A Computer-Assisted Proof of the Optimal Density Bound for Pinwheel Covering

Akitoshi Kawamura
Research Institute for Mathematical Sciences, Kyoto University, Japan

Yusuke Kobayashi
Research Institute for Mathematical Sciences, Kyoto University, Japan

Abstract

In the covering version of the pinwheel scheduling problem, a daily task must be assigned to agents under the constraint that agent i can perform the task at most once in any a_i-day interval. In this paper, we determine the optimal constant $\alpha^* = 1.264\ldots$ such that every instance with $\sum_i 1/a_i \geq \alpha^*$ is schedulable. This resolves an open problem posed by Kawamura and Soejima (2020). Our proof combines Kawamura's (2026) techniques for the packing version with new mathematical insights to reduce the analysis to a finite set of instances, which are then verified through an exhaustive computer-aided search that draws on ideas from Gąsieniec, Smith, and Wild (2022). The same result was obtained independently by Mishra (2026).

2012 ACM Subject Classification Mathematics of computing → Combinatorics

Keywords and phrases pinwheel scheduling, pinwheel covering, density threshold

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.83

Related Version *Preprint*: `https://arxiv.org/abs/2510.06533` [8]

Supplementary Material *Software (Source Code)*: `https://github.com/a7kawamura/pinwheel_solver` [5], archived at `swh:1:dir:df76d28327c884f9cc2d178b6e1c1a4de8675761`

Funding *Akitoshi Kawamura*: Support Center for Advanced Telecommunications Technology Research (SCAT); Kyoto University Foundation.
Yusuke Kobayashi: JSPS KAKENHI Grant Numbers JP22H05001, 24K02901, 26K21777; JST ERATO Grant Number JPMJER2301.

1 Introduction

There is a task that must be performed every day, and it needs to be distributed among k agents. Each agent $i \in [k] = \{1, \ldots, k\}$ has an associated number a_i, called its *period*, and may be assigned the task at most once in any interval of a_i consecutive days. Under this constraint, we want to find a schedule that allows the task to be performed indefinitely. This problem is known as the *covering version of pinwheel scheduling*, or simply *pinwheel covering* [9, 11]. It was originally introduced as *point patrolling* [10], but here we prefer the terminology reflecting a contrast with the *packing* version of pinwheel scheduling [4], in which the constraint is that each agent i must be assigned the task *at least* once in any a_i-day interval.

Formally, an instance of pinwheel covering is a nonempty array $(a_i)_{i\in[k]}$ of positive integers, which we assume to be arranged in non-decreasing order, and we seek to find a *schedule* $S\colon \mathbb{Z} \to [k]$ (with $S(t)$ specifying the agent that works on day t) satisfying, for all $i \in [k]$, the following *frequency condition*, where S^{-1} denotes the inverse image of S:

$$\left|[m, m + a_i) \cap S^{-1}(i)\right| \leq 1 \text{ for each } m \in \mathbb{Z}.$$

An instance for which a schedule exists is said to be *covering-schedulable* (or simply *schedulable*). For example, $(2, 2)$, $(2, 4, 8, 8)$, and $(3, 5, 5, 5, 7)$ are schedulable, with a schedule S for the last one given by

$$S(t) = \begin{cases} 1 & \text{for } t \equiv 0, 3, 7, 10, 14, 17, \\ 2 & \text{for } t \equiv 1, 6, 11, 16, \\ 3 & \text{for } t \equiv 2, 8, 13, 18, \\ 4 & \text{for } t \equiv 4, 9, 15, 20, \\ 5 & \text{for } t \equiv 5, 12, 19 \pmod{21}. \end{cases}$$

For an instance $A = (a_i)_{i\in[k]}$ to be schedulable, its *density*

$$\mathrm{D}(A) = \sum_{i\in[k]} \frac{1}{a_i}$$

must necessarily be at least 1, because $\mathrm{D}(A)$ is the amount of workforce per day that the k agents can provide. This condition is not sufficient: $(2, 3, 5)$ is unschedulable, though $\frac{1}{2} + \frac{1}{3} + \frac{1}{5} \geq 1$. In fact, these three agents cannot even perform the task for just eight consecutive days: whichever of the eight days the period-5 agent covers, there are some four consecutive days left, which cannot be covered by the other two agents. This argument can be applied recursively, so that for every k, the instance $(2^{i-1} + 1)_{i\in[k]}$ is unschedulable, even for 2^k days [10, Theorem 17]. The density of this instance approaches

$$\alpha^* = \sum_{i=1}^{\infty} \frac{1}{2^{i-1} + 1} = 1.264\ldots$$

as $k \to \infty$. It has been suspected [10, Conjecture 18] (and remained open [6, Section 5]) that these are the densest unschedulable instances. We confirm this conjecture, improving on the best known bound of $1.545\ldots$ [10, Theorem 16]:

▶ **Theorem 1.** *Every instance A consisting of positive integers and satisfying $\mathrm{D}(A) \geq \alpha^*$ is covering-schedulable.*

We note that Theorem 1 was also obtained independently by Mishra [13]. Compared to [13], our proof seems to be simpler: apart from the computer-assisted part, the theoretical arguments (Section 2) fit within a few pages.

The explicit mention of the integrality of periods in Theorem 1 is because we will later extend the definitions and consider real-valued (non-integer) periods. This is an adaptation of the idea used in the recent proof by Kawamura [7] of an analogous optimal bound for the packing version of the problem, which we will review briefly below (Theorem 2).

Like this packing version, our proof of Theorem 1 involves exhaustive computer search for schedules of finitely many instances. Thus, the rest of the paper consists of the theoretical part (Section 2), which proves why checking this finite set of instances suffices, and the experimental part (Section 3) describing the techniques we used to schedule this finite but huge set of instances.

Related Work: Packing Version

As we mentioned, a better-studied problem is the packing version of pinwheel scheduling [4], where each agent i must be employed *at least* once every a_i days. Formally, an instance $A = (a_i)_{i\in[k]}$ is called *packing-schedulable* if there is $S\colon \mathbb{Z} \to [k]$ that satisfies the following for all $i \in [k]$:

$$\left|[m, m + a_i) \cap S^{-1}(i)\right| \geq 1 \text{ for each } m \in \mathbb{Z}.$$

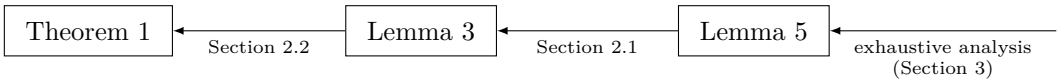

Figure 1 Outline of the proof of Theorem 1.

A packing-schedulable instance must have density at most 1. Conversely, there has been extensive research on sufficient conditions on the density that guarantee packing-schedulability. Building on prior work [1–3, 12], Kawamura [7] recently established the optimal threshold:

▶ **Theorem 2** (Kawamura [7, Theorem 1]). *Every instance A consisting of positive integers and satisfying $\mathrm{D}(A) \leq \frac{5}{6}$ is packing-schedulable.*

Our proof of Theorem 1 will partly mirror that of Theorem 2, augmented by a new idea. To highlight the contrast, we here outline the proof of Theorem 2 given in [7]. A key idea is to extend the problem to the setting where periods are positive real numbers, not just integers. An algorithm is then given [7, Lemma 5] that, for any integer $\theta \geq 1$, converts any packing-unschedulable instance A consisting of positive integers into a packing-unschedulable instance B consisting of periods in $\{1, 2, \ldots, \theta\} \cup (\theta, 2\theta]$ and satisfying $\mathrm{D}(B) < \mathrm{D}(A) + 1/(2\theta)$. It thus suffices to prove that for some integer $\theta \geq 1$,

> every instance $B = (b_i)_{i \in [k]}$ such that $b_i \in \{1, 2, \ldots, \theta\} \cup (\theta, 2\theta]$ for each $i \in [k]$ and $\mathrm{D}(B) < \frac{5}{6} + 1/(2\theta)$ is packing-schedulable.

This claim holds for $\theta = 11$, as was shown by an exhaustive computer analysis [7, Lemma 6].

2 Proof

The structure of the proof of Theorem 1 is shown in Figure 1. We first prove its special case where we do not have an agent with period 2:

▶ **Lemma 3.** *Every instance A consisting of integers > 2 and satisfying $\mathrm{D}(A) \geq \alpha^*$ is schedulable.*

2.1 Instances Without Period 2 (Proof of Lemma 3)

This part follows the same argument as the proof of Theorem 2 above. We start by extending the problem to the real-valued setting [9]: a nonempty array $(a_i)_{i \in [k]}$ of positive real numbers is *covering-schedulable* (or simply *schedulable*) if there exists $S \colon \mathbb{Z} \to [k]$ such that for all $i \in [k]$,

$$\left|[m, m + \lfloor r \cdot a_i \rfloor) \cap S^{-1}(i)\right| \leq r \text{ for each } m \in \mathbb{Z} \text{ and } r \in \mathbb{N}.$$

Note that this condition boils down to the one in Section 1 when a_i is an integer.

The following can be seen as the counterpart of the algorithm used for the packing problem at the end of Section 1.

▶ **Lemma 4** (Kawamura, Kobayashi, and Kusano [9, Lemma 6]). *Let $\theta \geq 1$, and let A be an unschedulable instance. We can convert A into another unschedulable instance B such that*

- *any element in B with a value $\leq \theta$ is in A as well,*
- *any element in B is at most 2θ, and*
- $\mathrm{D}(B) \geq \mathrm{D}(A) - 1/\theta$.

The first two claims about B in this lemma imply that if A consists of integers > 2 and θ is an integer > 2, then each element in B belongs to $\{3, 4, \dots, \theta\} \cup (\theta, 2\theta]$. Therefore, just as in the packing version, to prove Lemma 3, it suffices to prove the following for some integer $\theta > 2$:

($\star$) every instance $B = (b_i)_{i\in[k]}$ such that $b_i \in \{3, 4, \dots, \theta\} \cup (\theta, 2\theta]$ for each $i \in [k]$ and $\mathrm{D}(B) \geq \alpha^* - 1/\theta$ is schedulable.

As it turns out, this claim holds for $\theta = 10$. In fact, the instance $C = (\lceil b_i \rceil)_{i\in[k]}$ obtained by rounding up each period in B is still schedulable, as stated in the following lemma. Note that although C usually has density a bit lower than B, it satisfies $\mathrm{D}'(C) \geq \mathrm{D}(B) \geq \alpha^* - \frac{1}{10}$, where

$$\mathrm{D}'\big((c_i)_{i\in[k]}\big) = \sum_{i\in[k]} \begin{cases} 1/c_i & \text{for } c_i \leq 10, \\ 1/(c_i - 1) & \text{for } c_i > 10. \end{cases}$$

▶ **Lemma 5.** *Every instance C consisting of periods in $\{3, 4, \dots, 20\}$ and satisfying $\mathrm{D}'(C) \geq \alpha^* - \frac{1}{10}$ is schedulable.*

Note that this lemma is essentially about finitely many instances, because instances with ≥ 20 agents with periods ≤ 20 are trivially schedulable. It can thus be verified exhaustively by computer; see Section 3. We remark that replacing 10 in Lemma 5 (and in the definition of D') by a smaller number, say 9 (and accordingly replacing 20 by 18), would make it false. For example, the instance $(3, 4, 10, 10, 10, 12, 13, 17)$ is unschedulable (as can be checked by brute force computer search), even though $\frac{1}{3} + \frac{1}{4} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{11} + \frac{1}{12} + \frac{1}{16} = \frac{203}{176} > \alpha^* - \frac{1}{9}$.

We have thus proved Lemma 3. Note that the same approach alone cannot prove Theorem 1 directly: no integer θ satisfies the claim ($\star$) if $\{3, 4, \dots, \theta\}$ is replaced by $\{2, 3, \dots, \theta\}$, as can be seen by the unschedulable (see Section 1) instance $(2^{i-1} + 1)_{i\in[k]}$ for the largest k with $2^{k-1} + 1 \leq 2\theta$.

2.2 Eliminating Period 2 (Proof of Theorem 1 Assuming Lemma 3)

Now that we have proved Lemma 3, we are left with (integral) instances that have an agent with period 2. Luckily, such an instance readily reduces to a smaller one:

▶ **Lemma 6.** *An instance $(a_i)_{i\in[k]}$ with $a_1 = 2$ is schedulable if $(\lceil a_{i+1}/2 \rceil)_{i\in[k-1]}$ is.*

Proof. Given a schedule $S' \colon \mathbb{Z} \to [k-1]$ for the latter, we can obtain a schedule $S \colon \mathbb{Z} \to [k]$ for the former by inserting the period-2 agent every second day, i.e., by defining $S(t) = 1$ for odd t and $S(t) = S'(t/2) + 1$ for even t. ◀

To prove Theorem 1, we apply this reduction repeatedly, until we obtain an instance not starting with period 2 – that is, either an instance containing period 1, which is trivially schedulable, or one consisting of periods > 2, whose schedulability is hopefully guaranteed by Lemma 3. The problem is that this last instance may have density $< \alpha^*$, as the reduction step may cause the density to drop from, say, $\mathrm{D}(2, 3, 5, 5, 21) = 1.28\ldots \geq \alpha^*$ to $\mathrm{D}(2, 3, 3, 11) = 1.25\ldots < \alpha^*$. We will argue that this can happen only when we are on the easier track ending up with period 1.

Proof of Theorem 1. The *type* of an instance $A = (a_i)_{i\in[k]}$ is defined as the largest (nonnegative) integer $p \leq k$ such that $a_i \leq 2^i$ for all $i \in [p]$. This instance A is said to be *head-dense* if $a_i \leq 2^{i-1}$ for some $i \in [p]$, and *tail-dense* if

$$\sum_{i=p+1}^{k} \frac{1}{a_i} \geq \sum_{i=p+1}^{\infty} \frac{1}{2^{i-1} + 1}.$$

We prove by induction on p that *every integral instance $A = (a_i)_{i\in[k]}$ of type p that is head-dense or tail-dense is schedulable*. This implies the theorem, because an instance that is neither head-dense nor tail-dense has density $< \alpha^*$.

The claim for $p = 0$ is already proved, since (there is no head-dense instance of type 0, and) tail-dense instances of type 0 were shown schedulable in Lemma 3. So suppose $p > 0$. If $a_1 = 1$, then A is trivially schedulable, so suppose $a_1 = 2$. We claim that the instance $B = (\lceil a_{i+1}/2 \rceil)_{i\in[k-1]}$, which has type $p-1$, is also head-dense or tail-dense. Once we have proved this, B is schedulable by the induction hypothesis, and so is A by Lemma 6.

If A is head-dense, so is B and we are done. Suppose otherwise, so that A is tail-dense. For each $i \geq p$, the ith period $\lceil a_{i+1}/2 \rceil$ in B is roughly half of the corresponding period a_{i+1} in A: specifically, since $a_{i+1} \geq a_{p+1} > 2^{p+1}$, we have $\lceil a_{i+1}/2 \rceil \leq a_{i+1} \cdot (2^p+1)/(2^{p+1}+1)$. This implies the desired tail-density of B by

$$\sum_{i=p}^{k-1} \frac{1}{\lceil a_{i+1}/2 \rceil} \geq \frac{2^{p+1}+1}{2^p+1} \cdot \sum_{i=p+1}^{k} \frac{1}{a_i} \geq \frac{2^{p+1}+1}{2^p+1} \cdot \sum_{i=p+1}^{\infty} \frac{1}{2^{i-1}+1} \geq \sum_{i=p}^{\infty} \frac{1}{2^{i-1}+1},$$

where the last inequality follows from Lemma 7 below. ◀

▶ **Lemma 7.** *For a positive integer p, we have*

$$\frac{2^p}{2^p+1} \cdot \sum_{i=p+1}^{\infty} \frac{1}{2^{i-1}+1} \geq \frac{1}{2^{p-1}+1}.$$

Proof. For $i = 1, 2, \ldots$, let

$$\gamma_i = \frac{2^i}{2^i+1} \cdot \frac{2^{i-1}+1}{2^{i+1}+1}.$$

We have $\gamma_1 \geq \gamma_2 \geq \cdots$, because $\gamma_i \geq \gamma_{i+1}$ follows from

$$\begin{aligned}(\gamma_i - \gamma_{i+1}) \cdot (2^i+1) \cdot (2^{i+1}+1) \cdot (2^{i+2}+1) \\ = 2^i \cdot (2^{i-1}+1) \cdot (2^{i+2}+1) - 2^{i+1} \cdot (2^i+1)^2 = 2^i \cdot (2^{i-1}-1) \geq 0.\end{aligned}$$

In particular, for each $i > p$, we have $\gamma_p \geq \gamma_i$, and hence

$$\sum_{i=p+1}^{\infty} \frac{\gamma_p}{2^{i-1}+1} \geq \sum_{i=p+1}^{\infty} \frac{\gamma_i}{2^{i-1}+1} = \sum_{i=p+1}^{\infty} \left(\frac{1}{2^i+1} - \frac{1}{2^{i+1}+1} \right) = \frac{1}{2^{p+1}+1}.$$

Multiplying by $(2^{p+1}+1)/(2^{p-1}+1)$, we obtain the claimed bound. ◀

3 Computational Techniques (Verifying Lemma 5)

In scheduling an instance $A = (a_i)_{i\in[k]}$, all that matters on a given day is the number of days each agent i has to wait before it can work again, which is a nonnegative integer $< a_i$. Formally, a *state* is a k-tuple $(u_i)_{i\in[k]}$ with $u_i \in \{0, \ldots, a_i - 1\}$, and there is a transition from state $(u_i)_{i\in[k]}$ to state $(v_i)_{i\in[k]}$ with label $j \in [k]$ when $u_j = 0$ and

$$v_i = \begin{cases} a_i - 1 & \text{if } i = j, \\ \max\{0, u_i - 1\} & \text{otherwise.} \end{cases}$$

A schedule is (the sequence of labels of) an infinite walk in this state transition graph (which has $a_1 \cdots a_k$ states). Hence, A is schedulable if and only if this graph has a cycle. This allows us to decide schedulability of a given instance in a finite, albeit exponential, amount of time.

We can thus in principle verify Lemma 5 exhaustively. Yet, doing so in a reasonable amount of time calls for nontrivial techniques, some of which we describe briefly below. A computer program implementing these ideas is publicly available on GitHub at `https://github.com/a7kawamura/pinwheel_solver`.

Representation of States. Since we are interested in instances with periods ≤ 20, and the hard ones are those with many (i.e., 13–19) agents, they typically have multiple agents with the same period. We may exploit this fact (just as for the packing version [3, Section 4.2.2]) by identifying states up to permutation of entries corresponding to duplicate periods. This significantly reduces the size of the state transition graph, and also often the length of the repeating pattern of the schedule to be found. For example, the 21-day cycle of the schedule S in Section 1 for the instance $(3, 5, 5, 5, 7)$ can be regarded as repeating the 7-day cycle of employing agents with periods 3, 5, 5, 3, 5, 7, 5 in order, with the understanding that the three period-5 agents always work in a round-robin fashion.

Reducing the Number of Instances. We also make some effort to get away with fewer instances to check. We need not check an instance that has an agent without whom the instance still satisfies the modified density bound in Lemma 5. We can also ignore an instance containing period 10, because replacing 10 by 11 does not affect the modified density. That leaves 25 242 331 instances, still too many to run the exponential-time cycle-detecting algorithm on.

To further reduce the number of instances, we use the fact that a schedule for an instance $C = (c_i)_{i \in [k]}$ can be easily constructed from one for the instance C' obtained by replacing the last two agents with periods c_{k-1} and c_k by a single agent with period $\min\{c_{k-1}, \lceil c_k/2 \rceil\}$. Since C' has one agent fewer than C, it is often computationally easier to schedule, although of course there is a risk that C' may be unschedulable while C is schedulable. Thus, in order to prove C schedulable, we run the cycle-detecting algorithm in parallel on instances C, C', C'', ... (some of which may not satisfy the bound in Lemma 5), until one of them turns out schedulable. A similar idea was already used for the packing version [3, Section 5.2.2].

This operation of "folding" causes many instances to be proved schedulable via a common short instance, significantly reducing the number of instances on which we actually need to run the cycle-detecting algorithm. We ended up running it only 11 000–12 000 times (this number fluctuates depending on which of the parallel searches for schedules finishes first).

References

1 M. Y. Chan and Francis Chin. Schedulers for larger classes of pinwheel instances. *Algorithmica*, 9:425–462, 1993. doi:10.1007/BF01187034.

2 Peter C. Fishburn and J. C. Lagarias. Pinwheel scheduling: Achievable densities. *Algorithmica*, 34(1):14–38, 2002. doi:10.1007/S00453-002-0938-9.

3 Leszek Gąsieniec, Benjamin Smith, and Sebastian Wild. Towards the 5/6-density conjecture of pinwheel scheduling. In *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 91–103, 2022. doi:10.1137/1.9781611977042.8.

4 R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. 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*, volume 2, pages 693–702, 1989. doi:10.1109/HICSS.1989.48075.

5 Akitoshi Kawamura. Pinwheel Schedulers. Software, version v1.0.0., swhId: `swh:1:dir:df76d28327c884f9cc2d178b6e1c1a4de8675761` (visited on 2026-08-12). URL: `https://github.com/a7kawamura/pinwheel_solver`, `doi:10.4230/artifacts.27639`.

6 Akitoshi Kawamura. Perpetual scheduling under frequency constraints. In Shin-ichi Minato, Takeaki Uno, Norihito Yasuda, Takashi Horiyama, Ken-ichi Kawarabayashi, Shigeru Yamashita, and Hirotaka Ono, editors, *Algorithmic Foundations for Social Advancement: Recent Progress on Theory and Practice*. Springer, 2025.

7 Akitoshi Kawamura. Proof of the density threshold conjecture for pinwheel scheduling. *Proceedings of the National Academy of Sciences*, 123(32):e2530214123, 2026. `doi:10.1073/pnas.2530214123`.

8 Akitoshi Kawamura and Yusuke Kobayashi. A computer-assisted proof of the optimal density bound for pinwheel covering, 2025. `doi:10.48550/arXiv.2510.06533`.

9 Akitoshi Kawamura, Yusuke Kobayashi, and Yosuke Kusano. Pinwheel covering. In Irene Finocchi and Loukas Georgiadis, editors, *Algorithms and Complexity – 14th International Conference, CIAC 2025, Rome, Italy, June 10–12, 2025, Proceedings, Part II*, volume 15680 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2025. `doi:10.1007/978-3-031-92935-9_12`.

10 Akitoshi Kawamura and Makoto Soejima. Simple strategies versus optimal schedules in multi-agent patrolling. *Theoretical Computer Science*, 839:195–206, 2020. `doi:10.1016/j.tcs.2020.07.037`.

11 Yusuke Kobayashi, Bingkai Lin, and Joseph Swernofsky. Hardness and fixed parameter tractability for pinwheel scheduling problems. *Theoretical Computer Science*, 1077:115998, 2026. `doi:10.1016/j.tcs.2026.115998`.

12 Shun-Shii Lin and Kwei-Jay Lin. A pinwheel scheduler for three distinct numbers with a tight schedulability bound. *Algorithmica*, 19:411–426, 1997. `doi:10.1007/PL00009181`.

13 Ahan Mishra. An optimal density bound for discretized point patrolling. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2846–2875, 2026. `doi:10.1137/1.9781611978971.105`.