A 10/7-Approximation for Discrete Bamboo Garden Trimming and Continuous Trimming on Star Graphs

Felix Höhne
Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Rob van Stee
University of Siegen, Germany

Abstract

In the discrete bamboo garden trimming problem we are given n bamboo that grow at rates $v_1, \ldots, v_n$ per day. Each day a robotic gardener cuts down one bamboo to height 0. The goal is to find a schedule that minimizes the height of the tallest bamboo that ever exists.

We present a 10/7-approximation algorithm that is based on a reduction to the pinwheel problem. This is consistent with the approach of earlier algorithms, but some new techniques are used that lead to a better approximation ratio.

We also consider the continuous version of the problem where the gardener travels in a metric space between plants and cuts down a plant each time he reaches one. We show that on the star graph the previously proposed algorithm *Reduce-Fastest* is a 6-approximation and the known *Deadline-Driven Strategy* is a $(3 + 2\sqrt{2})$-approximation. The Deadline-Driven Strategy is also a $(9 + 2\sqrt{5})$-approximation on star graphs with multiple plants on each branch.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases bamboo garden trimming, approximation algorithms, scheduling

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.16

Category APPROX

Supplementary Material *Software*: `https://github.com/Felixhhne/bamboo`
archived at `swh:1:dir:dc21af15b5e2d50d6b27b4b6f8fce8de22b6e9cb`

1 Introduction

In the discrete bamboo garden trimming problem (BGT), first introduced by Gasieniec et al. [8] we are given a set of n bamboo that grow at rates $v_1, \ldots, v_n$ per day, i.e. the height of bamboo i extends by v_i each day. We assume that these growth rates are arranged such that $v_1 \geq v_2 \geq \cdots \geq v_n$. Initially the height is set to zero. Each day a robotic gardener cuts down one bamboo to height zero. The goal is to design a trimming schedule such that the height of the tallest bamboo is minimized. Gasieniec et al. [8] gave a 2-approximation for discrete BGT which has been improved by van Ee [14] to a $\frac{12}{7}$-approximation.

Both results are obtained by reducing BGT to the pinwheel scheduling problem. The pinwheel scheduling problem is motivated by the communication between a satellite and its ground station. The ground station wants to receive messages from n satellites. Time is slotted and a satellite i sends a message for p_i consecutive timeslots before switching to a different message. Each timeslot the ground station receives a message from a single satellite. This means in order to guarantee that no message is missed we need to find a schedule that allocates at least one timeslot to satellite i in any interval of p_i units of time. We discuss the literature on pinwheel scheduling in Section 2.

If a BGT algorithm maintains a height of K then each bamboo i must be visited at least once in a period of $p_i^* := \lfloor \frac{K}{v_i} \rfloor$ time steps. That is, there is a BGT schedule that maintains height K for the input $(v_1, \ldots, v_n)$ if and only if the pinwheel problem $(p_1^*, \ldots, p_n^*)$ is schedulable.

There are also a number of results that do not use the connection to the pinwheel problem, using various greedy-type algorithms. The *Deadline-Driven Strategy* always cuts the bamboo with the earliest deadline provided that the height of this bamboo has reached a certain threshold. The deadline of a bamboo is the time it reaches the height the algorithm wants to maintain. This algorithm has already been considered for discrete BGT and there it is a 2-approximation as shown by J. Kuszmaul [12].

A second simple algorithm is *Reduce-Fastest* which is a 2.62-approximation for discrete BGT as shown by Bilò et al. [2] This algorithm always cuts the fastest plant which has reached a certain threshold. A similar algorithm is *Reduce-Max* that always cuts the highest bamboo and is a 4-approximation. Both of these algorithms are online algorithms based on simple queries. This means they are flexible and can easily adapt to changes in the input (the set of growth rates).

In the first part of this paper we present an algorithm based on a pinweel reduction that improves the approximation ratio for discrete BGT to $\frac{10}{7}$. This algorithm combines a binary search with the known technique of porous schedules for pinwheel scheduling. This marks the first time that porous schedules are used to approximate the BGT problem.

In the second part, we consider the continuous version of the BGT problem, also introduced by Gasieniec et al. [8]. In this version the bamboo are located in some metric space and the gardener needs to travel between the bamboo to cut them. Cutting is done instantly and the goal is to find a route that minimizes the maximum height of the bamboos. Gasieniec et al. give a $O(\log n)$-approximation algorithm that works in any metric space.

We consider the case where the underlying metric is a star graph. In this context we study some of the simple algorithms above (compared to the relative complicatedness of a pinwheel schedule) that achieve constant approximation ratios for this case.

1.1 Our results

For discrete BGT we propose a 10/7-approximation algorithm that is based on a reduction to the pinwheel-problem. Previous work used only the sum of the speeds H as a lower bound on the optimum value. We improve on this by using binary search in the interval $[H, 2H]$ and by planning to lose a certain factor in advance. By this we mean that all periods are multiplied by our goal ratio after calculating them from the speeds, making it easier to find a pinwheel schedule. We can show that our algorithm always finds a schedule of height at most $10K/7$ if a schedule of height K exists. This is achieved by using the technique of porous schedules which we will discuss in Section 2. Thus, the binary search is used to essentially find the best possible lower bound for the optimal value within this framework.

The result can be improved to $\frac{7}{5}$ using a computer-assisted proof. Unfortunately the proof is too long to fit into the appendix but can be found under the link `https://github.com/Felixhhne/bamboo`

In the continuous version of the problem we consider a star graph. We show that the 2.62-approximate algorithm Reduce-Fastest for discrete BGT, which can be seen as a special instance of a star graph, still works on arbitrary star graphs with one bamboo on each branch and we show that it is a 6-approximation in this case.

Furthermore the deadline driven algorithm gives us a $(3+2\sqrt{2})$-approximation on the star. This result can be extended to the case where there are multiple bamboo on each branch of the star. Here we pay a price in the approximation ratio and achieve a $(9+2\sqrt{5})$-approximation.

1.2 Related work

Both the discrete and continuous version of BGT were first introduced by Gasieniec et al. [8]. They provide a variety of results. The first algorithm they present is Reduce-Fastest, which in each step cuts the fastest growing bamboo above a certain height treshhold. This algorithm is a 2.62-approximation as shown by Bilò el al. [2]. A similar algorithm is Reduce-Max that always cuts the highest bamboo and is a 4-approximation. The final algorithm in this set of algorithms based on simple queries is the Deadline-Driven Strategy which is a 2-approximation. The results for Reduce-Max as well as the Deadline-Driven Strategy are from J. Kuszmaul [12].

Gasieniec et al. also give a fully offline 2-approximation algorithm that preprocesses the input and reduces the problem to a pinwheel-instance. This approach has been improved by van Ee [14] to a $\frac{12}{7}$-approximation.

There are other problems where the goal is to minimize the maximum height or backlog reached on a machine. One example is the Minimum Backlog Problem [3, 1, 13] where an adversary distributes water among a set of cups and the player may empty one or more cups on their turn.

In the problem of Buffer minimization with conflicts [5] there is a set of machines on a graph and load may arrive on these machines at any time. The algorithm runs machines to decrease their load but machines that are adjacent to each other on the graph may not run at the same time.

2 Pinwheel problems

We represent an instance of the pinwheel scheduling problem by the vector $p = (p_1, \ldots, p_n)$ of periods with which each plant should be cut (or: each machine should be scheduled).

▶ **Definition 1.** *The* density *of a pinwheel scheduling instance* $p = (p_1, \ldots, p_n)$ *is* $d(p) = \sum_{i=1}^{n} \frac{1}{p_i}$.

For example, the instance $(2, 3)$ has density $\frac{5}{6}$ and can be scheduled by repeating the sequence 12. Because $121212\ldots$ is the *only* feasible schedule, the instance $(2, 3, p_3)$ can not be scheduled regardless of the value of p_3.

The pinwheel problem was first introduced by Holte et al. [11] and then picked up by Chan and Chin [4]. They conjecture that any pinwheel instance with density up to 5/6 can be scheduled. The above example shows that a better guarantee is not possible. This conjecture is supported by a variety of works, including Fishburn and Lagarias [7] who show that any instance with $p_1 = 2$ and density up to 5/6 can be scheduled. Dei Wing [6] shows that the claim also holds for low-dimensional vectors with dimension up to 5 and more recently Gasieniec et al. [9] improve this result further by proving the claim for instances with up to 12 elements. Additionally they give a set of schedules that solve all schedulable instances with at most 5 tasks.

Without loss of generality, the elements of the pinwheel instance are in nondecreasing order. An obvious requirement for schedulability is $d(p) \leq 1$. Hence, aside from the instance (1) an instance can only be scheduled if $p_1 \geq 2$. Consequently we assume $2 \leq p_1 \leq p_2 \leq \cdots \leq p_n$.

Given a pinwheel instance $p = (p_1, \ldots, p_n)$, a successful schedule for p is an infinite sequence over $\{1, \ldots, n\}$ such that any subsequence of length p_i contains at least one i. An important consideration for pinwheel scheduling and BGT is the representation of a solution. This is because in general an explicit representation of the schedule may take exponential space. The solutions for pinwheel scheduling provided by Fishburn and Lagarias as well as

Chan and Chin come in the form of *fast online schedulers*. These are algorithms that can decide whether they can schedule an instance in polynomial time and then generate each symbol of the schedule in constant time.

Porous schedules

Fishburn and Lagarias introduced the concept of porous schedules [7] as a useful tool in the construction of pinwheel schedules.

Let $p = (p_1, \ldots, p_n)$. Let $\mathcal{S}$ be a subset of $\{1, \ldots, n\}$ and define $\mathcal{U} = \{1, \ldots, n\} \setminus \mathcal{S}$. A *porous schedule* for $\mathcal{S}$ (the set of *scheduled* tasks) is a schedule that allocates a slot for plant $i \in \mathcal{S}$ at least once every p_i positions, but also contains slots that are not allocated to any plant. When writing out the schedule the first type of slot is denoted by i and the second type, which we also call a hole, is denoted by _. In this paper we will only consider schedules that consist of infinite repeats of finite lists, and describe such a schedule by giving only that list. Even that part may have exponential size as a function of the size of the input.

Let

$$D_{\mathcal{U}} = 1 - \sum_{k \notin \mathcal{U}} \frac{1}{p_k^*}$$

be the maximum possible density of the unscheduled machines.

For example, given the instance (2,4,8,9), we could set $\mathcal{U} = \{3, 4\}$, which corresponds to the (unscheduled) plants with speed 8 and 9, and $\mathcal{S} = \{1, 2\}$, which corresponds to the plants with speed 2 and 4. The schedule 12_ (meaning 12_12_12_...) is a porous schedule for the scheduled tasks $\mathcal{S}$, while $\mathcal{U}$ is the set of leftover plants that still need to be scheduled.

▶ **Definition 2.** *Let $p = (p_1, \ldots, p_n)$ be a pinwheel instance and let f be a porous schedule for $\mathcal{S} \subseteq \{1, \ldots, n\}$. Then for $i \in \mathcal{U} = \{1, \ldots, n\} \setminus \mathcal{S}$, we define h_i as the minimum number of holes in p_i consecutive positions of f.*

Sometimes we also write the function $h(x)$ to represent the minimum number of holes in x consecutive positions. The set $h = \{h_i | i \in \mathcal{U}\}$ is again a pinwheel instance unless there exists an i with $h_i = 0$. Of course, in general h might not be schedulable. Lemma 2 of [7] explains how solving the pinwheel problem h can lead to a solution of p. A version of it is given below.

▶ **Lemma 3.** *Consider a pinwheel instance p and let a porous schedule f for a subset $\mathcal{S}$ of the tasks. Let $\mathcal{U} := \{1, \ldots, n\} \setminus \mathcal{S}$. If $h = \{h_i | i \in \mathcal{U}\}$ is a schedulable pinwheel instance, then p is schedulable.*

Proof. By assumption, there is a schedule $g : \mathbb{Z} \to \mathcal{U}$ (that we can enumerate using a fast online scheduler) for the input h. Let β be an order preserving map from the set of holes in f to $\mathbb{Z}$. Let $f' = f$ except for the holes in f where $f'(k) = g(\beta(k))$. Then f' is a schedule for p. ◀

Let $b \in [0, 1]$. We say that b is a *density guarantee* for the pinwheel problem if all instances p with $d(p) \leq b$ can be scheduled. This is always taken to mean that there exists a schedule for p and we can efficiently find a fast online scheduler to generate the sequence. Our results rely on the following theorem.

▶ **Theorem 4** ([7])**.** *The value $3/4$ is a density guarantee for the pinwheel problem. For instances with $p_1 = 2$, the density guarantee is $5/6$. In both instances, schedules can be found in time $O(n^3)$.*

Note that solving or scheduling a pinwheel instance for our purposes means finding a fast online scheduler but does not include explicitly writing out the schedule (not even just the finite part that repeats). Chan and Chin give an algorithm that runs in time $O(n^3)$ and schedules any instance with density at most $\frac{7}{10}$. Their algorithm achieves the following guarantees based on p_1:

p_1	2	3	4	5	6	7	8	≥ 9
guarantee	0.75	0.70	0.708	0.721	0.733	0.738	0.744	> 0.75

In particular this means any instance with $p_1 \geq 9$ and density at most 0.75 can be scheduled in time $O(n^3)$. Fishburn and Lagarias find schedules for the remaining cases with small values for p_1 and density up to 0.75 based on the following idea.

They find a porous schedule for the first few elements and they extend this schedule using the results from Chan and Chin and Lemma 3 to find a schedule for the remaining machines that fits into the holes. For example for the instance $(3, 4, p_3, \dots)$ we can create a porous schedule 12_ as discussed above which has a hole at every third position. The remaining machines after removing machines 1 and 2 have density at most $\frac{3}{4} - \frac{1}{3} - \frac{1}{4} = \frac{1}{6}$, thus $p_i \geq 6$ for $i \geq 3$. For this pattern of holes we get the values for h_i by dividing the remaining periods by 3. For $p_3 = 6, 7, 8$ we get $h_3 = 2$ which increases the density by a factor of not more than 4, thus the density of the resulting instance is at most $\frac{1}{6} \cdot 4 < \frac{3}{4}$ with the first element of the new instance being 2. By the table of Chan and Chin this is schedulable. A similar argument holds for $p_3 \geq 9$ with the density of the new schedule being not more than $\frac{7}{10}$. They further refine this approach in the paper.

3 A 10/7-approximation for discrete BGT

We claim that $H = \sum_{i=1}^{n} v_i$ is a lower bound on the optimal value. As long as all bamboo have height at most $H' < H$ the sum of all bamboo heights increases by at least $H - H' > 0$ each step until it eventually exceeds nH'. Then there must be a bamboo with height more than H'. On the other hand, there is an algorithm that produces a schedule of height $2H$ [8]. Thus, the optimal value is indeed somewhere in the interval $[H, 2H]$.

Our algorithm is based on the following key result.

▶ **Lemma 5.** *Consider a pinwheel instance p and a porous schedule f for a subset $\mathcal{S}$ of the tasks. Let $\mathcal{U} := \{1, \dots, n\} \setminus \mathcal{S}$. If $\frac{h_i}{p_i} \geq D_{\mathcal{U}}$ for each $i \in \mathcal{U}$ in a porous schedule for $\mathcal{S}$ and $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ for all $i \in \mathcal{U}$, then $(p_1, \dots, p_n)$ is schedulable for Alg.*

Proof. If $\frac{h_i}{p_i} \geq D_{\mathcal{U}}$ then

$$\sum_{i \in \mathcal{U}} \frac{1}{h_i} \leq \frac{1}{D_{\mathcal{U}}} \sum_{i \in \mathcal{U}} \frac{1}{p_i} \leq \frac{1}{D_{\mathcal{U}}} \frac{3}{4} D_{\mathcal{U}} = \frac{3}{4}.$$

Thus the instance is schedulable by Lemma 3 and by Theorem 4. ◀

Our algorithm begins by defining periods for the pinwheel problem based on the growth speeds. These periods are then multiplied by a certain factor to get an instance that is easier to schedule. It would be easiest if we could multiply all periods by 4/3. That way we would get an instance with density at most 3/4 which is therefore schedulable by Theorem 4. However, the input for the pinwheel problem must be a set of natural numbers. This means that we must round down after scaling the periods. Even when choosing a scaling factor larger than 4/3 (in our case, 10/7), after rounding down there can still be some periods that

become too small, resulting in a density that is too high. This means that we cannot rely on the previous work as a black box. Typically, there will be a few difficult periods remaining. For instance, the period 4 becomes $\lfloor \frac{10}{7} \cdot 4 \rfloor = 5$ after scaling, and 5 becomes 7. Typically, longer periods are easier to handle as the condition $\lfloor \frac{10}{7} p^* \rfloor \geq \frac{4}{3} p^*$ is often satisfied.

We overcome this difficulty by finding a porous schedule for the few difficult periods by hand and then showing that the remaining periods can be scheduled in the holes. To do this we need to consider some inputs for the pinwheel problem that were not considered in previous works, since (as discussed) the overall density of the instance may still be above 3/4 at this point. Thus we need to find a porous schedule for the cases not covered by the table of Chan and Chin and a schedule for the remaining machines.

Algorithm 1 Binary Search for BGT.

Let $H = \sum_{i=1}^{n} v_i$ and $R = \frac{10}{7}$. Using binary search in the interval $[H, 2H]$ find the smallest K such that the following procedure returns a valid schedule and return this schedule.

1. Given K define $p_i^* = \lfloor \frac{K}{v_i} \rfloor$ and $p_i = \lfloor R p_i^* \rfloor$.
2. Solve the pinwheel problem $(p_1, \ldots, p_n)$.

In order to analyze Algorithm 1, we first show that whenever there exists a schedule for the pinwheel problem p^*, then the algorithm can find a schedule for p.

Assume that there exists a schedule for p^*. Then $d(p^*) \leq 1$. Whenever $\frac{p_i^*}{p_i} \leq \frac{3}{4}$ for all i we have $d(p) \leq \frac{3}{4}$ which means p is schedulable.

We can find schedules for cases that contain i with $\frac{p_i}{p_i^*} > \frac{3}{4}$ by assigning those periods to the subset $\mathcal{S}$ and applying Lemma 5. We create a porous schedule for $\mathcal{S}$ and define h_i for $i \in \mathcal{U} := \{1, \ldots, n\} \setminus \mathcal{S}$ as the minimum number of holes in p_i consecutive positions of the porous schedule. In some cases, the porous schedule is completely regular, with holes in every k-th location (and only there). In that case we get

$$\frac{h_i}{p_i} = \frac{\lfloor \frac{1}{k} \lfloor \frac{10}{7} p_i^* \rfloor \rfloor}{\lfloor \frac{10}{7} p_i^* \rfloor}$$

and for Lemma 5 we need to show that $h_i/p_i \geq D_{\mathcal{U}}$. If we write $p_i^* = 7a + b$ for some values a, b then this means showing that

$$\frac{5a + \lfloor \frac{1}{k} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a + \lfloor \frac{10}{7} b \rfloor} \geq D_{\mathcal{U}}.$$

▶ **Lemma 6.** *If there is a schedule of height K then the procedure in Algorithm 1 finds a schedule of height at most $\frac{10}{7}K$.*

Proof. Let $p_i^* = \lfloor \frac{K}{v_i} \rfloor$. Assume there is a schedule of height K. Then there is a solution to the pinwheel problem $p^* = (p_1^*, ..., p_n^*)$ and thus $\sum_{i=1}^{n} \frac{1}{p_i^*} \leq 1$. If $p_1^* = 1$ then there is only one plant and the schedule is trivial. Thus we consider $p_1^* \geq 2$. The algorithm calculates the periods $p_i = \lfloor \frac{10 p_i^*}{7} \rfloor$ and solves the resulting pinwheel problem. Because of the definition of p_i^* the resulting schedule has height at most $\frac{10K}{7}$.

We show that if a schedule of height K exists, there is always a schedule for the pinwheel problem of ALG. If $p_i^* \geq 11$ then $\frac{1}{p_i}$ is smaller then $\frac{1}{p_i^*}$ by a factor of at least $\frac{3}{4}$ since

$$\frac{1}{p_i} = \frac{1}{\lfloor 10/7 p_i^* \rfloor} \leq \frac{1}{(10/7 p_i^* - 1)} \leq \frac{3}{4} \frac{1}{p_i^*} \text{ for } p_i^* \geq 11.$$

The same is true for every other value of p_i^* except $p_i^* = 2$ and $p_i^* = 4$ which can easily be verified. Thus if there is no p_i^* with value 2 or 4 the density of the pinwheel problem is at most 3/4 and it can be scheduled. We now consider all cases where these values do occur and use Lemma 5 to show that the pinwheel problem can be solved.

We denote these cases by listing the initial values of the instance p^*. For example, the notation 4,4,5 means that we are considering the case where $p_1^* = p_2^* = 4$ and $p_3^* = 5$. In that case $p = (5, 5, 7, \dots)$ with zero or more plants after the first three.

For each case, we create a porous schedule for a subset $\mathcal{S}$ of periods in p. We then need to show that $\frac{h_i}{p_i} \geq D_{\mathcal{U}}$ as well as $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for all periods $\mathcal{U}$ that are not yet scheduled by the porous schedule. If that is the case, then p is schedulable by Lemma 5.

2 We have $p_i/p_i^* \geq 6/5$ for all p_i. This can easily be verified for $p_i^* = 2$ and $p_i^* = 4$ and it holds for all other periods because for those periods we have $p_i/p_i^* \geq 4/3$. This means the density in the pinwheel problem of ALG is at most 5/6. Since $p_1 = 2$ this problem can be solved.

3, 3, 4 and 3, 4, 4 In this case there are only three plants and the schedule is 123.

3, 4, $p_3^* \geq 5$ Then $p = (4, 5, p_3 \geq 7, \dots)$. ALG schedules plants 1 and 2 using the schedule 1_2_. Then $D_{\mathcal{U}} = 5/12$ and $\frac{h_i}{p_i} = \frac{\lfloor \frac{1}{2} \lfloor \frac{10}{7} p_i^* \rfloor \rfloor}{\lfloor \frac{10}{7} p_i^* \rfloor}$. Let $p^* = 7a + b$. Then $\frac{h_i}{p_i} = \frac{5a + \lfloor \frac{1}{2} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a + \lfloor \frac{10}{7} b \rfloor}$.

If $\frac{5a + \lfloor \frac{1}{2} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a + \lfloor \frac{10}{7} b \rfloor} \geq D_{\mathcal{U}}$ then $\frac{5(a+1) + \lfloor \frac{1}{2} \lfloor \frac{10}{7} b \rfloor \rfloor}{10(a+1) + \lfloor \frac{10}{7} b \rfloor} \geq D_{\mathcal{U}}$ since $\frac{5}{10} > D_{\mathcal{U}}$.

Therefore if the inequality holds for a particular p_i^*, then it also holds for all subsequent p_i^* that have the same remainder after division by 7. In particular, this means that if there are 7 consecutive periods p_i^* with $h_i/p_i \geq D_{\mathcal{U}}$ then $h_i/p_i \geq D_{\mathcal{U}}$ also holds for all subsequent periods p_i^*.

We can determine that $h_i/p_i \geq D_{\mathcal{U}}$ holds for $p_i^* \geq 5$ by looking at enough periods (in this case periods 5 to 11). This argument is repeated in many of the subsequent cases. Appendix A contains tables that show the values for each case.

Furthermore $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for $p_i^* \geq 5$. This means we can schedule this case.

4, 4 Then $p = (5, 5, \dots)$. ALG schedules plants 1 and 2 using the schedule 1_2__. Then $D_{\mathcal{U}} = 1/2$. For $p^* = 7a + b$ we get $\frac{h_i}{p_i} = \frac{6a + h(\lfloor \frac{10}{7} b \rfloor)}{10a + \lfloor \frac{10}{7} b \rfloor}$. By looking at 7 periods we can determine that $h_i/p_i \geq D_{\mathcal{U}}$ holds for $p_i^* \geq 4$.

Since $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for $p_i^* \geq 5$ but not $p_i^* = 4$ we still need to consider the case where $p_1^* = p_2^* = p_3^* = 4$ but we can schedule this case for $p_3^* \geq 5$.

4, 4, 4 Then $p = (5, 5, 5, \dots)$. ALG schedules plants 1 to 3 using the schedule 123__. Then $D_{\mathcal{U}} = 1/4$. Using the same approach as in the previous case we can determine that $h_i/p_i \geq D_{\mathcal{U}}$ holds for $p_i^* \geq 4$. As in the previous case this means we can schedule all cases except the case where $p_1^* = \dots = p_4^* = 4$ since in this case $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ is not guaranteed. However in this case there are only four plants and therefore this case can be scheduled as well.

In all subsequent cases, all periods except the first are greater than 4. This means $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for all periods after the first and we only need to verify $h_i/p_i \geq D_{\mathcal{U}}$. We get the following results.

p^*	p	Schedule	$D_{\mathcal{U}}$	p_i^*	h_i/p_i
$4, 5, \geq 9$	$5, 7$	1_2__	$11/20$	$-$	$-$
$4, 5, 6$	$5, 7, 8$	31__2 _13__ 21__3 _12__	$23/60$	$14a + b$	$\frac{10a+h(\lfloor \frac{10}{7} b \rfloor)}{20a+\lfloor \frac{10}{7} b \rfloor}$
$4, 5, 6, 6$	$5, 7, 8, 8$	see 3,4,5			
$4, 5, 7$	$5, 7, 10$	see next line			
$4, 5, 8$	$5, 7, 11$	1_3__ 12___ 1_32_ 1___2	$17/40$	$14a + b$	$\frac{11a+h(\lfloor \frac{10}{7} b \rfloor)}{20a+\lfloor \frac{10}{7} b \rfloor}$
$4, 6, \geq 9$	$5, 8$	1____ 1_2__ 1___2	$7/12$	$21a + b$	$\frac{20a+h(\lfloor \frac{10}{7} b \rfloor)}{30a+\lfloor \frac{10}{7} b \rfloor}$
$4, 6, 6$	$5, 8, 8$	see next line			
$4, 6, 7$	$5, 8, 10$	1__3_ 1_2__ 13__2	$37/84$	$21a + b$	$\frac{16a+h(\lfloor \frac{10}{7} b \rfloor)}{30a+\lfloor \frac{10}{7} b \rfloor}$
4,6,8	5,8,11	1__3_ 1_2__ 1__32 1____ 1_23_ 1___2	$\frac{11}{24}$	$21a + b$	$\frac{17a+h(\lfloor \frac{10}{7} b \rfloor)}{30a+\lfloor \frac{10}{7} b \rfloor}$
4,6,8,8	5,8,11,11	1__3_ 1_24_ 1__32 1__4_ 1_23_ 1__42	$\frac{1}{3}$	$21a + b$	$\frac{14a+h(\lfloor \frac{10}{7} b \rfloor)}{30a+\lfloor \frac{10}{7} b \rfloor}$
$4, 7$	$5, 10$	see next line			
$4, 8$	$5, 11$	1_2__1____	$5/8$	$7a + b$	$\frac{7a+h(\lfloor \frac{10}{7} b \rfloor)}{10a+\lfloor \frac{10}{7} b \rfloor}$
$4, \geq 9$	$5, \geq 12$	1____	$3/4$	$7a + b$	$\frac{8a+\lfloor \frac{4}{5} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a+\lfloor \frac{10}{7} b \rfloor}$

◀

▶ **Theorem 7.** *Algorithm 1 is a polynomial-time $\frac{10}{7}$-approximation.*

Proof. The approximation ratio follows directly from Lemma 6. Since any height K that is at least the optimal height is schedulable, the procedure in algorithm 1 finds a valid schedule of height $\frac{10}{7}K$ for any height that is at least Opt. This means the binary search settles on a value that is at most Opt (since it is integer) and the algorithm returns a schedule of height at most $\frac{10}{7}$ times that value.

The porous schedules are given in the paper of Fishburn and Lagarias (in particular see table 1 of [7]) and are thus available in $O(1)$. Then the algorithm in [4] with runtime $O(n^3)$ is used to schedule the remaining machines. This means pinwheel instances with density at most 0.75 can be solved in time $O(n^3)$. In addition, as mentioned above Fishburn and Lagarias show that instances with $p_1 = 2$ and density up to $\frac{5}{6}$ can also be solved in time $O(n^3)$. ◀

4 Continuous BGT on a star

We now examine the continuous case of the BGT-problem. For this problem Gasieniec et al. propose a $O(\log n)$-approximation algorithm that works in any metric space. We consider the star graph and show that many of the constant approximation factor algorithms for discrete BGT can be extended to this case with only small losses in the approximation ratio.

4.1 Notation

Plants $1, \ldots, n$ grow at the end of each of n branches on a star graph. We define d_i as the time it takes the server to travel to plant i cut it and return to the center. The growth rate of plant i is denoted by h_i. Each *round* the server visits a plant and returns back to the center.

A *schedule* for continuous BGT on a star graph, is an infinite sequence over $\{1, \ldots, n\}$, that describes the order plants are visited in. While BGT is an infinite problem, it is sufficient to consider cyclic schedules. This is because any schedule that maintains a finite maximum height of a must visit each plant i at least once in $\frac{a}{h_i b}$ rounds, where b is the minimum distance needed to visit a plant. This means a schedule for BGT (on a star) solves the pinwheel problem with periods $\frac{a}{h_i b}$. Since any pinwheel schedule is cyclic (see Theorem 2.1 in [10]) the same is true for BGT schedules.

4.2 A lower bound for the star

Consider a cyclic schedule of length L. Let m_i be the amount of times plant i is visited in a segment of length L. Then $\frac{L}{m_i}$ is the average period between visits of i.

▶ **Lemma 8.** *The average height of p after a cut is $h_p(d_p + \sum_{i \neq p} \frac{m_i}{m_p} d_i)$.*

Proof. Consider a plant p. Let $\lambda_1, \ldots, \lambda_{m_p}$ be the heights plant p reaches in the schedule. The sum of all heights is the same as the sum of all distances that are traversed times the speed of the plant, i.e. $\sum_{k=1}^{m_p} \lambda_k = h_p \sum_{i=1}^{n} d_i m_i$. The average height is

$$\frac{1}{m_p} \sum_{k=1}^{m_p} \lambda_k = \frac{1}{m_p} h_p \sum_{i=1}^{n} d_i m_i = h_p(d_p + \sum_{i \neq p} \frac{m_i}{m_p} d_i). \qquad \blacktriangleleft$$

Naturally the average height of p is a lower bound on the maximum height of p. This inspires the following model: The algorithm may visit each plant i at a certain fixed period f_i. These periods do not need to match a feasible schedule and may even be fractional. We only require that $\sum_{i=1}^{n} \frac{1}{f_i} = 1$ which means that the frequencies add up to one plant visited per round. The height a plant p reaches in this model is defined as $h_p(d_p + \sum_{i \neq p} \frac{f_p}{f_i} d_i)$.

By setting $f_i = \frac{L}{m_i}$ we get a solution for the new model with a height that is equal to the average height of a schedule in the original model and not more than the maximum height. This means a lower bound on the height in the new model is also a lower bound on the average as well as maximum height for continuous BGT on a star graph.

▶ **Lemma 9.** *$R := \sum_{i=1}^{n} h_i d_i$ is a lower bound on the optimal height for continuous BGT on a star graph.*

Proof. Consider the new model. We set $K = \sum_{i=1}^{n} h_i$ and $f_i = \frac{K}{h_i}$. Then $\sum_{i=1}^{n} \frac{1}{f_i} = 1$ and this is a valid solution to the new model. Then the height of plant p is $h_p(d_p + \sum_{i \neq p} \frac{f_p}{f_i} d_i) = h_p(d_p + \sum_{i \neq p} \frac{h_i}{h_p} d_i) = \sum_{i=1}^{n} h_i d_i = R$.

It follows that all plants reach a height of R. Because of the requirement $\sum_{i=1}^{n} \frac{1}{f_i} = 1$ it is not possible to reach a maximum height lower than R, since visiting one plant more often would mean visiting other plants less often. This means R is a lower bound on the maximum height in the new model and therefore also a lower bound on the maximum height for continuous BGT on a star graph. ◀

4.3 Algorithms on the star

The algorithm Reduce-Fastest(x), introduced by Gasieniec et al. [8], cuts the next fastest growing bamboo among those with height at least $x \cdot R$. The proof of the following lemma is structured in a way similar to the proof by J. Kuszmaul for discrete BGT [13]; see the appendix.

▶ **Lemma 10.** *Reduce-Fastest*$(2(R + Dh_{max}))$ *is a 6-approximation on the star.*

Proof. Assume that at some point there is a bamboo b_i that reaches height $3(R + Dh_{max})$. Let t_1 be the most recent time b_i reaches height $2(R + Dh_{max})$ and t_3 the time it reaches height $3(R + Dh_{max})$. Furthermore let t_2 be the first time in between t_1 and t_2 where the gardener visits the center. Since it takes at most distance D to visit a plant and return to the center the height of b_i is at most $2R + 3Dh_{max}$ at this time. We consider the set S of bamboo that are cut at least once during the time interval $[t_2, t_3)$. For bamboo $j \in S$, let m_j be the number of times the bamboo is cut in the interval $[t_2, t_3)$.

Since b_i already has height at least $2(R + Dh_{max})$ when the algorithm decides to cut j we have $h_j \geq h_i$ for all $j \in S$.

For $m_j \geq 2$ we have $h_j(t_3 - t_2) > 2(R + Dh_{max})(m_j - 1)$ or $\frac{h_j(t_3-t_2)}{2(m_j-1)} > R + Dh_{max}$ because bamboo b_j needs to grow to height $2(R+Dh_{max})$ at least $m_j - 1$ times in the interval in order to be cut m_j times. Meanwhile bamboo b_i grows by at most $R + Dh_{max}$ during this interval and therefore $R + Dh_{max} > h_i(t_3 - t_2)$. It follows that $h_j > 2(m_j - 1)h_i \geq m_j h_i$.

During the interval the algorithm visits each plant $b_j \in S$ a total of m_j times and travels distance d_j each time. Additionally distance $\frac{d_i}{2}$ is traversed to reach plant b_i.

$$\begin{aligned} t_3 - t_2 &= \sum_{b_j \in S} m_j d_j + \frac{d_i}{2} \\ &< \sum_{b_j \in S} \frac{h_j d_j}{h_i} + \frac{d_i}{2} = \frac{1}{h_i} \sum_{b_j \in S} h_j d_j + \frac{d_i}{2} \\ &\leq \frac{1}{h_i}(R - h_i d_i) + \frac{d_i}{2} = \frac{R}{h_i} - \frac{d_i}{2} \end{aligned}$$

This however means that the inverval is too short for plant b_i to grow by height R since this takes time $\frac{R}{h_i}$. This is a contradiction and thus no plant can reach height $3(R + Dh_{max}) \leq 6L$. ◀

Define $L = \max(R, Dh_1)$, where D is the diameter of the star and h_1 is the growth rate of the fastest bamboo. This is a lower bound on Opt.

The Deadline-Driven algorithm always cuts the bamboo with the earliest deadline among those with height at least L where the deadline is determined by some height the algorithm wants to maintain. This means that in each round the algorithm chooses the plant with the earliest deadline, travels towards this plant and cuts it before returning to the center. The algorithm does not turn around when a more urgent plant reaches height L.

We say a plant is *requested* when it reaches height $(1 + \sqrt{2})L$ and before that this request is *initialized* at the time the plant has height 0. This happens either after a cut or at the beginning of the process. We choose the *deadline* as the time a bamboo reaches height $(3 + 2\sqrt{2})L$.

This means each request to cut a plant consists of an initialization time, a request time and a deadline.

▶ **Lemma 11.** *The Deadline-Driven algorithm maintains a height of* $(3 + 2\sqrt{2})L$.

Proof. Assume plant i reaches height $(3 + \sqrt{2})L$ at time T and is not cut. We scale the time (and distance) such that $L/h_i = 1$ which means plant i grows by L in one timestep. This is possible because whenever we scale the length of all edges by a factor then the heights of all plants reached in any schedule is scaled by the same factor. This holds for both Alg and Opt which means the approximation ratio is unaffected. We can also see this as changing the timescale using L/h_i as our unit of time.

Let t_1 be the last time the algorithm is idle or busy processing a request with deadline after T. Let time 0 be the most recent time before t_1 where plant i has height 0.

This means between time t_1 and T the algorithm is only processing requests with deadlines at or before T and it is not idle. Let the set of these requests be S and let v be the earliest request time among all request in S. We have $i \in S$, so $v \leq 1 + \sqrt{2}$ and $t_1 \geq v$.

We further divide the set S into old requests S_0 which are initialised before v and new requests S_1 which are initialised after v. The time required to process S_0 is $\sum_{j \in S_0} d_j$ because any bamboo with an old request must be visited once to fulfill the request. Afterwards the bamboo either has a deadline after T and is not visited again or becomes part of the bamboo with new requests. We next show that all bamboo with new requests also have an old request. Consider the earliest new request for a bamboo. This new request gets initialized inbetween v and T. This means there was a previous request for that bamboo with a deadline between v and T. This request is an old request.

The time required to process S_1 is $\sum_{j \in S_1} m_j d_j$ where m_j is the number of new requests of bamboo j which are requests with initialization time after v and a deadline before or at T. (Here $j \in S$ means there is a request for bamboo j in the set of requests S. We may also see S as a multi-set of bamboo instead.)

It is possible that just before v a request with a deadline after T arrives and is processed by the algorithm. The algorithm traverses a distance of r to serve this request if it exists, otherwise $r = 0$. This means $t_1 \leq v + r$.

We now show that $\sum_{j \in S_0} d_j + \sum_{j \in S_1} m_j d_j + v + r < T$ which contradicts the assumption that plant i is not cut before time T.

We begin by finding upper bounds on the time required to process the requests in S.

A bamboo with an old request must grow by at least $(2 + \sqrt{2})L$ in time at most $T - v$ to have a deadline before T which means $h_j(T - v) \geq (2 + \sqrt{2})L$. Meanwhile plant i grows by $(T - v)L$ in time $(T - v)$, that is $h_i(T - v) = (T - v)L$. It follows that

$$\frac{h_j}{h_i} \geq \frac{2 + \sqrt{2}}{T - v} \text{ for } j \in S_0 \tag{1}$$

A bamboo with new requests must reach the threshold $(1 + \sqrt{2})L$ exactly m_j times in time at most $T - v$ in order to be requested m_j times before T. Then $h_j(T - v) \geq m_j(1 + \sqrt{2})$. It follows that

$$\frac{h_j}{h_i} \geq m_j \frac{1 + \sqrt{2}}{T - v} \text{ for } j \in S_1 \tag{2}$$

We can now find upper bounds for the processing times of the requests in S. We first get

$$\sum_{j \in S_0} d_j \overset{(1)}{\leq} \frac{1}{h_i} \frac{T - v}{2 + \sqrt{2}} \sum_{j \in S_0} d_j h_j < \frac{T - v}{2 + \sqrt{2}} \frac{R}{h_i}$$

and then

$$\sum_{j \in S_1} m_j d_j \overset{(2)}{\leq} \frac{1}{h_i} \frac{T - v}{1 + \sqrt{2}} \sum_{j \in S_0} d_j h_j < \frac{T - v}{1 + \sqrt{2}} \frac{R}{h_i}$$

Given the timescale and because $R \leq L$ it takes time less than $\frac{T-v}{2+\sqrt{2}}$ to process the old requests and less than $\frac{T-v}{1+\sqrt{2}}$ to process the new requests.

It remains to show $\frac{T-v}{2+\sqrt{2}} + \frac{T-v}{1+\sqrt{2}} + v + r \leq T$.

We have $r \leq L/h_1 \leq L/h_i = 1$ where the first inequality holds because otherwise plant 1 would grow by more than L in the time it takes to travel distance r and the second follows from the ordering of the plants by their growth rate. Furthermore since $v < 1 + \sqrt{2}$ and the earliest deadline of i is $3 + 2\sqrt{2}$ we have $T - v \geq (2 + \sqrt{2})$.

It follows

$$r \leq 1 = \left(\frac{1}{2+\sqrt{2}}\right)\left(2+\sqrt{2}\right) \leq \left(\frac{1}{2+\sqrt{2}}\right)(T-v)$$

and then

$$\begin{aligned}
& r \leq \left(\frac{1}{2+\sqrt{2}}\right)(T-v) \\
& \Rightarrow \left(\frac{1}{2+\sqrt{2}}\right)v + r \leq \left(\frac{1}{2+\sqrt{2}}\right)T \\
& \Rightarrow \left(\frac{1}{2+\sqrt{2}}\right)v + r \leq \left(\frac{1}{2+\sqrt{2}}\right)T + \left(1 - \frac{1}{2+\sqrt{2}}\right)T - \left(1 - \frac{1}{2+\sqrt{2}}\right)T \\
& \Rightarrow \left(1 - \frac{1}{2+\sqrt{2}}\right)T + \left(\frac{1}{2+\sqrt{2}}\right)v + r \leq T \\
& \Rightarrow \left(1 - \frac{1}{2+\sqrt{2}}\right)T - \left(1 - \frac{1}{2+\sqrt{2}}\right)v + v + r \leq T \\
& \Rightarrow \frac{T-v}{2+\sqrt{2}} + \frac{T-v}{1+\sqrt{2}} + v + r \leq T
\end{aligned}$$

This means we can process all requests (including the request for b_i with deadline at time T) before time T and plant i does not grow above height $(3 + 2\sqrt{2})L$. ◀

4.4 Extending the star graph

It is also possible to extend some of these results to an extended version of a star graph, where several plants are placed on each branch at different distances from the center. We again number the plants from 1 to n and the definition of h_i and d_i remains the same.

This means that d_i is the time it takes the server to travel the path leading up to node i from the center two times (once in both directions). Additionally we denote with δ_i the time it takes the server to travel the singular edge, connected to node i from direction of the center, two times (once in both directions). The definitions of d_i and δ_i are visualized in Figure 1.

We assume that the speeds are decreasing on each branch. (If there is a plant a that is further away than plant b on a branch and has the same speed or more, then plant b gets visited whenever plant a is visited.)

For the star graph with multiple plants on each branch, there is more than one possibility to accurately represent the schedule. One possibility is to list each visit to a plant, but it is also possible to omit plants that are visited by the server in passing, as it travels to a plant further on the branch. The representation we will be using, lists all plants that are visited in a round, but omits the second visit, where the server passes the plant again on its way to the center.

▶ **Lemma 12.** *The value $\frac{1}{2}R$ with $R = \sum_{i=1}^{n} h_i\delta_i$ is a lower bound on the optimal height for continuous BGT on a star graph with multiple plants on each branch.*

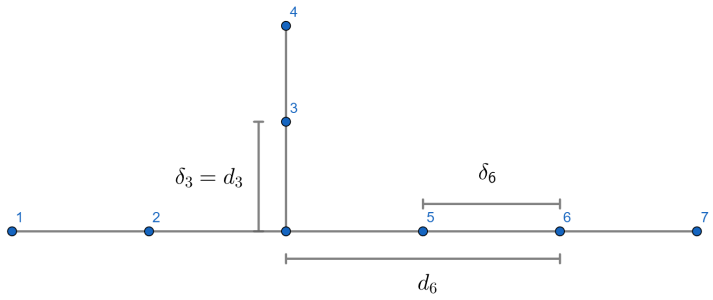

Figure 1 Multiple plants on each branch: example for values d_i and δ_i.

For the star with multiple plants on each branch we use the Deadline-Driven algorithm with a small modification. Whenever the algorithm visits a bamboo b_j on a branch it walks double the distance and cuts all additional bamboo it encounters. These bamboo can be removed from the graph since they are always cut together with b_j. Then the distance between b_j and the next bamboo b_i (i.e δ_i) on the branch is at least as much as the distance from b_j to the center (i.e. $d_i - \delta_i$). In particular it follows that $\delta_i \geq d_i - \delta_i$ and thus $d_i \leq 2\delta_i$.

Furthermore for the bamboo with multiple plants our lower bound is $\frac{1}{2}R$ with $R := \sum_{i=1}^{n} h_i\delta_i$ and therefore we define $L = \max(\frac{1}{2}R, Dh_1)$ to get a lower bound on Opt. For the star with multiple plants on each branch the deadline is the point in time a plant reaches height $(9+2\sqrt{5})L$ and it gets requested at height $(4+2\sqrt{5})L$. The initialization time remains the same.

▶ **Lemma 13.** *The modified Deadline-Driven algorithm maintains a height of* $(9+2\sqrt{5})L$.

5 Conclusions

It is possible to achieve a better approximation ratio by setting R in algorithm 1 to a lower value but this requires a much larger case analysis. This is because the periods p_i will be smaller. Using some computer assistance it is for example possible to achieve a ratio of $\frac{7}{5}$ but the length of the proof encourages finding new ideas. Using more cases, we could potentially get even closer to the ratio 4/3, but we could never reach it as long as we can only rely on pinwheel instances with density at most 3/4 being schedulable.

It should be noted that the pinwheel instances that our algorithm encounters are not general ones. For instance, the 10/7-approximation never encounters the period 6. It is conceivable that for the limited set of pinwheel instances that need to be considered, the density guarantee could be improved. Finally, of course the conjecture by Chan and Chin that all instances with density at most 5/6 are schedulable is still open. If that were proved, we could potentially get close to 6/5 (instead of only to 4/3). An approximation scheme currently seems out of reach.

References

1 Michael A. Bender, Sándor P. Fekete, Alexander Kröller, Vincenzo Liberatore, Joseph S. B. Mitchell, Valentin Polishchuk, and Jukka Suomela. The minimum backlog problem. *Theor. Comput. Sci.*, 605:51–61, 2015. doi:10.1016/j.tcs.2015.08.027.

2 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. doi:10.4230/LIPIcs.FUN.2021.5.

3 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. doi:10.1007/978-3-642-33475-7_5.

4 Mee Yee Chan and Francis Y. L. Chin. General schedulers for the pinwheel problem based on double-integer reduction. *IEEE Trans. Computers*, 41(6):755–768, 1992. doi:10.1109/12.144627.

5 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. doi:10.1007/3-540-48224-5_70.

6 Wei Ding. A branch-and-cut approach to examining the maximum density guarantee for pinwheel schedulability of low-dimensional vectors. *Real Time Syst.*, 56(3):293–314, 2020. doi:10.1007/s11241-020-09349-w.

7 Peter C. Fishburn and J. C. Lagarias. Pinwheel scheduling: Achievable densities. *Algorithmica*, 34(1):14–38, 2002. doi:10.1007/s00453-002-0938-9.

8 Leszek Gasieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2017. doi:10.1007/978-3-319-51963-0_18.

9 Leszek Gasieniec, Benjamin Smith, and Sebastian Wild. Towards the 5/6-density conjecture of pinwheel scheduling. In Cynthia A. Phillips and Bettina Speckmann, editors, *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2022, Alexandria, VA, USA, January 9-10, 2022*, pages 91–103. SIAM, 2022. doi:10.1137/1.9781611977042.8.

10 Robert Holte, Aloysius Mok, Louis Rosier, Igor Tulchinsky, and Donald Varvel. Pinwheel: a real-time scheduling problem. In *Proceedings of the Hawaii International Conference on System Science*, pages 693–702 vol.2, 1989. doi:10.1109/HICSS.1989.48075.

11 Robert Holte, Louis E. Rosier, Igor Tulchinsky, and Donald A. Varvel. Pinwheel scheduling with two distinct numbers. *Theor. Comput. Sci.*, 100(1):105–135, 1992. doi:10.1016/0304-3975(92)90365-M.

12 John Kuszmaul. Bamboo trimming revisited: Simple algorithms can do well too. *CoRR*, abs/2201.07350, 2022. arXiv:2201.07350.

13 William Kuszmaul. Achieving optimal backlog in the vanilla multi-processor cup game. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1558–1577. SIAM, 2020. doi:10.1137/1.9781611975994.96.

14 Martijn van Ee. A 12/7-approximation algorithm for the discrete bamboo garden trimming problem. *CoRR*, abs/2004.11731, 2020. arXiv:2004.11731.

A Tables for checking $h_i/p_i \geq D_{\mathcal{U}}$ in the proof of Lemma 6

Case $3, 4, p_3^* \geq 5$ with schedule 1_2_:

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
5	7	3	0.429	0.417
6	8	4	0.5	
7	10	5	0.5	
8	11	5	0.45	
9	12	6	0.5	
10	14	7	0.5	
11	15	7	0.47	
12	17	8	0.47	

Case $4, 4$ with schedule 1_2__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
4	5	3	0.6	0.5
5	7	4	0.57	
6	8	4	0.5	
7	10	6	0.6	
8	11	6	0.54	
9	12	7	0.58	
10	14	8	0.57	
11	15	9	0.6	

Case $4, 4, 4$ with schedule 123__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
4	5	2	0.4	0.25
5	7	2	0.28	
6	8	2	0.25	
7	10	4	0.4	
8	11	4	0.36	
9	12	4	0.33	
10	14	5	0.36	
11	15	6	0.4	

Case $4, 5$ with schedule 1_2__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
9	12	7	0.5833333333	0.55
10	14	8	0.5714285714	
11	15	9	0.6	
12	17	10	0.5882352941	
13	18	10	0.5555555556	
14	20	12	0.6	
15	21	12	0.5714285714	
16	22	13	0.5909090909	

Case $4, 5, 6$ with schedule 31__2 _13__ 21__ 3 _12__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
7	10	5	0.5	0.38
8	11	5	0.45	
9	12	5	0.42	
10	14	6	0.43	
11	15	7	0.47	
12	17	8	0.47	
13	18	8	0.44	
14	20	10	0.5	
15	21	10	0.48	
16	22	10	0.45	
17	24	11	0.45	
18	25	12	0.48	
19	27	13	0.48	
20	28	13	0.46	
21	30	15	0.5	

Case $4, 5, 8$ with schedule 1_3__ 12___ 1_32_ 1___2

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
7	10	5	0.5	0.425
8	11	5	0.45	
9	12	6	0.5	
10	14	7	0.5	
11	15	7	0.47	
12	17	8	0.47	
13	18	9	0.5	
14	20	11	0.55	
15	21	11	0.52	
16	22	11	0.5	
17	24	12	0.5	
18	25	13	0.52	
19	27	14	0.525	
20	28	14	0.5	
21	30	16	0.53	

Case $4, 6$ with schedule 1____ 1_2__ 1___2

p_i^*	p_i	h_i	h/p	$D_{\mathcal{U}}$
9	12	7	0.58	0.583
10	14	9	0.64	
11	15	10	0.67	
12	17	10	0.59	
13	18	11	0.61	
14	20	13	0.65	
15	21	13	0.62	
16	22	14	0.64	
17	24	15	0.63	
18	25	16	0,64	
19	27	17	0.63	
20	28	18	0.64	
21	30	20	0.67	
22	31	20	0.65	
23	32	20	0.63	
24	34	22	0.65	
25	35	23	0.66	
26	37	24	0.65	
27	38	25	0.66	
28	40	26	0.65	
29	41	26	0.63	
30	42	27	0.649	

Case $4, 6, 7$ with schedule 1__3_ 1_2__ 13__2

p^*	p	h	h/p	D_A
7	10	6	0.6	0.440
8	11	6	0.54	0.458
9	12	7	0.58	0.458
10	14	8	0.57	
11	15	8	0.53	
12	17	10	0.58	
13	18	10	0.55	
14	20	11	0.55	
15	21	12	0.57	
16	22	12	0.55	
17	24	14	0.58	
18	25	14	0.56	
19	27	15	0.56	
20	28	16	0.57	
21	30	16	0.53	
22	31	17	0.55	
23	32	18	0.56	
24	34	19	0.56	
25	35	19	0.54	
26	37	20	0.54	
27	38	21	0.55	
28	40	22	0.55	

Case $4, 6, 8$ with schedule 1__3_ 1_2__ 1__32 1____ 1_23_ 1___2

p^*	p	h	h/p	D_A
9	12	6	0.5	0.458
10	14	7	0.5	
11	15	8	0.53	
12	17	8	0.47	
13	18	9	0.5	
14	20	11	0.55	
15	21	11	0.52	
16	22	12	0.55	
17	24	12	0.5	
18	25	13	0.52	
19	27	14	0.52	
20	28	15	0.54	
21	30	17	0.57	
22	31	17	0.55	
23	32	17	0.53	
24	34	18	0.53	
25	35	19	0.54	
26	37	20	0.54	
27	38	21	0.55	
28	40	22	0.55	
29	41	22	0.53	
30	42	23	0.546	

Case $4, 6, 8, 8$ with schedule 1__3_ 1_24_ 1__ 32 1__4_ 1_23_ 1__42

p^*	p	h	h/p	D_A
4	5	2	0.4	0.333
5	7	3	0.43	
6	8	3	0.375	
7	10	4	0.4	
8	11	4	0.36	
9	12	5	0.42	
10	14	6	0.43	
11	15	7	0.47	
12	17	7	0.41	
13	18	7	0.39	
14	20	9	0.45	
15	21	9	0.43	
16	22	10	0.45	
17	24	10	0.42	
18	25	11	0.44	
19	27	12	0.44	
20	28	12	0.43	
21	30	14	0.47	
22	31	14	0.45	
23	32	14	0.44	
24	34	15	0.44	
25	35	16	0.46	

Case $4, 8$ with schedule 1_2__ 1____

p^*	p	h	h/p	D_A
6	8	5	0.625	0.625
7	10	7	0.7	
8	11	7	0.64	
9	12	8	0.67	
10	14	9	0.64	
11	15	10	0.67	
12	17	11	0.65	
13	18	12	0.67	

B Multiple plants on each branch

We now find a lower bound for continuous BGT on a star graph with multiple plants on each branch. In this scenario, we make a distinction between cuts where the server comes from the center, and cuts where the server comes from the direction opposite the center. For both types of cuts, we find a lower bound on the maximum height a plant reaches in a particular schedule.

Consider plant p, and let A be the set of plants that are on the same branch but further from the center, and B the set of plants that are closer to the center on the same branch or on another branch.

Since the algorithm needs to visit the end of the branch at some point, p reaches a height of at least $h_p \sum_{i \in A} \delta_i$. This happens as the server is coming from the direction opposite the center and this is a lower bound on the maximum height.

Next, we develop a lower bound based on the cuts where the server comes from the direction of the center. Consider a cyclic schedule of length L, where only the first cut of each round, where the server comes from the center, is listed in this schedule. Each round the algorithm visits a branch and possibly multiple plants. We define m_i as the amount of rounds plant i is visited in. Then $\frac{L}{m_i}$ is the average period between visits of plant i, but only counting the first cut each round where the server comes from the direction of the center.

▶ **Lemma 14.** *The average height of p after a cut, with the server coming from the direction of the center, is $h_p(\delta_p + \sum_{i \in B} \frac{m_i}{m_p} \delta_i)$.*

Proof. Consider a plant p. Let $\lambda_1, \ldots, \lambda_{m_p}$ be the heights plant p reaches in the schedule, while only considering cuts coming from the center. The sum of these heights is the same as the sum of all distances travelled to visit plants in set $B \cup \{p\}$ multiplied by the speed of the plant. This means $\sum_{k=1}^{m_p} \lambda_k = h_p \sum_{i \in B \cup \{p\}} \delta_i m_i$. The average height is

$$\frac{1}{m_p} \sum_{k=1}^{m_p} \lambda_k = \frac{1}{m_p} h_p \sum_{i \in B \cup \{p\}} \delta_i m_i = h_p(\delta_p + \sum_{i \in B} \frac{m_i}{m_p} \delta_i) \qquad \blacktriangleleft$$

We now adapt our model. The algorithm may visit each plant i at a certain fixed period f_i. Let C be the set of all plants, that are closest to the center on their branch. We require $\sum_{i \in C} \frac{1}{f_i} = 1$, which means that the frequencies add up to one branch visited each round. Furthermore, the periods of plants on the same branch should be non decreasing as distance to the center increases. We define the height of a plant p as $\max(h_p(\sum_{i \in A} \delta_i, h_p(\delta_p + \sum_{i \in B} \frac{f_p}{f_i} \delta_i))$.

By setting $f_i = \frac{L}{m_i}$ we get a solution for the new model with a height that is equal to the average height of a schedule in the original model and not more than the maximum height. A lower bound on the maximum height in this new model is then a lower bound on the maximum height for continuous BGT on a star graph with multiple plants on each branch.

▶ **Lemma 12.** *The value $\frac{1}{2}R$ with $R = \sum_{i=1}^{n} h_i \delta_i$ is a lower bound on the optimal height for continuous BGT on a star graph with multiple plants on each branch.*

Proof. Consider the new model. We set $K = \sum_{i \in C}^{n} h_i$ and $f_i = \frac{K}{h_i}$. Let H_p be the maximum height that plant p reaches in the new model. By the definition of height we get $H_p \geq h_p \sum_{i \in A} \delta_i$ as well as $H_p \geq h_p(d_p + \sum_{i \in B} \frac{f_p}{f_i} d_i)$. Then

$$2H_p \geq h_p \sum_{i \in A} \delta_i + h_p(d_p + \sum_{i \in B} \frac{f_p}{f_i} \delta_i) = \sum_{i \in A} h_p \delta_i + h_p \delta_p + \sum_{i \in B} h_i \delta_i \geq \sum_{i=1}^{n} h_i \delta_i$$

The last inequality follows from the fact that the speeds are decreasing with distance to the center and therefore $h_p \geq h_i$ for $i \in A$. It follows that all plants reach a height of at least $\frac{1}{2}R$ and again it is not possible to achieve a lower height on all plants, since visiting one plant more often increases the height on other plants. ◀

▶ **Lemma 13.** *The modified deadline driven algorithm maintains a height of $(9 + 4\sqrt{5})L$.*

Proof. Assume plant i reaches height $(9+4\sqrt{5})L$ at time T. We scale the time (and distance) such that $L/h_i = 1$ which means plant i grows by L in one timestep. The old and new requests S_0 and S_1 as well as v and r are defined analogously to the proof for the star.

The time required to process S_0 is at most $\sum_{j\in S_0} d_j$ because any bamboo with an old request must be visited once to fulfill the request and in the worst case we start from the center traversing distance d_j to fulfill the request. It is possible that the algorithm is more efficient whenever plants with consecutive deadlines lie behind each other on the same branch.

Similarly the time required to process S_1 is at most $\sum_{j\in S_1} m_j d_j$ where m_j is the number of new requests on plant b_j.

Therefore in order to arrive at a contradiction we again need to show $\sum_{j\in S_0} d_j + \sum_{j\in S_1} m_j d_j + v + r < T$ which means plant i can be cut before time T.

Again we find upper bounds on the time required to process the old and new requests. These bounds are slightly different due to the changes in the algorithm as well as the lower bound.

A bamboo with an old request must grow by at least $(5+2\sqrt{5})L$ in time at most $T-v$ to have a deadline before T which means $h_j(T-v) \geq (5+2\sqrt{5})L$. Meanwhile plant i grows by $(T-v)L$ in time $(T-v)$, that is $h_i(T-v) = (T-v)L$. It follows that

$$\frac{h_j}{h_i} \geq \frac{5+2\sqrt{5}}{T-v} \text{ for } j \in S_0 \tag{3}$$

A bamboo with new requests must grow by $(4+2\sqrt{5})L$ exactly m_j times in time at most $T-v$ in order to have m_j deadlines before T. Then $h_j(T-v) \geq m_j(4+2\sqrt{5})$. It follows that

$$\frac{h_j}{h_i} \geq m_j \frac{4+2\sqrt{5}}{T-v} \text{ for } j \in S_1 \tag{4}$$

We now get

$$\sum_{j\in S_0} d_j \overset{(3)}{\leq} \frac{1}{h_i}\frac{T-v}{5+2\sqrt{5}} \sum_{j\in S_0} d_j h_j \leq \frac{1}{h_i}\frac{T-v}{5+2\sqrt{5}} \sum_{j\in S_0} 2\delta_j h_j < \frac{T-v}{5+2\sqrt{5}}\frac{2R}{h_i} \leq \frac{T-v}{5+2\sqrt{5}}\frac{4L}{h_i}$$

where we are using $d_i \leq 2\delta_i$ in the second inequality and $R \leq 2L$ in the last inequality. Analogously the following holds for the time required to process the new requests

$$\sum_{j\in S_1} m_j d_j \overset{(4)}{\leq} \frac{1}{h_i}\frac{T-v}{4+2\sqrt{5}} \sum_{j\in S_0} d_j h_j \leq \frac{1}{h_i}\frac{T-v}{4+2\sqrt{5}} \sum_{j\in S_0} 2\delta_j h_j < \frac{T-v}{4+2\sqrt{5}}\frac{2R}{h_i} \leq \frac{T-v}{4+2\sqrt{5}}\frac{4L}{h_i}$$

Given the timescale this means it takes time at most $4\frac{T-v}{5+2\sqrt{5}}$ to process the old requests and at most $4\frac{T-v}{4+2\sqrt{5}}$ to process the new requests. Furthermore since $v < 4+2\sqrt{5}$ and the earliest deadline of i is $9+4\sqrt{5}$ we have $T-v \geq (5+2\sqrt{5})$. It follows

$$\begin{aligned}
&r \leq 1 = \left(\frac{1}{5+2\sqrt{5}}\right)(5+2\sqrt{5}) \leq \left(\frac{1}{5+2\sqrt{5}}\right)(T-v)\\
&\Rightarrow \left(\frac{1}{5+2\sqrt{5}}\right)v + r \leq \left(\frac{1}{5+2\sqrt{5}}\right)T\\
&\Rightarrow \left(\frac{1}{5+2\sqrt{5}}\right)v + r \leq \left(\frac{1}{5+2\sqrt{5}}\right)T + \left(1-\frac{1}{5+2\sqrt{5}}\right)T - \left(1-\frac{1}{5+2\sqrt{5}}\right)T\\
&\Rightarrow \left(1-\frac{1}{5+2\sqrt{5}}\right)T + \left(\frac{1}{5+2\sqrt{5}}\right)v + r \leq T\\
&\Rightarrow \left(1-\frac{1}{5+2\sqrt{5}}\right)T - \left(1-\frac{1}{5+2\sqrt{5}}\right)v + v + r \leq T\\
&\Rightarrow 4\frac{T-v}{5+2\sqrt{5}} + 4\frac{T-v}{4+2\sqrt{5}} + v + r \leq T
\end{aligned}$$

This means we can process all requests (including the request for b_i with deadline at time T) before time T and plant i does not grow above height $(9+4\sqrt{5})L$. ◀