 1. Let V be an instance of Pinwheel which has two equal even frequencies $f_i = f_j = 2f$. If the instance V' obtained from V by replacing these two frequencies with one frequency f is feasible, then so is instance V . Note that such updates of a sequence of frequencies do not change its density. We obtain a schedule for V from a schedule for V' by replacing the occurrences of the frequency f alternately with the frequencies f_i and f_j . In our algorithm we will be replacing pairs of equal frequencies $2^k(1 + j/C)$ with one frequency $2^{k-1}(1 + j/C)$.

Observation 2. Generalizing the previous observation, let V be an instance of Pinwheel which has m equal frequencies $f_{i_1} = f_{i_2} = \dots = f_{i_m} = mf$, where f is an integer. If the instance V' obtained from V by replacing these m frequencies with one frequency f is feasible, then so is instance V and a schedule for V can be easily obtained from a schedule for V' (in the schedule for V' , for each $i \geq 0$ and $1 \leq q \leq m$, replace the $(im + q)$ -th occurrence of f with f_{i_q}). As before, such updates of a sequence of frequencies do not change its density. In our algorithm we will be combining $m_j = C + j$ frequencies $2^k(1 + \frac{j}{C})$ into one frequency $2^k(1 + j/C)/m_j = 2^k/C$, which will be a power of 2.

We are now ready to describe our algorithm.

The Main Algorithm

Input: BGT instance $I = (h_1 \geq h_2 \geq \dots \geq h_n)$.

Output: A (cyclic) perpetual schedule of I specified by pairs (p_i, q_i) , for $1 \leq i \leq n$. Item b_i occurs in the schedule at positions $p_i + kq_i$, for $k \geq 0$.

1. Set the parameter $\delta = 3\sqrt{h_1/H}$.

Form a pseudo-instance $(f_1'' \leq f_2'' \leq \dots \leq f_n'')$ of Pinwheel, by setting $f_i'' = (1 + \delta)H/h_i$. The density of this pseudo-instance is $1/(1 + \delta)$. Let $min \geq 1$ be the largest integer such that $2^{min} \leq f_1''$ (the setting of the parameter δ implies that $f_1'' \geq 2$ regardless of the value of H/h_1) and let $max \geq min$ be the smallest integer such that $2^{max+1} > f_n''$.

2. Reduce each frequency f_i'' to the closest value f_i of the form $2^k(1 + j/C)$, where k , C and j are integers such that $k \geq min$, $C = 2^q$ for $q = \lfloor min/2 \rfloor$, and $0 \leq j \leq C - 1$. These conditions imply that the new frequencies f_i are integral. Since this update of frequencies has reduced them at most by a

factor of $1 + 1/C$, the density of the whole sequence of frequencies increases at most by a factor of $1 + 1/C$ to at most $(1 + 1/C)/(1 + \delta)$.

The sequence (f_1, f_2, \dots, f_n) is our (proper) instance $V(I, \delta)$ of the Pinwheel problem. The remaining steps compute a schedule of this sequence. Steps 3-5 use Observations 1 and 2, and further reductions of the frequencies if needed, to transform the sequence (f_1, f_2, \dots, f_n) to a sequence (g_1, g_2, \dots, g_r) , $r \leq n$ where all frequencies are powers of 2.

We refer to the range $[2^k, 2^{k+1})$ of frequencies as *layer* k , and to the set of frequencies of the same value $2^k(1 + \frac{j}{C})$ as the *group* j in layer k .

3. For $k = \max, \max - 1, \dots, \min + 1$, apply Observation 1 in layer $[2^k, 2^{k+1})$ as many times as possible to combine pairs of the same frequencies different to powers of 2, pushing them down to the lower layer.

On the conclusion of this step, there is at most one frequency $2^k(1 + \frac{j}{C})$, for each $k > \min$ and $j \in (0, C)$.

4. Apply Observation 2 in the layer $[2^{\min}, 2^{\min+1})$ until there are at most $C + j - 1$ frequencies $2^{\min}(1 + \frac{j}{C})$ left, for any $j \in (0, C)$.
5. For $k = \max, \max - 1, \dots, \min$, reduce all remaining frequencies in range $[2^k, 2^{k+1})$ which are different to powers of 2, group by group starting from the top group defined by $j = C - 1$, and pushing each frequency down to the next lower group. While progressing down through the groups, keep applying Observation 1 whenever possible, if in a layer $k \geq \min + 1$, and Observation 2 in the lowest layer for $k = \min$.

On the conclusion of this step, we have a sequence of frequencies (g_1, g_2, \dots, g_r) , $r \leq n$, which are powers of 2, but the density further increases by some value $\Delta D \geq 0$. Thus the density of this final powers-of-2 instance (g_i) of the Pinwheel problem is at most $(1 + \frac{1}{C})/(1 + \delta) + \Delta D$. We will show that the setting of the parameters δ and C imply that this bound is at most 1, so sequence (g_i) is feasible.

6. Compute a cyclic schedule for the sequence of frequencies (g_1, g_2, \dots, g_r) using the algorithm for powers-of-2 Pinwheel instances from [20]. Such a schedule is specified by pairs (p_i, q_i) , for $1 \leq i \leq r$, which mean that the frequency g_i is placed in the perpetual schedule of (g_1, g_2, \dots, g_r) at positions $p_i + kq_i$, for $k \geq 0$.

Expand the schedule of (g_1, g_2, \dots, g_r) to the schedule of (f_1, f_2, \dots, f_n) (and consequently the schedule of (h_1, h_2, \dots, h_n)) by tracing back the applications of Observations 1 and 2.

The execution of steps 1-5 of the algorithm (the computation of the powers-of-2 Pinwheel instance (g_i)) is illustrated in Figure 2. The final step 6 of the algorithm is illustrated in Figure 1. A frequency f_j (computed in step 2) was subsequently (in steps 3-5) paired twice with other frequencies by applications of Observation 1 (contributing first to a new frequency $f_j/2$ and then to a new frequency $f_j/4$) and ended up in a group of 10 equal frequencies $f_j/4$, which were replaced with one new frequency $g_i = f_j/40$ by an application of Observation 2.

Let the pair (p_i, q_i) represent the positions of g_i in the computed cyclic schedule of (g_1, g_2, \dots, g_r) : frequency g_i is placed at positions $p_i + kq_i$, for $k \geq 0$.

Expanding the schedule of (g_1, g_2, \dots, g_r) to the schedule of (f_1, f_2, \dots, f_n) , every 40-th occurrence of frequency g_i is replaced by frequency f_j . More precisely, we first replace every 10-th occurrence of g_i with the frequency a , starting from the 4-th occurrence of g_i . Then every second occurrence of a is replaced with y , starting from the 2-nd occurrence of a , and finally every other occurrence of y is replaced with v , that is, with the frequency f_j . Thus frequency f_j is placed in the schedule at positions $(p_i + 13q_i) + k(40q_i)$.

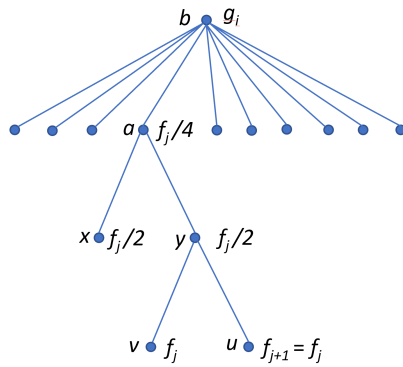


Fig. 1. Illustration of combining frequencies in the Main Algorithm. In steps 3–5 of the algorithm, frequency f_j is paired twice with other frequencies by applications of Observation 1, creating a frequency $f_j/4$ (node a). The resulting frequency is then put into a group of 10 frequencies $f_j/4$ and this group is replaced with one new frequency $g_i = f_j/40$ by an application of Observation 2.

Theorem 2. *For each BGT input instance, the Main Algorithm computes a $(1 + 3\sqrt{h_1/H})$ -approximation schedule.*

Proof. We show that the density of the sequence of frequencies (g_1, g_2, \dots, g_r) computed in step 5 of the algorithm is at most 1. Since these frequencies are powers of 2, the sequence (g_1, g_2, \dots, g_r) is feasible and gives a $(1 + \delta)$ -approximation schedule for the initial BGT input instance (Lemma 1).

The density of the Pinwheel pseudo-instance $(f''_1, f''_2, \dots, f''_n)$ is $D'' = 1/(1 + \delta)$. To bound the density D of the Pinwheel instance (f_1, f_2, \dots, f_n) , observe that if $f_i = 2^k(1 + j/C)$, then f_i has been obtained from the rational frequency f''_i such that

$$2^k \left(1 + \frac{j}{C}\right) \leq f''_i < 2^k \left(1 + \frac{j+1}{C}\right).$$

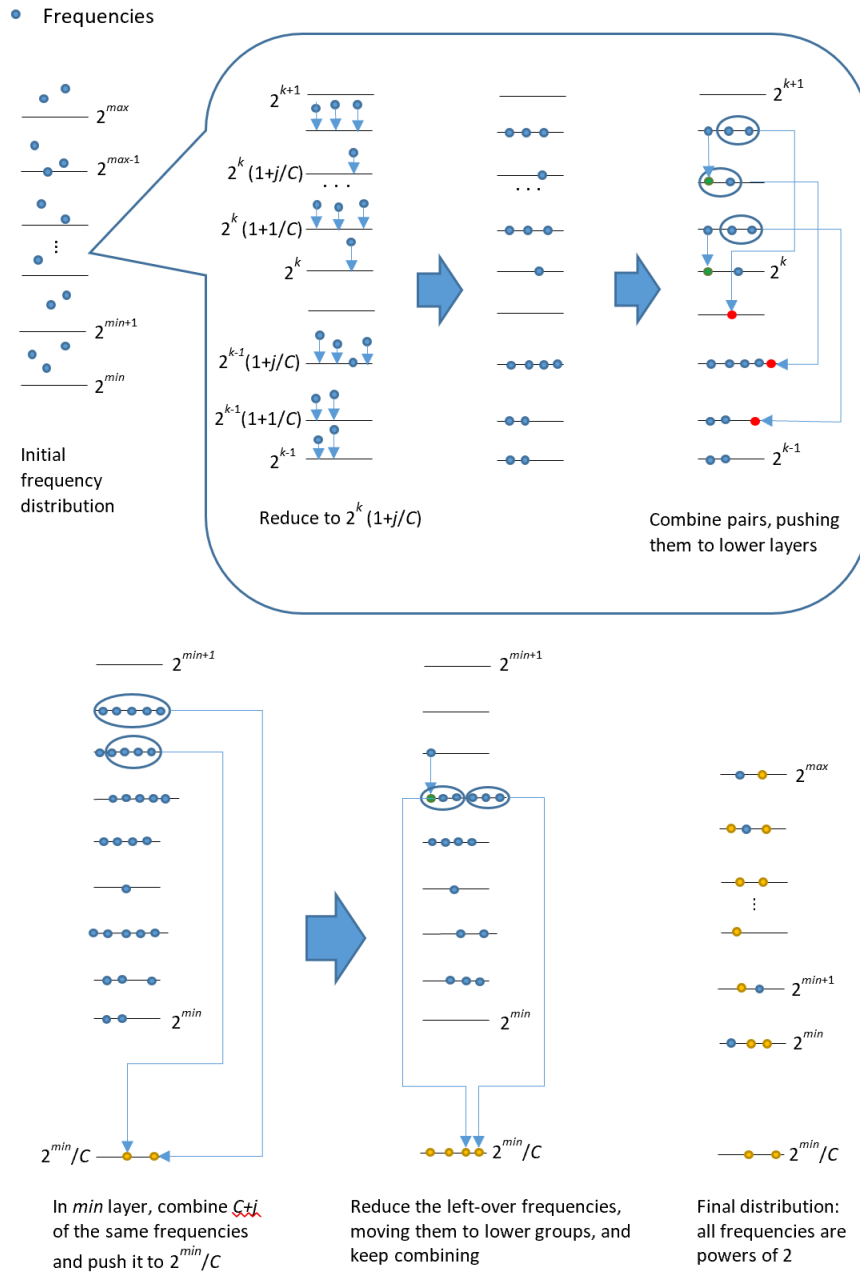


Fig. 2. Illustration of the execution of the Main Algorithm. The top half shows the transformation of the frequencies for the layers other than the min layer. The bottom half shows the transformations in the min layer.

This means that

$$f_i = 2^k \left(1 + \frac{j}{C}\right) > f_i'' \left(1 + \frac{j}{C}\right) / \left(1 + \frac{j+1}{C}\right) = f_i'' \frac{C+j}{C+j+1} \geq f_i'' \frac{C}{C+1},$$

so

$$D \leq (1 + 1/C)D'' = (1 + 1/C)/(1 + \delta).$$

Steps 3 and 4 of the algorithm transform the Pinwheel instance (f_1, f_2, \dots, f_n) using Observations 1 and 2, but without changing its density.

Step 5 further transforms the sequence of frequencies, increasing the density of the sequence whenever individual frequencies are reduced. We bound separately the increase ΔD_{above} of the density when we modify the frequencies in layers $max, max - 1, \dots, min + 1$, and the increase ΔD_{min} of the density when we modify the frequencies in layer min .

$$\begin{aligned} \Delta D_{above} &\leq \sum_{k=min+1}^{max} \sum_{j=1}^{C-1} \frac{1}{2^k} \left(\frac{1}{1 + (j-1)/C} - \frac{1}{1 + j/C} \right) \\ &= \sum_{k=min+1}^{max} \frac{1}{2^k} \left(1 - \frac{1}{1 + (C-1)/C} \right) \leq \frac{1}{2^{min}} \cdot \frac{C-1}{2C-1}. \end{aligned}$$

$$\begin{aligned} \Delta D_{min} &\leq \sum_{j=1}^{C-1} (C+j-1) \frac{1}{2^{min}} \left(\frac{1}{1 + (j-1)/C} - \frac{1}{1 + j/C} \right) \\ &= \frac{C}{2^{min}} \cdot \sum_{j=1}^{C-1} \frac{1}{C+j}. \end{aligned}$$

Let $K = 2^{min}/C^2 \in \{1, 2\}$. Then the density of the final powers-of-two Pinwheel instance (g_1, g_2, \dots, g_r) is at most

$$\begin{aligned}
& D + \Delta D_{above} + \Delta D_{min} \\
& \leq \frac{1+1/C}{1+\delta} + \frac{1}{2^{min}} \cdot \frac{C-1}{2C-1} + \frac{C}{2^{min}} \cdot \sum_{j=1}^{C-1} \frac{1}{C+j} \\
& = \frac{1+1/C}{1+\delta} + \frac{1}{KC} \cdot \left(\frac{C-1}{C \cdot (2C-1)} + \sum_{j=1}^{C-1} \frac{1}{C+j} \right) \tag{3}
\end{aligned}$$

$$\begin{aligned}
& = \frac{1+1/C}{1+\delta} + \frac{1}{KC} \cdot \left(\frac{1}{C} - \frac{1}{2C-1} + \sum_{j=1}^{C-1} \frac{1}{C+j} \right) \\
& = \frac{1+1/C}{1+\delta} + \frac{1}{KC} \cdot \sum_{j=C}^{2C-2} \frac{1}{j} \\
& \leq \frac{1+1/C}{1+\delta} + \frac{1}{KC} \cdot \ln 2 \tag{4}
\end{aligned}$$

$$\begin{aligned}
& = \frac{1}{1+\delta} + \frac{1}{C} \cdot \left(\frac{1}{1+\delta} + \frac{\ln 2}{K} \right) \\
& \leq \frac{1}{1+\delta} + \frac{\sqrt{2K}}{3} \cdot \frac{\delta}{\sqrt{1+\delta}} \cdot \left(\frac{1}{1+\delta} + \frac{\ln 2}{K} \right) \tag{5}
\end{aligned}$$

$$\leq 1. \tag{6}$$

To get Equality (3), use $2^{min} = KC^2$. To get Inequality (4), use the known fact that for the harmonic numbers $H_k = 1 + 1/2 + 1/3 + 1/4 + \dots + 1/k$, we have $H_k = \ln k + \delta_k$, where (δ_k) is a sequence of positive numbers strictly monotonically decreasing. Hence, we have $H_{2k} - H_k = \ln 2 + \delta_{2k} - \delta_k < \ln 2$. To get Inequality (5), use

$$C \cdot \sqrt{2K} = \sqrt{2^{min+1}} > \sqrt{f_1''} = \sqrt{(1+\delta) \frac{H}{h_1}} = \frac{3\sqrt{1+\delta}}{\delta}.$$

Inequality (6) follows from the fact that for $0 \leq \delta \leq 3$ and $K \in \{1, 2\}$, the function $f_K(\delta)$ in line (5) is maximized for $\delta = 0$. To see this, substitute $\gamma \equiv \sqrt{\delta+1}$ (so for δ increasing from 0 to 3, γ increases from 1 to 2) to obtain

$$\begin{aligned}
f_K(\delta) - 1 &= f_K(\gamma^2 - 1) - 1 = \frac{1}{\gamma^2} + \frac{\sqrt{2K}}{3} \cdot \frac{(\gamma^2 - 1)}{\gamma} \cdot \left(\frac{1}{\gamma^2} + \frac{\ln 2}{K} \right) - 1 \\
&= \frac{\gamma^2 - 1}{3\gamma^3} \left(\frac{\sqrt{2} \ln 2}{\sqrt{K}} \gamma^2 - 3\gamma + \sqrt{2K} \right). \tag{7}
\end{aligned}$$

The quadratic in the parentheses in (7) has two distinct real roots

$$\frac{3 \pm \sqrt{9 - 8 \ln 2}}{2\sqrt{2} \ln 2 / \sqrt{K}}.$$

For both $K \in \{1, 2\}$, the smaller root $r_{K,1}$ is less than 1 ($r_{1,1} < r_{2,1} = 0.823\dots$) and the larger root $r_{K,2}$ is greater than 2 ($r_{2,2} > r_{1,2} = 2.478\dots$). This means that (7) is equal to 0 for $\gamma = 1$ and negative for $\gamma \in (1, 2]$, or equivalently, $f_K(\delta) - 1$ is equal to 0 for $\delta = 0$ and negative for $\delta \in (0, 3]$. \square

The discussion in Section 2 and Theorem 2 imply the following corollary.

Corollary 1. *For Pinwheel instances $2 \leq f_1 \leq f_2 \leq \dots \leq f_n$, the Main Algorithm applied to sequences $(1/f_1, 1/f_2, \dots, 1/f_n)$ is a Pinwheel scheduler with the guarantee density threshold $1 - 3/\sqrt{f_1}$. Thus each Pinwheel instance with density at most $1 - 3/\sqrt{f_1}$ is feasible.*

The Main Algorithm can be implemented so that its running time is $O(n \log n)$, assuming constant-time operations on real numbers, including the logarithm operation and rounding. Steps 1 and 2 take linear time (or $O(n \log n)$, if the input sequence is not provided in the sorted order). In steps 3–5, the non-empty groups are maintained in a dictionary data structure, which can be created in $O(n \log n)$ time.

In steps 3 and 4, the groups are considered in the decreasing order (from the highest group in the highest layer). In step 3, if two frequencies are paired by Observation 1, then the group to which the new frequency should be added can be found (or created, if empty) in $O(\log n)$ time. This can happen only $O(n)$ times, since each pairing reduces the number of frequencies, so step 3 takes $O(n \log n)$ time. In step 4, if m frequencies are combined by Observation 2 into one new frequency $2^{\min}/C$, then the required computation takes $O(m)$ time (in this case, no need to look for the group with frequencies $2^{\min}/C$). Thus step 4 takes $O(n)$ time.

In step 5, the groups are again considered in the decreasing order. For each group, frequencies are paired by Observation 1 or combined by Observation 2 (as above, all this computation takes $O(n \log n)$ time) and the remaining frequencies are appended to the next non-empty group in constant time. Thus step 5 takes $O(n \log n)$ time.

Step 6, the computation of a cyclic schedule of the feasible powers-of-2 frequencies (g_1, g_2, \dots, g_r) , is done by the following algorithm from [20], which can be implemented to run in linear time, assuming that the input sequence is sorted. Take two largest frequency values g_{r-1} and g_r , replace them with a new frequency $g' = 2g_{r-1}$ and compute a cyclic schedule for the new $(r-1)$ -element sequence. To get a schedule for the original sequence (g_1, g_2, \dots, g_r) , replace the occurrences of g' with alternating occurrences of g_{r-1} and g_r . That is, if g' occurs at positions $p + kq$, for $k \geq 0$, then g_{r-1} occurs at positions $p + 2kq$ and g_r occurs at positions $p + (2k+1)q$, for $k \geq 0$.

During the computation in steps 3–5, the pairing and combining of frequencies is recorded in ordered trees as illustrated in Figure 1. The additional time required to create one such tree is linear in the size of the tree, so $O(n)$ time for creating all trees. By traversing these trees, we can expand in $O(n)$ time the cyclic schedule of (g_1, g_2, \dots, g_r) to a cyclic schedule of (f_1, f_2, \dots, f_n) .

The Main Algorithm computes the representation of a cyclic schedule in the form of pairs (p_i, q_i) , which enables constant-time answers to queries: given the current step T , when will be the next step when a given element b_i is serviced? To generate the schedule step by step, taking $O(\log n)$ time per step, maintain a priority queue (e.g. as the heap data structure) with the pairs (i, t_i) , where $i = 1, 2, \dots, n$ and t_i is the next step when element b_i will be serviced. Initially the priority queue contains pairs (i, p_i) . For the current step of the schedule, remove from the priority queue the pair (i, t_i) with the smallest t_i , schedule the element b_i , and insert to the priority queue the pair $(i, t_i + q_i)$.⁵

5 Continuous BGT

We consider now the continuous variant of the BGT problem. Since this variant models scenarios when bamboos are spread over some geographical area, we will now refer not only to bamboos b_1, b_2, \dots, b_n but also to the points v_1, v_2, \dots, v_n (in the implicit underlying space) where these bamboos are located. We will denote by V the set of these points.

Recall that input I for the continuous BGT problem consists of the rates of growth of bamboos $(h_i : 1 \leq i \leq n)$ and the travel times between bamboos $(t_{i,j} : 1 \leq i, j \leq n)$. We assume that $h_1 \geq h_2 \geq \dots \geq h_n$, as before, and normalize these rates, for convenience, so that $h_1 + h_2 + \dots + h_n = H = 1$ (this normalization is done without loss of generality, since only the relative heights of bamboos are relevant). We assume that the travel distances form a metric on V .

For any $V' \subseteq V$, the minimum growth rate among all points in V' is denoted by $h_{\min}(V')$, and the maximum growth rate among all points in V' is denoted by $h_{\max}(V')$. Let $h_{\min} = h_{\min}(V) = h_n$, and $h_{\max} = h_{\max}(V) = h_1$.

The diameter of the set V is denoted by $D = D(V) = \max\{t_{i,j} : 1 \leq i, j \leq n\}$. For any $V' \subseteq V$, $MST(V')$ denotes the minimum weight of a spanning tree on V' (the travel times are the weights of the edges). Recall that for an algorithm \mathcal{A} and input I , $MH(\mathcal{A}(I))$ denotes the maximum height that any bamboo ever reaches, if trimming is done according to the schedule computed by \mathcal{A} , and $OPT(I)$ is the optimal (minimal) maximum height of a bamboo over all schedules.

5.1 Lower bounds and simple approximation based on discrete BGT

We first show some simple lower bounds on the maximum height of a bamboo. For notational brevity, we omit the explicit reference to the input I . For example, the inequality $MH(\mathcal{A}) \geq Dh_{\max}$ in the lemma below is to be understood as $MH(\mathcal{A}(I)) \geq D(V(I)) \cdot h_{\max}(V(I))$, for each input instance I .

⁵ This works, if the cyclic schedule does not have gaps, and the cyclic schedule computed in the Main Algorithm does not have gaps. If the cyclic schedule may have gaps, then we should keep track of the current step T and schedule b_i only if $t_i = T$.

Lemma 2. $MH(\mathcal{A}) \geq Dh_{\max}$, for any algorithm \mathcal{A} .

Proof. The robot must visit another point x in V at distance at least $D/2$ from v_1 . When the robot comes back to v_1 after visiting x (possibly via a number of other points in V), the bamboo at v_1 has grown at least to the height of $Dh_1 = Dh_{\max}$. \square

Lemma 3. $MH(\mathcal{A}) \geq h_{\min}(V') \cdot MST(V')$, for any algorithm \mathcal{A} and $V' \subseteq V$.

Proof. Let v be the point in V' visited last: all points in $V' \setminus \{v\}$ have been visited at least once before the first visit to v . The distance traveled until the first visit to v is at least $MST(V')$, so the bamboo at v has grown to the height at least $h_v \cdot MST(V') \geq h_{\min}(V') \cdot MST(V')$. \square

One may ask how good are the schedules for continuous BGT which are computed taking into account only the growth rates, ignoring the travel times. If we use, for example, the schedules computed by the 2-approximation algorithm from Section 4, which guarantee that the maximum height of a bamboo does not grow above 2 (recall that we normalize the growth rates, so $H = 1$), then there are at most $\lfloor 2/h_i \rfloor - 1$ cutting actions between two consecutive cuttings of bamboo b_i . Otherwise bamboo b_i would grow in discrete BGT to the height at least $h_i(\lfloor 2/h_i \rfloor + 1) > 2 = 2H$. Thus in continuous BGT the time between two consecutive cuttings of bamboo b_i is at most $D\lfloor 2/h_i \rfloor$, so the height of this bamboo is never greater than $h_i D\lfloor 2/h_i \rfloor \leq 2D$. Combining this with Lemma 2, we conclude that the 2-approximation algorithm for discrete BGT is a $(2/h_{\max})$ -approximation algorithm for continuous BGT. In particular, this approach gives $\Theta(1)$ -approximate algorithm for continuous BGT for inputs with $h_1 = \Theta(1)$. To derive good approximation algorithms for other types of input, we will use the lower bound from Lemma 3.

5.2 Approximation algorithms

We present in this section three approximation algorithms for continuous BGT which are based on computing spanning trees. Algorithm 1 computes only one spanning tree of all points and the schedule for the robot is to traverse repeatedly an Euler tour of this tree. This simple algorithm ignores the growth rates of bamboos and computes a schedule of cutting taking into account only the travel times between points.

Algorithms 2 and 3 group the bamboos according to the similarity of their growth rates and compute a separate spanning tree for each group. The robot moves from one tree to the next in the Round-Robin fashion. At each spanning tree T , the robot walks along the Euler-tour of this tree for time D before moving to the next tree. When the robot eventually comes back to tree T , it resumes traversing the Euler tour of T from the point when it last left this tree. The growth rates of bamboos in the same spanning tree are within a factor of 2. Algorithm 3 differs from Algorithm 2 by using spanning trees only for the first $\Theta(\log n)$ groups, which include bamboos with growth rates greater than

Algorithm 1: An $O(h_{\max}/h_{\min})$ -approximation algorithm for continuous BGT.

1. [*Pre-processing*] Calculate a minimum spanning tree T of the point set V .
 2. [*Walking*] Repeatedly perform an Euler-tour traversal of T .
-

Algorithm 2: An $O(\log(h_{\max}/h_{\min}))$ -approximation algorithm for continuous BGT.

[*Pre-processing*]

1. Let $s = \lfloor \log_2(h_{\max}/h_{\min}) \rfloor + 1$.
 2. For $i \in \{1, 2, \dots, s\}$, let $V_i = \{v_j \in V \mid 2^{i-1} \cdot h_{\min} \leq h_j < 2^i \cdot h_{\min}\}$.
 3. **for** $i = 1$ **to** s such that $V_i \neq \emptyset$ **do**
 4. Let T_i be a minimum spanning tree on V_i .
 5. Let C_i be a directed Euler-tour traversal of T_i .
 6. Set an arbitrary point on C_i as the *last visited point* on C_i .
- [*Walking*]
7. **repeat forever**
 8. **for** $i = 1$ **to** s such that $V_i \neq \emptyset$ **do**
 9. Walk to the last visited point on C_i .
 10. If V_i has at least 2 points, then walk along C_i in the direction of the tour, stopping as soon when the distance covered is at least D .
-

$1/n^2$. The remaining bamboos, which all have low growth rates, are dealt with individually.

We describe our Algorithms 1, 2 and 3 in pseudocode and give their approximation ratio in the theorems below. The description of each algorithm consists of two parts: *pre-processing* and *walking*. We do not explicitly mention the actions of cutting/attending bamboos, assuming that whenever the robot passes a point in V , it cuts the bamboo growing at this point.

Theorem 3. *Algorithm 1 is an $O(h_{\max}/h_{\min})$ -approximation algorithm for the continuous BGT problem.*

Proof. Let \mathcal{A}_1 denote Algorithm 1. Every point $v_i \in V$ is visited by \mathcal{A}_1 at least every $2 \cdot MST(V)$ time units. Hence,

$$MH(\mathcal{A}_1) = O(h_{\max}(V) \cdot MST(V)). \quad (8)$$

According to Lemma 3,

$$OPT = \Omega(h_{\min}(V) \cdot MST(V)). \quad (9)$$

Combining the two bounds (8) and (9), it follows that Algorithm 1 is an $O(h_{\max}/h_{\min})$ -approximation algorithm for BGT. \square

Theorem 4. *Algorithm 2 is an $O(\log(h_{\max}/h_{\min}))$ -approximation algorithm for the continuous BGT problem.*

Algorithm 3: An $O(\log n)$ -approximation algorithm for continuous BGT.

[Pre-processing]

1. Let $s = \lceil 2 \cdot \log_2 n \rceil$.
2. Let $V_0 = \{v_i \in V \mid h_i \leq n^{-2}\}$. Let $V_0 = \{v'_0, v'_1, \dots, v'_{\ell-1}\}$.
3. For $i \in \{1, 2, \dots, s\}$, let $V_i = \{v_j \in V \mid 2^{i-1} \cdot n^{-2} < h_j \leq 2^i \cdot n^{-2}\}$.
4. **for** $i = 1$ **to** s such that $V_i \neq \emptyset$ **do**
5. Let T_i be a minimum spanning tree on V_i .
6. Let C_i be a directed Euler-tour traversal of T_i .
7. Set an arbitrary point on C_i as the *last visited point* on C_i .

[Walking]

8. $j = 0$.
 9. **repeat forever**
 10. **for** $i = 1$ **to** s such that $V_i \neq \emptyset$ **do**
 11. Walk to the last visited point on C_i .
 12. If V_i has at least 2 points, then walk along C_i in the direction of the tour, stopping as soon when the distance covered is at least D .
 13. If $V_0 \neq \emptyset$, then walk to v'_j and let $j = j + 1 \pmod{\ell}$.
-

Proof. For any $i \in \{1, 2, \dots, s\}$, consider any point $v \in V_i$. The robot walks along one tree for at most distance $2D$ and then covers at most distance D to move to the next tree. After a visit to point v , the robot comes back to tree T_i at most $\lceil 2 \cdot MST(V_i)/D \rceil$ times before visiting v again. Therefore, recalling from the algorithm that there are at most $s = \lceil \log_2(h_{\max}/h_{\min}) \rceil + 1$ trees, the distance traveled between two consecutive visits to point v is at most

$$3Ds \left\lceil \frac{2 \cdot MST(V_i)}{D} \right\rceil \leq 3s(D + 2 \cdot MST(V_i)).$$

Hence, the height of the bamboo at v is always

$$O\left(h_{\max}(V_i) \cdot \log\left(\frac{h_{\max}}{h_{\min}}\right) \cdot \max\{D, MST(V_i)\}\right). \quad (10)$$

On the other hand, using Lemmas 2 and 3, we obtain

$$OPT \geq h_{\min}(V_i) \cdot \max\{D, MST(V_i)\}. \quad (11)$$

Combining the two bounds (10) and (11), and observing that $h_{\max}(V_i) \leq 2 \cdot h_{\min}(V_i)$, we obtain that the height of the bamboo at v is always $O(OPT \cdot \log(h_{\max}/h_{\min}))$, so Algorithm 2 is an $O(\log(h_{\max}/h_{\min}))$ -approximation algorithm for BGT. \square

Theorem 5. *Algorithm 3 is an $O(\log n)$ -approximation algorithm for the continuous BGT problem.*

Proof. Each round of Algorithm 3, that is, each iteration of the repeat loop, is a cycle over all $s = \Theta(\log n)$ trees and a visit to one point in the set V_0 . Consider any point $v \in V_i$, for any $i \in \{1, 2, \dots, s\}$. The distance traveled between two consecutive visits of v is at most

$$\begin{aligned} (3Ds + D) \left\lceil \frac{2 \cdot \text{MST}(V_i)}{D} \right\rceil &\leq (3s + 1)(D + 2 \cdot \text{MST}(V_i)) \\ &= O(\log n \cdot \max\{D, \text{MST}(V_i)\}). \end{aligned}$$

Hence, the height of the bamboo at v is always

$$O(h_{\max}(V_i) \cdot \log n \cdot \max\{D, \text{MST}(V_i)\}).$$

Using the lower bound (11) on OPT and the fact that $h_{\max}(V_i) \leq 2 \cdot h_{\min}(V_i)$, we conclude that the height of the bamboo at v is always $O(OPT \cdot \log n)$.

Consider now a point $v \in V_0$. The distance traveled between two consecutive visits of v is at most

$$(3Ds + D)|V_0| = O(n \cdot D \cdot \log n).$$

Hence, the height of the bamboo at v is always

$$O(h_{\max}(V_0) \cdot n \cdot D \cdot \log n) = O(n^{-2} \cdot n \cdot D \cdot \log n) = O(h_{\max} \cdot D \cdot \log n).$$

This and Lemma 2 imply that the height of the bamboo at v is always $O(OPT \cdot \log n)$. Thus Algorithm 3 is an $O(\log n)$ -approximation algorithm for BGT. \square

Note that the pre-processing in all Algorithms 1, 2 and 3 is dominated by the minimum-spanning tree computation, which can be implemented in $O(n^2)$ time (e.g. by using Prim's algorithm [27]). Then the running time to produce the schedule (the consecutive steps of the schedule) is constant per one step of the schedule.

5.3 How tight are the upper and lower bounds?

Our $O(\log n)$ -approximation algorithm for the continuous BGT (Algorithm 3) can return schedules which are worse than the optimal schedules by a factor of $\Theta(\log n)$. For example, consider the input which consists of two sets V' and V'' of $n/2$ points each such that in each set the points are very close to each other (with the total distance to visit all points in this set less than D), but the sets are at distance greater than $D/2$ from each other. The rates of growth in set V' include the $\Theta(\log n)$ values $1/4, 1/8, \dots, 1/2^i, \dots, 1/n$, and the same rates are in set V'' . For this input instance the value of the optimal schedule is $\Theta(D)$: visit all points in V' , then all points in V'' (for the total distance $\Theta(D)$), and repeat. The schedule computed by Algorithm 3 uses $\Theta(\log n)$ trees and makes the robot traverse each tree for a distance at least D before returning to the bamboo with the highest rate of growth of $1/4$. Thus this bamboo grows to the height $\Theta(D \log n)$, which is a factor of $\Theta(\log n)$ worse than the optimum.

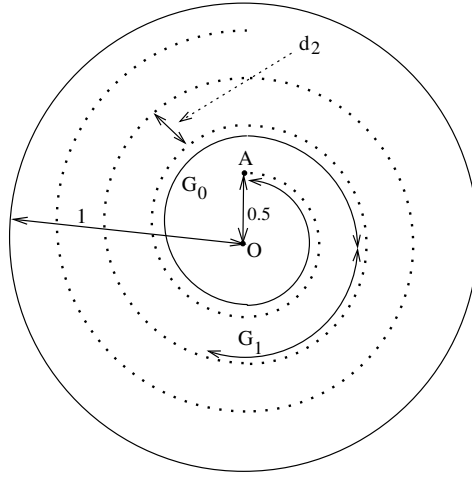


Fig. 3. Example of a spiral input.

The approximation bounds which we presented in Section 5.2 are derived by comparing the upper bounds on the maximum bamboo heights guaranteed by the algorithms with the lower bounds shown in Section 5.1. We show now a class of instances, for which any schedule leads to bamboo heights greater than our lower bounds by a $\Theta(\log n)$ factor. Thus for these input instances our general lower bounds turn out to be weak, while our $O(\log n)$ -approximation algorithm computes in fact constant approximation schedules. To improve the approximation ratio of algorithms for the continuous BGT to $o(\log n)$, one will therefore require both: new algorithmic techniques as well as stronger lower bounds.

We consider the following input V^* for the continuous BGT problem. The n points in V^* are placed on the Euclidean plane within a unit-radius circle. The points lie evenly spaced along the spiral inside this circle, which starts at a point A at distance $1/2$ from the center of the circle and swirls outward creating rings separated by distance $d_2 = n^{-1/3}$; see Figure 3, but note that the drawing is not to scale. The first point in V^* is at A and the distance between two consecutive points in V^* along the spiral is $d_1 = n^{-2/3}$. Thus the length of the spiral occupied by the points in V^* is equal to $n^{1/3}$, so all points are indeed inside the circle. (The length of the part of the spiral which is within the circle is at least $(1/2)\pi n^{1/3}$, since there are $n^{1/3}/2$ rings and each ring has length at least π .)

The points in V^* are grouped into subsets $G_1, G_2, \dots, G_{(\log n)/3}$ and G' of consecutive points along the spiral, starting from position A . (For convenience, we assume that $(\log n)/3$ is an integer.) The first two groups G_0 and G_1 are indicated in Figure 3. For $i = 1, 2, \dots, (\log n)/3$, the size of group G_i is $n/2^i$ and each point in G_i has the same growth rate $h_i = (3 - \epsilon)2^i/(n \log n)$, where

$0 < \epsilon = o(1)$. The last group G' contains the remaining $o(n)$ points in V^* and the growth rate of each point in G' is equal to $h' = 1/n^{4/3} = h_{\min}(V^*)$. The exact value of $\epsilon = o(1)$ is such that all growth rates sum up to 1.

The diameter of V^* is $1 \leq D \leq 2$ and $h_{\max}(V^*) = h_{(\log n)/3} = \Theta(1/(n^{2/3} \log n))$, so Lemma 2 gives the lower bound of $\Omega(1/(n^{2/3} \log n))$ on the optimal solution. We check now what lower bounds we can get from Lemma 3. The MST of this set of points V^* is obtained by following the spiral and the weight of this MST is equal to $nd_1 = n^{1/3}$, giving the lower bound of $h_{\min}(V^*) \cdot MST(V^*) = 1/n$. For each $i = 1, 2, \dots, (\log n)/3$, the weight of the MST of the set of points $V' = \bigcup_{j=i}^{(\log n)/3} G_j$, which is the subset of all points in V^* with growth rates at least h_i , is equal to $d_1 \left| \bigcup_{j=i}^{(\log n)/3} G_j \right| = \Theta(d_1 |G_i|) = \Theta(n^{1/3}/2^i)$. This gives the lower bound of $h_i \cdot MST(V') = \Omega(1/(n^{2/3} \log n))$. This is the best lower bound which we can obtain from Lemma 3, since we maximize $h_{\min}(V') \cdot MST(V')$ by including in V' *all* points with growth rates at least $h_{\min}(V')$. Thus Lemmas 2 and 3 give for this input instance the lower bound $\Omega(1/(n^{2/3} \log n))$.

The $O(\log n)$ -approximation Algorithm 3 gives the schedule for V^* with the maximum bamboo height $\Theta(1/n^{2/3})$, which is a $\log n$ factor above our lower bounds. Indeed, observe that the sets V_i , $i \geq 0$ defined in Algorithm 3 coincide with the sets G' and G_i , for $i = 1, 2, \dots, (\log n)/3$. This means that a bamboo in set G_i grows up to the height of $\Theta(h_i D \log n \lceil MST(G_i)/D \rceil) = \Theta(1/n^{2/3})$. We show next that for this input any possible schedule produces bamboos of height $\Omega(1/n^{1/3})$.

Lemma 4. *For each schedule for the input V^* , there must be a bamboo which grows to the height $d_1/2 = \Omega(1/n^{2/3})$.*

Proof. Assume that there is a schedule such that the height of each bamboo in $\bigcup_{i=1}^{(\log n)/3} G_i$ is always at most $d_1/2$. In such a schedule, each point in each set G_i is serviced after at most distance $(d_1/2) \cdot (n \log n) / ((3-\epsilon)2^i) \leq d_1 \cdot (n \log n) / 2^{i+2}$. Since the distance between each two points in V^* is at least d_1 , the growth rate of each point in G_i in this schedule must be at least $2^{i+2}/(n \log n)$. Thus the sum of the growth rates of all points in $\bigcup G_i$ in this schedule is at least

$$\sum_{i=1}^{(\log n)/3} \frac{2^{i+2}}{n \log n} \cdot \frac{n}{2^i} = 4/3,$$

which is a contradiction since the growth rates of the points in any valid schedule must sum up to at most 1. This contradiction implies that in each valid schedule there is a bamboo which grows higher than $d_1/2$. \square

6 Open questions

There are several interesting open questions about approximation algorithms for the BGT problems. For discrete BGT, we still do not know the exact

worst-case approximation ratios of simple strategies such as Reduce-Max and Reduce-Fastest. Assuming the growth rates are normalized to $H = 1$, the best known upper bound on the maximum height of a bamboo under the Reduce-Fastest strategy is 2.62, and the only known constant upper bound for Reduce-Max is 9 – both bounds shown recently by Bilò *et al.* [6]. For both strategies, there are simple examples showing that bamboos can reach heights arbitrarily close to 2, but we do not know of any examples when the heights would go above 2.

Other set of questions are about improving approximation ratios of offline Pinwheel-based algorithms. Van Ee [31] has recently improved on the simple 2-approximation described in Section 4, designing a 12/7-approximation algorithm. Can this bound be further improved? Another question is whether the $1 + O(\sqrt{h_1/H})$ approximation ratio of our main algorithm for discrete BGT can be improved. Can we achieve an approximation ratio of $1 + O(h_1/H)$, or even just $1 + o(\sqrt{h_1/H})$?

For continuous BGT, we do not know whether our Algorithm 3, or any other algorithm, achieves an approximation ratio of $o(\log n)$. The two special cases of continuous BGT – discrete BGT and the metric TSP – have both constant-ratio polynomial-time approximation algorithms, not giving any indication why we should not expect the same for the more general problem. Note that a constant approximation for the path was given in [13], and very recently at IWOCA 2020 (best paper) a PTAS was presented by Peter Damaschke [17].

References

1. Sultan S. Alshamrani, Dariusz Kowalski, and Leszek Gašieniec. Efficient discovery of malicious symptoms in clouds via monitoring virtual machines. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1703–1710, Oct 2015.
2. Annelieke C. Baller, Martijn van Ee, Maaïke Hoogeboom, and Leen Stougie. Complexity of inventory routing problems when routing is easy. *Networks*, 75(2):113–123, 2020.
3. Sanjoy K. Baruah, N. K. Cohen, C. Greg Plaxton, and Donald A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, Jun 1996.
4. Sanjoy K. Baruah and Shun-Shii Lin. Pfair scheduling of generalized pinwheel task systems. *IEEE Transactions on Computers*, 47(7):812–816, July 1998.
5. Michael A. Bender, Sándor P. Fekete, Alexander Kröller, Vincenzo Liberatore, Joseph S. B. Mitchell, Valentin Polishchuk, and Jukka Suomela. The minimum backlog problem. *Theoretical Computer Science*, 605:51–61, 2015.
6. Davide Bilò, Luciano Gualà, Stefano Leucci, Guido Proietti, and Giacomo Scornavacca. Cutting bamboo down to size. In Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara, editors, *10th International Conference on Fun with Algorithms, FUN 2021, May 30 to June 1, 2021, Favignana Island, Sicily, Italy*, volume 157 of *LIPICs*, pages 5:1–5:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

7. Marijke H. L. Bodlaender, Cor A. J. Hurkens, Vincent J. J. Kusters, Frank Staals, Gerhard J. Woeginger, and Hans Zantema. Cinderella versus the wicked stepmother. In Jos C. M. Baeten, Thomas Ball, and Frank S. de Boer, editors, *Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference, TCS 2012, Amsterdam, The Netherlands, September 26-28, 2012. Proceedings*, volume 7604 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2012.
8. Gerth Stølting Brodal, George Lagogiannis, and Robert Endre Tarjan. Strict fibonacci heaps. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1177–1184. ACM, 2012.
9. Mee Yee Chan and Francis Y. L. Chin. General schedulers for the pinwheel problem based on double-integer reduction. *IEEE Transactions on Computers*, 41(6):755–768, June 1992.
10. Mee Yee Chan and Francis Y. L. Chin. Schedulers for larger classes of pinwheel instances. *Algorithmica*, 9(5):425–462, May 1993.
11. Wei-pang Chin and Simeon C. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39–44, 1988.
12. Marek Chrobak, János Csirik, Csanád Imreh, John Noga, Jirí Sgall, and Gerhard J. Woeginger. The buffer minimization problem for multiprocessor scheduling with conflicts. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 862–874. Springer, 2001.
13. Huda Chuangpishit, Jurek Czyzowicz, Leszek Gąsieniec, Konstantinos Georgiou, Tomasz Jurdzinski, and Evangelos Kranakis. Patrolling a path connecting a set of points with unbalanced frequencies of visits. In A Min Tjoa, Ladjel Bellatreche, Stefan Biffl, Jan van Leeuwen, and Jiri Wiedermann, editors, *SOFSEM 2018: Theory and Practice of Computer Science - 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29 - February 2, 2018, Proceedings*, volume 10706 of *Lecture Notes in Computer Science*, pages 367–380. Springer, 2018.
14. Andrew Collins, Jurek Czyzowicz, Leszek Gąsieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, Russell Martin, and Oscar Morales Ponce. Optimal patrolling of fragmented boundaries. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA 2013, pages 241–250, New York, NY, USA, 2013. ACM.
15. Jurek Czyzowicz, Leszek Gąsieniec, Adrian Kosowski, and Evangelos Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, pages 701–712. Springer, 2011.
16. Jurek Czyzowicz, Leszek Gąsieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, and Najmeh Taleb. When patrolmen become corrupted: Monitoring a graph using faulty mobile robots. *Algorithmica*, 79(3):925–940, 2017.
17. Peter Damaschke. Two robots patrolling on a line: Integer version and approximability. In Leszek Gąsieniec, Ralf Klasing, and Tomasz Radzik, editors, *Combinatorial Algorithms - 31st International Workshop, IWOCA 2020, Bordeaux, France, June 8-10, 2020, Proceedings*, volume 12126 of *Lecture Notes in Computer Science*, pages 211–223. Springer, 2020.

18. Peter C. Fishburn and J. C. Lagarias. Pinwheel scheduling: Achievable densities. *Algorithmica*, 34(1):14–38, Sep 2002.
19. Leszek Gašieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In *SOFSEM 2017: Theory and Practice of Computer Science - 43rd Int. Conf. on Current Trends in Theory and Practice of Computer Science, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2017.
20. Robert Holte, Al Mok, Louis Rosier, Igor Tulchinsky, and Donald Varvel. The pinwheel: a real-time scheduling problem. In *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track*, pages 693–702, Jan 1989.
21. Robert Holte, Louis Rosier, Igor Tulchinsky, and Donald Varvel. Pinwheel scheduling with two distinct numbers. *Theoretical Computer Science*, 100(1):105–135, 1992.
22. Haim Kaplan, Robert E. Tarjan, and Kostas Tsioutsoulouklis. Faster kinetic heaps and their use in broadcast scheduling. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01*, page 836–844, USA, 2001. Society for Industrial and Applied Mathematics.
23. Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81(8):1665–1677, 2015.
24. Akitoshi Kawamura and Yusuke Kobayashi. Fence patrolling by mobile agents with distinct speeds. *Distributed Computing*, 28(2):147–154, April 2015.
25. Shun-Shii Lin and Kwei-Jay Lin. A pinwheel scheduler for three distinct numbers with a tight schedulability bound. *Algorithmica*, 19(4):411–426, Dec 1997.
26. Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166 – 204, 1981.
27. R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, Nov. 1957.
28. Theodore H. Romer and Louis E. Rosier. An algorithm reminiscent of Euclidean computing a function related to pinwheel scheduling. *Algorithmica*, 17(1):1–10, Jan 1997.
29. Paolo Serafini and Walter Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.
30. Jorge Urrutia. Chapter 22 - art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, Amsterdam, 2000.
31. Martijn van Ee. A $12/7$ -approximation algorithm for the discrete bamboo garden trimming problem. *Operations Research Letters*, 49(5):645–649, 2021.