Polyamorous Scheduling

Leszek Gąsieniec
University of Liverpool, UK

Benjamin Smith
University of Liverpool, UK

Sebastian Wild
University of Liverpool, UK

Abstract

Finding schedules for pairwise meetings between the members of a complex social group without creating interpersonal conflict is challenging, especially when different relationships have different needs. We formally define and study the underlying optimisation problem: Polyamorous Scheduling.

In Polyamorous Scheduling, we are given an edge-weighted graph and try to find a periodic schedule of matchings in this graph such that the maximal weighted waiting time between consecutive occurrences of the same edge is minimised. We show that the problem is NP-hard and that there is no efficient approximation algorithm with a better ratio than 4/3 unless P = NP. On the positive side, we obtain an $O(\log n)$-approximation algorithm; indeed, an $O(\log \Delta)$-approximation for Δ the maximum degree, i.e., the largest number of relationships of any individual. We also define a generalisation of density from the Pinwheel Scheduling Problem, "poly density", and ask whether there exists a poly-density threshold similar to the 5/6-density threshold for Pinwheel Scheduling [Kawamura, STOC 2024]. Polyamorous Scheduling is a natural generalisation of Pinwheel Scheduling with respect to its optimisation variant, Bamboo Garden Trimming.

Our work contributes the first nontrivial hardness-of-approximation reduction for any periodic scheduling problem, and opens up numerous avenues for further study of Polyamorous Scheduling.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Scheduling algorithms

Keywords and phrases Periodic scheduling, Pinwheel scheduling, Edge-coloring, Chromatic index, Approximation algorithms, Hardness of approximation

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.15

Related Version *Full Version*: `https://arxiv.org/abs/2403.00465`

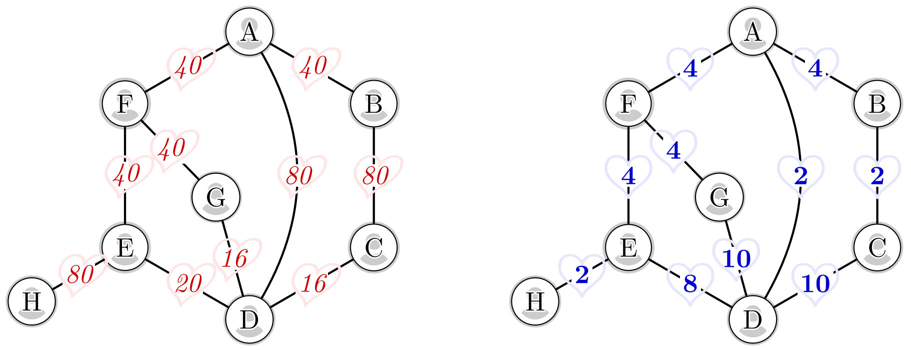
Funding *Sebastian Wild*: Engineering and Physical Sciences Research Council grant EP/X039447/1.

Acknowledgements We are grateful to Viktor Zamaraev for setting us on the right track with the chromatic-index problem, and for several fruitful initial discussions. We also wish to thank Casper Moldrup Rysgaard, Justin Dallant, and Oliver Kim for their helpful contributions; especially Casper, who acted as our rubber duck for a brutally unpolished version of the original hardness-of-approximation reduction.

1 Introduction

We study a natural periodic scheduling problem faced by groups of regularity-loving polyamorous people: Consider a set of persons and a set of pairwise relationships between them, each with a value representing its neediness, importance, or emotional weight. Find a periodic schedule of pairwise meetings between couples that minimizes the maximal weighted waiting time between such meetings, given that each person can meet with at most one of their partners on any particular day.

Before formally defining the Polyamorous Scheduling Problem (Poly Scheduling for short), we illustrate some features of the problem on an example. Figure 1 shows an instance using the natural graph-based representation: We have vertices for people and weighted



Day	A–B	A–D	A–F	B–C	C–D	D–E	D–G	E–F	E–H	F–G
0		♥		♥					♥	♥
1	♥				♥			♥		
2		♥		♥					♥	
3			♥				♥			
4		♥		♥					♥	♥
5	♥							♥		
6		♥		♥					♥	
7			♥			♥				

Figure 1 An example Optimisation Polyamorous Scheduling instance with 8 persons: Adam, Brady, Charlie, Daisy, Eli, Frankie, Grace, and Holly. **Top left:** Graph representation with *edge labels* showing the weight (desire growth rates) of each pairwise relationship. **Bottom:** An optimal schedule for the instance. On each day, a set of meetings is scheduled as indicated by ♥s. The schedule has a period of 8 days: after day 7, we start from day 0 again. **Top right:** A decision version of the instance obtained for heat 160. The **edge labels** here are the frequencies with which edges have to be scheduled stay below heat 160.

(undirected) edges for relationships. It is easy to check that the schedule given at the bottom of Figure 1 never schedules more than one daily meeting for any of the 8 persons in the group; in the graph representation, the set of meetings for each day must form a matching. Each day the mutual desire for a meeting experienced by each couple grows by the weight or *desire growth rate* of that relationship[1] – that is, until a meeting occurs and their desire is reset to zero. We will refer to the highest desire ever felt by any pair when following a given schedule as the *heat* of the schedule. The heat of the schedule in Figure 1 is 160: as the reader can verify, no pair ever feels a desire greater than 160 before meeting and resetting their desire to zero. Desire 160 is also attained; e.g., Adam and Daisy are scheduled to see each other every other day, but over the period of 2 days between subsequent meetings, their desire grows to $2 \cdot 80 = 160$.

For the instance in Figure 1, it is easy to show that no schedule with heat < 160 exists. For that, we first convert from desire growth rates to required *frequencies:* Under a heat-160 schedule, a pair with desire growth rate g must meet at least every $\lfloor 160/g \rfloor$ days. The top-right part of Figure 1 shows the result. It is easy to check that the given schedule indeed achieves these frequencies. However, any further reduction of the desired heat to

[1] "Remember, absence makes the heart grow fonder" [10]. (`https://getyarn.io/yarn-clip/ae628721-c1d1-49d1-bd7c-78cbffceabf0`)

$160-\varepsilon$ would leave, e.g., Adam hopelessly overcommitted: the relation with Daisy would get frequency $\lfloor(160-\varepsilon)/80\rfloor = 1$, forcing them to meet *every* day; but then Brady and Frankie, each with frequency $\lfloor(160-\varepsilon)/40\rfloor \le 3$ cannot be scheduled at all.

While local arguments suffice for our small example, in general, Poly Scheduling is NP-hard (as shown below). We therefore focus this paper on approximation algorithms and inapproximability results.

1.1 Formal Problem Statement

We begin by defining a decision version of Poly Scheduling. In the *Decision Polyamorous Scheduling Problem*, we are given a set of people and pairwise relationships with "attendance frequencies" $f_{i,j}$, and we are trying to find a daily schedule of two-person meetings such that each couple $\{i,j\}$ meets at least every $f_{i,j}$ days. The only constraint on the number of meetings that can occur on any given day is that each person can only participate in at most one of them. A Decision Polyamorous Scheduling instance can naturally be modelled as a graph of people with the edges representing their relationships. Because each person can participate in at most one meeting per day, the edges scheduled on any given day must form a matching in this graph.

▶ **Definition 1** (Decision Polyamorous Scheduling (DPS)). *A DPS instance $\mathcal{P}_d = (P, R, f)$ (a "decision polycule") consists of an undirected graph (P, R) where the vertices $P = \{p_1, \ldots, p_n\}$ are n* persons *and the edges R are pairwise* relationships*, with integer* frequencies *$f : R \to \mathbb{N}$ for each relationship.*

The goal is to find an infinite schedule $S : \mathbb{N}_0 \to 2^R$, such that

(1) *(no conflicts) for all days $t \in \mathbb{N}_0$, $S(t)$ is a matching in $\mathcal{P}_d$, and*

(2) *(frequencies) for all $e \in R$ and $t \in \mathbb{N}_0$, we have $e \in S(t) \cup S(t+1) \cup \cdots \cup S(t+f(e)-1)$;*

or to report that no such schedule exists. In the latter case, $\mathcal{P}_d$ is called infeasible*.*

We write $f_{i,j}$ and f_e as shorthands for $f(\{p_i, p_j\})$ resp. $f(e)$. An infinite schedule exists if and only if a *periodic* schedule exists, i.e., a schedule where there is a period $T \in \mathbb{N}$ such that for all t, we have $S(t) = S(t+T)$: any feasible schedule corresponds to an infinite walk in the finite configuration graph of the problem (see Section 3), implying the existence of a finite cycle. A periodic schedule can be finitely described by listing $S(0), S(1), \ldots, S(T-1)$.

By relaxing the hard maximum frequencies of meetings between couples to "desire growth rates", we obtain the Optimisation Polyamorous Scheduling (OPS) Problem. Our objective is to find a schedule that minimizes the *"heat"*, i.e., the *worst pain of separation* ever felt in the polycule by any couple.

▶ **Definition 2** (Optimisation Polyamorous Scheduling). *An OPS instance (or "optimisation polycule") $\mathcal{P}_o = (P, R, g)$ consists of an undirected graph (P, R) along with a* desire growth rate *$g : R \to \mathbb{R}_{>0}$ for each relationship in R. An infinite schedule $S : \mathbb{N}_0 \to 2^R$ is valid if, for all days, $t \in \mathbb{N}_0$, $S(t)$ is a matching in $\mathcal{P}_o$.*

The goal is to find a valid schedule that minimizes the heat *$h = h(S)$ of the schedule where $h(S) = \max_{e \in R} h_e(S)$ and*

$$h_e(S) = \sup_{d \in \mathbb{N}} \begin{cases} (d+1) \cdot g(e) & \exists t \in \mathbb{N}_0 : e \notin S(t) \cup S(t+1) \cup \cdots \cup S(t+d-1); \\ g(e) & \textit{otherwise.} \end{cases}$$

As for DPS, S can be assumed to be periodic without loss of generality, meaning that S is finitely representable.

1.2 Related Work

Polyamorous Scheduling itself has not been studied to our knowledge. Other variants of periodic scheduling have attracted considerable interest recently [24, 18, 1], including FUN [4].

The simplest periodic scheduling problem is arguably *Pinwheel Scheduling.* In Pinwheel Scheduling [19] we are given k positive integer *frequencies* $f_1 \le f_2 \le \cdots \le f_k$, and the goal is to find a Pinwheel schedule, i.e., an infinite schedule of tasks $1, \ldots, k$ such that any contiguous time window of length f_i contains at least one occurrence of i, for $i = 1, \ldots, k$, (or to report the non-existence of such a schedule).

Pinwheel Scheduling is NP-hard [22], but unknown to be in NP [24], (see [14] for more discussion). Poly Scheduling inherits these properties.

The *density* of a Pinwheel Scheduling instance is given by $d = \sum_{i=1}^{k} 1/f_i$. It is easy to see that $d \le 1$ is a necessary condition for A to be schedulable, but this is not sufficient, as the infeasible instance $(2, 3, M)$ with $d = \frac{5}{6} + 1/M$, for any $M \in \mathbb{N}$ shows. However, there is a threshold d^* so that $d \le d^*$ implies schedulability: Whenever $d \le \frac{1}{2}$, we can replace each frequency f_i by $2^{\lceil \lg(f_i) \rceil}$ without increasing d above 1; then a periodic Pinwheel schedule always exists using the largest frequency as period length. A long sequence of works [19, 6, 20, 26, 7, 13, 11, 14] successively improved bounds on d^*, culminating very recently in Kawamura's proof [24] that it is indeed a sharp threshold, $d^* = \frac{5}{6}$, confirming the corresponding conjecture of Chan and Chin from 1993 [20]. Generalizations of Pinwheel Scheduling have also been studied, e.g., with jobs of different lengths [17, 12].

Pinwheel Scheduling is a special case of DPS, where the underlying graph (P, R) is a *star*, i.e., a centre connected to k pendant vertices with edges of frequencies $f_1, \ldots, f_k$. Note that it is not generally possible to obtain a polyamorous schedule by combining the local schedule of each person[2]; see for example a triangle with edge frequencies 2: In the DPS instance $(\{A, B, C\}, \{A–B, B–C, A–C\}, f)$ with $f(e) = 2$ for all edges, the local problem for each person is feasible by alternating between their two partners, but the global DPS instance has no solution. This example also shows that the simple strategy of replacing f_i by $2^{\lceil \lg(f_i) \rceil}$ is not sufficient to guarantee the existence of a schedule for Poly Scheduling. Indeed, it is unclear whether any such constant-factor scaling of frequencies exists which applies to all Poly Scheduling instances.

There are two natural optimisation variants of Pinwheel Scheduling. In *Windows Scheduling* [3] tasks with frequencies are given and the goal is to find a perpetual scheduling that minimizes the *number* of tasks that need to be done *simultaneously* while respecting all frequencies (i.e., the number of channels or servers needed to schedule all tasks). Efficient constant-factor approximation algorithms are known that use the connection to Bin Packing [2] (where we bin tasks by used channels), even when the sets of tasks to schedule changes over time [8].

The *Bamboo Garden Trimming (BGT) Problem* [16, 15] retains the restriction of one task per day, but converts the frequencies into *growth rates* $g_1 \le \cdots \le g_k$ (of k bamboo plants $1, \ldots, k$) and asks to find a perpetual schedule that minimizes the *height* ever reached by any plant. BGT also allows efficient constant-factor approximations whose approximation factor has seen a lively race of successively improvements over last few years: from 2 [16] over $\frac{12}{7} \approx 1.71$ [28], 1.6 [15], and 1.4 [18], down to the current record, $\frac{4}{3} \approx 1.33$, again by Kawamura [24]. As for the Windows Scheduling problem, no hardness of approximation results are known. It remains open whether it is possible to obtain a PTAS for the Bamboo Garden Trimming Problem [15] or the Windows Scheduling Problem. We show that the same is not true for Poly Scheduling (see Theorem 3 below).

[2] The current state-of-the-art approach in practice, usually via Google Calendar.

As for Pinwheel Scheduling and DPS, Bamboo Garden Trimming is the special case of OPS on star graphs. Although BGT can be approximated well, since it is in general not possible to combine local schedules into a global schedule for a polycule (as noted above), it is not clear whether Poly Scheduling allows an efficient constant-factor approximation.

All mentioned problems above have simple *fractional* counterparts that are much easier to solve and hence provide necessary conditions. Indeed, this is the motivation for density in Pinwheel Scheduling: if we allow a schedule to spend arbitrary fractions of the day on different tasks, we obtain a schedule if and only if the density is at most 1. (Spending a $1/f_i$ fraction on task i each day is best possible). For Windows Scheduling, any valid schedule must partition the tasks into bins (channels/servers), so that each bin admits a Pinwheel schedule. Relaxing the latter constraint to "density at most 1" yields a standard bin packing problem, to which we can apply existing techniques; (packing bins only up to density 5/6 guarantees a Pinwheel schedule, at the expense of a 6/5 factor increase in channels). For Bamboo Garden Trimming, the optimal fractional schedule spends a G/g_i fraction of each day with task i, where G is the sum of all growth rates, thus achieving height exactly G. For Poly Scheduling, we can similarly define a fractional problem, but its structure is much richer (see Section 6).

There are further periodic scheduling problems with less direct connections to Poly Scheduling that received attention in the literature. Patrolling problems typically involve periodic schedules: for example, [1] finds schedules for a fleet of k identical robots to patrol (unweighted) points in a metric space, whereas the "Continuous BGT Problem" [15] sends a single robot to points with different frequencies requirements; [25] tasks k robots with patrolling a line or a circle. The underlying geometry in these problems requires different techniques from our work. The *Point Patrolling Problem* studied in [25] can be seen as a "covering version" of Pinwheel Scheduling: each day, we have to assign one of n workers to a single, daily recurring task, where worker i requires a break of a_i days before they can be made to work again. Yet another twist on a patrolling problem is the *Replenishment Problems with Fixed Turnover Times* given in [5], where vertices in a graph have to be visited with given frequencies, but instead of restricting the number of vertices that can be visited per day, the *length of a tour* to visit them (starting at a depot node) shall be minimized.

In the *Fair Hitting Sequence* Problem [9], we are given a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$, each consisting of a subset of the set of elements $\mathcal{V} = \{v_1, \ldots, v_n\}$. Each set S_j has an urgency factor g_j, which is comparable to the growth rates in BGT instances with one key difference: A set S_j is hit whenever any $v_i \in S_j$ is scheduled. The goal is again similar to BGT; to find a perpetual schedule of elements $v_i \in \mathcal{V}$ that minimizes the time between visits to each set S_j, weighted by g_j. There is also a decision variant, similar to Pinwheel Scheduling in that growth rates are replaced by frequencies. We use a similar layering technique in our approximation algorithm (Section 5) as the $O(\log^2 n)$-approximation from [9], but we obtain a better approximation ratio for Poly Scheduling. Their $O(\log n)$-approximation based on randomized rounding does not extend to Poly Scheduling since the used linear program has exponentially many variables for Poly Scheduling (Section 6).

1.3 Our Results

Despite the recent flurry of results on periodic scheduling, Polyamorous Scheduling seems not to have been studied before. Apart from its immediate practical applications, some quirks make Polyamorous Scheduling an interesting combinatorial optimization problem in its own right. The first version of this manuscript used a direct reduction from 3SAT to introduce the following hardness-of-approximation result, which rules out the existence of a PTAS (polynomial-time approximation scheme) for Optimisation Polyamorous Scheduling.

▶ **Theorem 3** (SAT Hardness of approximation). *Unless P = NP, there is no polynomial-time $(1+\delta)$-approximation algorithm for the Optimisation Poly Scheduling problem for any $\delta < \frac{1}{12}$.*

We retain this original proof in the appendix of the extended online version[3], both for the record and because we expect future works to expand on the methods it develops. We have, however, since found a substantially simpler and stronger hardness-of-approximation result, Theorem 4, by containing the 3-Regular Chromatic Index Problem as a special case.

▶ **Theorem 4** (Hardness of approximation). *Unless P = NP, there is no polynomial-time $(1+\delta)$-approximation algorithm for the Optimisation Poly Scheduling problem for any $\delta < \frac{1}{3}$.*

Though the current form of Theorem 3 follows from Theorem 4, the direct 3SAT reduction is significantly more versatile and we hope to improve the lower bound on the approximation ratio in future work. The core idea of the reduction in Theorem 3 is to force any valid schedule to have a periodic structure with a 3-day period, where edges scheduled on days t with $t \equiv 0 \pmod 3$ represent the value *True* and edges scheduled on days with $t \equiv 1 \pmod 3$ represent *False*; the remaining slots, $t \equiv 2 \pmod 3$, are required to enforce correct propagation along logic gadgets. The appendix, available online, includes a detailed construction of all gadgets, the proof of Theorem 3, and a worked example – the DPS instance corresponding to an example 3-CNF formula.

Theorems 3 and 4 of course imply the NP-hardness of Polyamorous Scheduling; overall, we have 3 independent reductions establishing this. Section 3 surveys these and shows that the best-known upper bound for the complexity of Polyamorous Scheduling is PSPACE. We could thus call Polyamorous Scheduling *very* NP-hard; yet, efficient approximation algorithms are possible. Finding an edge colouring and using a simple round-robin schedule of its colours yields a good approximation if *both* the maximum degree and the ratio between the smallest and the largest desire growth rates are small (Theorem 5).

▶ **Theorem 5** (Colouring approximation). *For an Optimisation Poly Scheduling instance $\mathcal{P}_o = (P, R, g)$ set $g_{\min} = \min_{e\in R} g(e)$, $g_{\max} = \max_{e\in R} g(e)$, and let Δ be the maximum degree in (P, R) and h^* be the heat of an optimal schedule. There is an algorithm that computes in polynomial time a schedule S of heat h with $\frac{h}{h^*} \le \min\left\{\frac{\Delta+1}{\Delta} \cdot \frac{g_{\max}}{g_{\min}},\ \Delta + 1\right\}$.*

A fully general approximation seems only possible with much weaker ratios; we provide an $O(\log \Delta)$-approximation by applying Theorem 5 to groups with similar weight and interleaving the resulting schedules.

▶ **Theorem 6** (Layering approximation). *For an Optimisation Poly Scheduling instance $\mathcal{P}_o = (P, R, g)$, let Δ be the maximum degree in (P, R) and h^* be the heat of an optimal schedule. There is an algorithm that computes in polynomial time a schedule S of heat h with $\frac{h}{h^*} \le 3\lceil \lg(\Delta + 1)\rceil = O(\log n)$, where $n = |P|$.*

Finally, we generalize the notion of density to Polyamorous Scheduling. As discussed above, density has proven instrumental in understanding the structure of Pinwheel Scheduling and in devising better approximation algorithms, by providing a simple, instance-specific lower bound. For Polyamorous Scheduling, the fractional problem is much richer, and indeed remains nontrivial to solve. We devise a generalization of density[4] for Poly Scheduling from the dual of the Linear Program (LP) corresponding to a fractional variant of Polyamorous Scheduling, which gives the following instance-specific lower bound.

[3] extended online version at `https://arxiv.org/abs/2403.00465`

[4] Note that poly density describes how tightly the polycule packs meetings together, not the density of its members.

▶ **Theorem 7** (Fractional lower bound). *Let $\mathcal{P}_o = (P, R, g)$ be an OPS instance with optimal heat h^*. For any set of values $z_e \in [0,1]$, for $e \in R$, with $\sum_{e\in R} z_e = 1$, we have*

$$h^* \;\geq\; \bar{h}(z) \;=\; \frac{1}{\max\limits_{M\in\mathcal{M}} \sum\limits_{e\in M} \dfrac{z_e}{g(e)}}$$

with the maximum ranging over the set of all inclusion-maximal matchings ($\mathcal{M}$) in (P,R). The largest value $\bar{h}^*$ of $\bar{h}(z)$ over all feasible z, is the poly density *of $\mathcal{P}_o$.*

The bound implies (and formally establishes) simple ad-hoc bounds such as the following, which corresponds to the lower bound of G on the height in Bamboo Garden Trimming (setting $z_e = g(e)/G$).

▶ **Corollary 8** (Total growth bound). *Given an OPS instance $\mathcal{P}_o = (P, R, g)$ with optimal heat h^*, let $G = \sum\limits_{e\in R} g(e)$ and m be the size of a maximum matching in (P,R); then $h^* \geq G/m$.*

More importantly though, Theorem 7 allows us to define a *poly density* similarly to the Pinwheel Scheduling Problem, and allows us to formulate the most interesting open problem about Poly Scheduling. For a DPS instance $\mathcal{P}_d = (P, R, f)$, define the poly density of $\mathcal{P}_d$, $\bar{h}^*(\mathcal{P}_d)$, as the poly density of the OPS instance $\mathcal{P}_o = (P, R, 1/f)$ (see also Lemma 10).

▶ **Open Problem 9** (Poly Density Threshold). *Is there a constant c such that every Decision Poly Scheduling instance $\mathcal{P}_d = (P, R, f)$ with poly density $\bar{h}^*(\mathcal{P}_d) \leq c$ admits a valid schedule?*

2 Preliminaries

In this section, we introduce some general notation and collect a few simple facts about Poly Scheduling used later.

We write $[n..m]$ for $\{n, n+1, \ldots, m\}$ and $[n]$ for $[1..n]$. For a set A, we denote its powerset by 2^A. All graphs in this paper are simple and undirected. We denote by $\mathcal{M} = \mathcal{M}(V,E)$ the set of *inclusion-maximal matchings* in graph (V,E), where matching has the usual meaning of an edge set with no two edges incident to the same vertex. By $\Delta = \Delta(V,E)$, we denote the *maximum degree* in (V,E). A *pendant vertex* is a vertex with degree 1. The *chromatic index* $\chi_1 = \chi_1(V,E)$ is the smallest number C of "colours" in a proper edge colouring of (V,E) (i.e., the number of disjoint matchings required to cover E); by Vizing's Theorem [29], we have $\Delta \leq \chi_1 \leq \Delta + 1$ for every graph. Misra and Gries provide a polynomial-time algorithm for edge colouring any graph using at most $\Delta + 1$ colours [27].

Given a schedule $S : \mathbb{N}_0 \to 2^R$ and an edge $e \in R$, we define the *(maximal) recurrence time* $r(e) = r_S(e)$ of e in S as the maximal time between consecutive occurrences of e in S, formally:

$$r_S(e) \;=\; \sup_{d\in\mathbb{N}} \begin{cases} d+1 & \exists t \in \mathbb{N}_0 : e \notin S(t) \cup S(t+1) \cup \cdots \cup S(t+d-1); \\ 0 & \text{otherwise.} \end{cases}$$

Using recurrence time, the heat $h = h(S)$ of a schedule S in an OPS instance (P, R, g) is $h(S) = \max\limits_{e\in R} g(e) \cdot r(e)$. Clearly, for any schedule $S : \mathbb{N}_0 \to 2^R$, we can obtain $S' : \mathbb{N}_0 \to \mathcal{M}$ by adding edges to $S(t)$ until we have a maximal matching $S'(t) \supseteq S(t)$; then $r_{S'}(e) \leq r_S(e)$ for all $e \in R$ and hence S' is a valid schedule for any DPS instance for which S is valid, and if S schedules an OPS instance with heat $h(S)$ then S' does too, with $h(S') \leq h(S)$.

We use Lemma 10 to reduce OPS to DPS, and Lemma 11 to formalize how DPS solves OPS:

▶ **Lemma 10** (OPS to DPS). *For every combination of OPS instance $\mathcal{P}_o = (P, R, g)$ and heat value h, there exists a DPS instance $\mathcal{P}_d = (P, R, f)$ such that*
(1) *any feasible schedule $S : \mathbb{N}_0 \to 2^R$ for $\mathcal{P}_d$ is a schedule for $\mathcal{P}_o$ with heat $\leq h$, and*
(2) *any schedule S' for $\mathcal{P}_o$ with heat $h' > h$ is not feasible for $\mathcal{P}_d$.*

Proof. Consider an OPS polycule $\mathcal{P}_o = (P, R, g)$; we set $\mathcal{P}_d = (P, R, f)$ where $f(e) = \lfloor \frac{h}{g(e)} \rfloor$ for all $e \in R$. Schedules satisfying $\mathcal{P}_d$ when applied to $\mathcal{P}_o$ will allow heat of at most $\max_{e \in R} g(e) \cdot f(e) = \max_{e \in R} g(e) \lfloor \frac{h}{g(e)} \rfloor \leq h$.

Now consider a schedule S' for $\mathcal{P}_o$ with heat $h' > h$. By definition, $h' = \max_{e \in R} r_{S'}(e) \cdot g(e)$, where $r(e) = r_{S'}(e)$ is the recurrence time of e in S'. Assume towards a contradiction that $r(e) \leq f(e)$ for all $e \in R$. This implies that $h' = \max_{e \in R} r(e) \cdot g(e) \leq \max_{e \in R} \lfloor \frac{h}{g(e)} \rfloor \cdot g(e) \leq h$, a contradiction to the assumption. ◀

▶ **Lemma 11** (DPS to OPS). *Let $\mathcal{P}_d = (P, R, f)$ be a DPS instance. Set $F = \max_{e \in R} f(e)$. There is an OPS instance $\mathcal{P}_o = (P, R, g)$ such that the following holds.*
(1) *If $\mathcal{P}_d$ is feasible, then $\mathcal{P}_o$ admits a schedule of height $h \leq 1$.*
(2) *If $\mathcal{P}_d$ is infeasible, then the optimal heat h^* of $\mathcal{P}_o$ satisfies $h^* \geq \frac{F+1}{F}$.*

Proof. Consider a DPS instance $\mathcal{P}_d = (P, R, f)$; we set $\mathcal{P}_o = (P, R, g)$ with $g(e) = 1/f(e)$ for $e \in R$. By definition, any feasible schedule S for $\mathcal{P}_d$ has recurrence time $r_e = r_S(e) \leq f(e)$ for all $e \in R$, so its heat in $\mathcal{P}_o$ is given by $h(S) = \max_{e \in R} r(e) \cdot g(e) = \max_{e \in R} \frac{r(e)}{f(e)} \leq 1$. Conversely, if $\mathcal{P}_d$ is infeasible, then for every $S : \mathbb{N}_0 \to 2^R$ there exists an edge $e' \in R$ where $r(e') > f(e')$, i.e., $r(e') \geq f(e') + 1$. In $\mathcal{P}_o$, the heat $h(S)$ must then be $h(S) = \max_{e \in R} r(e) \cdot g(e) \geq r(e') \cdot g(e') \geq \frac{f(e')+1}{f(e')} \geq \frac{F+1}{F}$. ◀

We will often use the *Normal Form* of OPS instances in proofs; this can be assumed without loss of generality but is not generally useful for algorithms unless h^* is known:

▶ **Lemma 12** (Normal Form OPS). *For every OPS instance $\mathcal{P}_o = (P, R, g)$, there is an equivalent OPS instance $\mathcal{P}'_o = (P, R, g')$ with optimal heat 1 where $g' : R \to \mathcal{U}$ for $\mathcal{U} = \{1/m :\in \mathbb{N}_{\geq 1}\}$, i.e., the set of unit fractions. More precisely, for every schedule $S : \mathbb{N}_0 \to 2^R$ holds: S has optimal heat h^* in $\mathcal{P}_o$ if and only if S has heat 1 in $\mathcal{P}'_o$. That is, any optimal schedule S^* for either problem is also optimal for the other problem.*

Proof. Let (P, R, g) be an arbitrary OPS instance with optimal heat h^*. Setting $\hat{g}(e) = g(e)/h^*$ yields OPS instance $(P, R, \hat{g})$ with optimal heat 1. We now start by setting $g'(e) = \hat{g}(e)$ for all $e \in R$. Consider a particular optimal schedule S^*. Suppose that for some edge $e \in R$, we have $g'(e) \notin \mathcal{U}$. In S, there is a maximal separation $r(e) = q \in \mathbb{N}$ between consecutive occurrences of e with $q \cdot g(e) \leq h^*$. But then, increasing $g(e)$ to h^*/q would not affect the heat of S. We can thus set $g'(e) = 1/q$. By induction, we thus obtain $g' : R \to \mathcal{U}$ without affecting the heat of S. ◀

3 Computational Complexity

One proof of the NP-hardness of the Decision Poly Scheduling (DPS) Problem is that it contains Pinwheel Scheduling as a special case, an NP-hard problem [22]. We show in Section 4 that OPS also contains the Chromatic Index problem as a special case, which gives another proof of the NP-hardness of DPS using the conversion in Lemma 10. Since all good things come in threes, our inapproximability result in the appendix, available online, gives a third independent proof of NP-hardness by reducing 3SAT to DPS.

Upper bounds on the the complexity of DPS are much less clear. Similar to other periodic scheduling problems, characterizing the computational complexity of Poly Scheduling is complicated by the fact that there are feasible instances that require an exponentially large schedule. It is therefore not clear whether Decision Poly Scheduling is in NP since no succinct Yes-certificates are known; this is unknown even for the more restricted Pinwheel Scheduling Problem [24].

The following simple algorithm shows that DPS is at least in PSPACE (see also [14], [25]): Given the polycule $\mathcal{P}_d = (P, R, f)$ with $|P| = n$ and $|R| = m$, construct the *configuration graph* $\mathcal{G}_c = (V, E)$, where V consists of "countdown vectors" listing for each edge e how many days remain before e has to be scheduled again. $v \in V$ has an outgoing edge for every maximal matching M in $\mathcal{M}(P, R)$, and leads to a successor configuration where all $e \in M$ have their urgency reset to $f(e)$ and all $e \notin M$ have their countdown decremented. Feasible schedules for $\mathcal{P}_d$ correspond to infinite walks in the finite $\mathcal{G}_c$, and hence must contain a cycle. Conversely, any cycle forms a valid periodic schedule. Our algorithm for DPS thus checks in time $O(|V| + |E|)$ whether $\mathcal{G}_c$ contains a cycle.

The configuration graph $\mathcal{G}_c$ has single exponential size: $V = \{(u_e)_{e\in R} : u_e \in [0..f(e)]\}$ and E has an edge for every matching in (P, R). So $|E| \le |V| \cdot 2^m$ (since we have at most $2^{|R|}$ matchings) and $|V| \le \prod_{e\in R} f(e)$. To further bound this, we use that all $f(e)$ need to be encoded explicitly in binary in the input. $\prod_{e\in R} f(e) \le \prod_{e\in R} 2^{|f_e|} = 2^{\sum |f_e|} \le 2^N$ for N the size of the encoding of the input.

To obtain a PSPACE algorithm, we use the polylog-space s-t-connectivity algorithm (using Savich's Theorem on the NL-algorithm that guesses the next vertex in the path) on $\mathcal{G}_c$, computing the required part of the graph on-the-fly when queried; this yields overall polynomial space.

4 Unweighted Poly Scheduling & Edge Coloring

Given an OPS instance $\mathcal{P}_o = (P, R, g)$, one can always obtain a feasible schedule from a proper *edge colouring* $c : E \to [C]$ of the graph (P, R): any round-robin schedule of the colours is a valid schedule for $\mathcal{P}_o$, and the number of colours becomes the separation between visits. More formally, we can define a schedule S via $S(t) = \{e \in R : c(e) \equiv t \pmod{C}\}$. An example is shown in Figure 2.

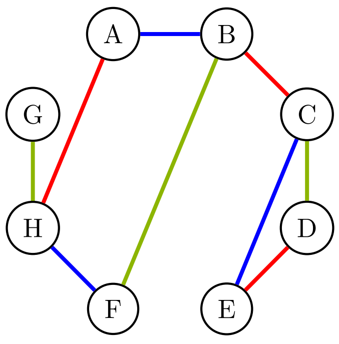


Figure 2 An unweighted polyamorous scheduling instance (that is, an OPS instance where all edges have growth rate 1). Edge colours show one optimal schedule, where every edge is visited exactly every three days: [3, 3, 3], i.e., all red edges are scheduled on days t with $t \equiv 0 \pmod 3$, all blue edges when $t \equiv 1 \pmod 3$ and green edges for $t \equiv 2 \pmod 3$.

Such a schedule can yield an arbitrarily bad solution to general instances of $\mathcal{P}_o$, but it gives optimal solutions for a special case: The non-hierarchical polycule $\mathcal{P}_u$, which is an OPS polycule where all growth rates are $g_{i,j} = 1$ (i.e., an unweighted graph). Recall that any graph with maximal degree $\varDelta$ can be edge-coloured with at most $\varDelta + 1$ colours and clearly needs at least $\varDelta$ colours.

▶ **Proposition 13** (Unweighted OPS = edge coloring)**.** *An unweighted OPS problem admits a schedule with heat h if and only if the corresponding graph is h-edge-colourable.*

Proof. First note that any k-edge-colouring immediately corresponds to a schedule that visits every edge every k days, since we can schedule all edges e with $c(e) = i$ on days $t \equiv i \pmod{C}$. Moreover, any schedule with height h must visit every edge at least once within the first h days (otherwise it would grow to desire $> h \cdot 1$). We can therefore assign h colours according to these first h days of the schedule; some edges might receive more than one colour, but we can use any of these and retain a valid colouring using h colours. ◀

Since it is NP-complete to decide whether a graph has chromatic index $\chi_1 = \varDelta$ (even when the graph is 3-regular [21]) unweighted Poly Scheduling is NP-hard. This provides a second restricted special case of the problem that is NP-hard, which also gives us the inapproximability result stated in Theorem 4:

Proof of Theorem 4 (page 6). Assume that there is a polynomial-time algorithm A that achieves an approximation ratio of $\frac{4}{3} - \varepsilon$ for some $\varepsilon > 0$. Given an input (V, E) to the 3-Regular Chromatic Index Problem (i.e., given a 3-regular graph, decide whether $\chi_1(G) = \varDelta = 3$ or $\chi_1(G) = \varDelta + 1 = 4$), we can apply A to (V, E, g), setting $g(e) = 1$ for all $e \in E$. By Proposition 13, A finds an edge colouring with $c \le (\frac{4}{3} - \varepsilon) \cdot \chi_1(G)$ colours. If $\chi_1(G) = \varDelta = 3$, then $c \le 4 - 3\varepsilon < 4$, so $c = 3$; if $\chi_1(G) = 4$, then $c \ge 4$. Comparing c to $\varDelta$ thus determines $\chi_1(G)$ exactly in polynomial time; in particular, for every 3-regular graph, this decides whether $\chi_1(G) = 3$. Since 3-Regular Chromatic Index is NP-complete [21], it follows that P = NP. ◀

We close this section with the remark that there are weighted DPS instances where any feasible schedule must "multi-colour" some edges, including the polycule shown in Figure 3. For the general problem, we thus cannot restrict our attention to edge colourings (though they may be a valuable tool for future work).

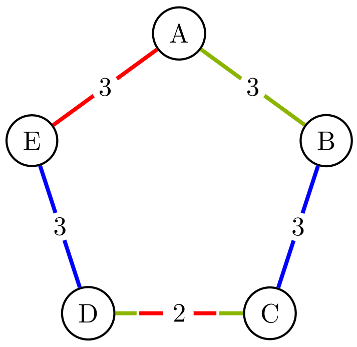


Figure 3 A discrete polyamorous scheduling instance which is solvable only by assigning multiple colours to the CD edge.

5 Approximation Algorithms

In this section, we present two efficient polynomial-time approximation algorithms for Poly Scheduling, thereby proving Theorems 5 and 6. Throughout this section, we assume a fixed instance $\mathcal{P}_o = (P, R, g)$ of Optimisation Polyamorous Scheduling (OPS) is given.

5.1 Lower Bounds

We first collect a few simple lower bounds used in the analysis later; note that Section 6 has further lower bounds.

▶ **Lemma 14** (Simple lower bound). *Given an OPS instance $\mathcal{P}_o = (P, R, g)$, set $g_{\min} = \min_{e\in R} g(e)$, $g_{\max} = \max_{e\in R} g(e)$, and $\varDelta = \max_{p\in P} \deg(p)$. Any periodic schedule for $\mathcal{P}_o$ has heat $h \ge \max\{\varDelta \cdot g_{\min},\, g_{\max}\}$.*

Proof. The chromatic number χ_1 of the unweighted graph (P, R) is $\chi_1 \in \{\varDelta, \varDelta + 1\}$. This means that under any periodic schedule, some edge desires will grow to at least to $\chi_1 \cdot g_{\min} \ge \varDelta \cdot g_{\min}$, since we cannot schedule any two edges incident to a degree-$\varDelta$ node on the same day. Moreover, we cannot prevent the weight-$g_{\max}$ edge from growing to heat $g_{\max}$. ◀

A second observation is that the lower bound for any subset of the problem is also a lower bound for the problem as a whole:

▶ **Lemma 15** (Subset bound). *Given two OPS instances $\mathcal{P}_o = (P, R, g)$ and $\mathcal{P}'_o = (P, R', g')$ with $R' \subseteq R$ and $g(e) = g'(e)$ for all $e \in R'$, i. e., $\mathcal{P}'_o$ results from $\mathcal{P}_o$ by dropping some edges. Assume further that any schedule for $\mathcal{P}'_o$ has heat at least h^*. Then, any schedule for $\mathcal{P}_o$ also has heat at least h^*.*

Proof. Suppose there is a schedule S for $\mathcal{P}_o$ of heat $h' < h$. We obtain a schedule S' for $\mathcal{P}'_o$ by dropping all edges $e \notin R'$. (The resulting schedule may have empty days.) By construction, when using S' to schedule $\mathcal{P}'_o$, all edges in R' will grow to the same heat as in $\mathcal{P}_o$ under S, and hence also to heat $h' < h$. ◀

5.2 Approximation for Almost Equal Growth Rates

We first focus on a special case of OPS instances with "almost equal weights", which is used as base for our main algorithm. Let the edge weights satisfy $g_{\min} \le g(e) \le g_{\max}$ for all $e \in R$. We will show that scheduling a proper edge colouring round-robin gives a $\frac{\varDelta+1}{\varDelta} \cdot \frac{g_{\max}}{g_{\min}}$ approximation algorithm, establishing Theorem 5.

Proof of Theorem 5 (page 6). We compute a proper edge colouring for (P, R) with $\varDelta + 1$ colours using the algorithm from [27] and schedule these $\varDelta + 1$ matchings in a round-robin schedule. No edge desire will grow higher than $(\varDelta + 1) \cdot g_{\max}$ in this schedule. Lemma 14 shows that OPT $\ge \max\{\varDelta \cdot g_{\min}, g_{\max}\}$. The edge-colouring schedule is thus never more than a $\min\{\frac{\varDelta+1}{\varDelta} \cdot \frac{g_{\max}}{g_{\min}},\, \varDelta + 1\}$ factor worse than OPT. ◀

5.3 Layering Algorithm

The colouring-based algorithm from Theorem 5 can be arbitrarily bad if desire growth rates are vastly different and $\varDelta$ is large. For these cases, a more sophisticated algorithm achieves a much better guarantee (Theorem 6). The algorithm consists of 3 steps:

1. breaking the graph into layers (by edge growth rates),

2. solving each layer using Theorem 5, and
3. interleaving the layer schedules into an overall schedule.

Let L be a parameter to be chosen later. We define layers of $\mathcal{P}_o = (P, R, g)$ as follows. For $i = 0, \dots, L-1$, set $\mathcal{P}_i = (P, R_i, g)$ where

$$R_i \;=\; \left\{ e \in R \;:\; \frac{g_{\max}}{2^{i+1}} < g(e) \leq \frac{g_{\max}}{2^i} \right\}.$$

Moreover, $\mathcal{P}_L = (P, R_L, g)$ with $R_L = \left\{ e \in R : g(e) \leq \frac{g_{\max}}{2^L} \right\}$.

Denote by $\varDelta_i$, for $i = 0, \dots, L$, the maximal degree in (P, R_i). Let S_i be the round-robin-$(\varDelta_i + 1)$-colouring schedule from Theorem 5 applied on the OPS instance $\mathcal{P}_i$. If run in isolation on $\mathcal{P}_i$, schedule S_i has heat $h_i \leq (\varDelta_i + 1) g_{\max}/2^i \leq (\varDelta + 1) g_{\max}/2^i$ by the same argument as in Section 5.2. Moreover, for $i < L$, S_i is a $2\frac{\varDelta_i+1}{\varDelta_i}$-approximation (on $\mathcal{P}_i$ in isolation); for $i = L$, we can only guarantee a $(\varDelta_L + 1)$-approximation.

To obtain an overall schedule S for $\mathcal{P}$, we schedule the $L+1$ layers in round-robin fashion, and within each layer's allocated days, we advance through its schedule as before, i.e., $S(t) = S_{(t \bmod (L+1))}\big(\lfloor t/(L+1) \rfloor\big)$. Any advance in layer i is now delayed by a factor $(L+1)$. Hence S achieves heat at most

$$\overline{h} \;=\; \max_{i \in [0..L]} (L+1) \cdot h_i \;\leq\; \max_{i \in [0..L]} (L+1)(\varDelta_i + 1) \cdot \frac{g_{\max}}{2^i}$$

Using Lemma 15 on the layers and Lemma 14, we obtain a lower bound for OPT of

$$\underline{h} \;=\; \max\left\{ \max_{i \in [0..L-1]} \varDelta_i \cdot \frac{g_{\max}}{2^{i+1}},\; g_{\max} \right\}$$

We now distinguish two cases for whether the maximum in $\overline{h}$ is attained for an $i < L$ or for $i = L$. First suppose $\overline{h} = (L+1)(\varDelta_i + 1) g_{\max}/2^i$ for some $i < L$. Since we also have $\underline{h} \geq \varDelta_i \cdot g_{\max}/2^{i+1}$, we obtain an approximation ratio of $2(L+1)\frac{\varDelta_i+1}{\varDelta_i} \leq 3(L+1)$ overall in this case. Here, we assume that $\varDelta_i \geq 2$; otherwise we have only monogamous couples in this layer and scheduling is trivial, giving $h_i = \varDelta_i \cdot g_{\max}/2^i$.

For the other case, namely $\overline{h} = (L+1)(\varDelta_L+1) g_{\max}/2^L > (L+1)\cdot(\varDelta_i+1) g_{\max}/2^i$ for all $i < L$, we do not have lower bounds on the edge growth rates. But we still know $\underline{h} \geq g_{\max}$, so we obtain a $(L+1)(\varDelta_L + 1)/2^L$-approximation overall in this case.

Equating the two approximation ratios suggests to choose L such that $L \approx \lg(\varDelta_L+1) - \lg 3$; with $L = \lceil \lg(\varDelta + 1) - \lg 3 \rceil$ and using $\varDelta_L \leq \varDelta$, we obtain an overall approximation ratio of at most $3(L+1) \leq 3\lceil \lg(\varDelta + 1) \rceil \leq 3 \lceil \lg n \rceil$. This concludes the proof of Theorem 6.

6 Fractional Poly Scheduling

In this section, we generalize the notion of density from Pinwheel Scheduling for the Polyamorous Scheduling Problem. For that, we consider the dual of the linear program corresponding to a fractional variant of Poly Scheduling.

6.1 Linear Programs for Poly Scheduling

In the fractional Poly Scheduling problem, instead of committing to a single matching M in (P, R) each day, we are allowed to devote an arbitrary *fraction* $y_M \in [0, 1]$ of our day to M, but then switch to other matchings without cost or delay for the rest of the day (a simple form of scheduling with preemption). The heat of a fractional schedule is again defined as $\max_{e\in R} r(e)g(e)$, but the recurrence time $r(e)$ now is the maximal time in S before the fraction of days devoted to matchings containing e sum to at least 1. (For a non-preemptive schedule with one matching per day, this coincides with the definition from Section 2.)

Schedules for the fractional problem are substantially easier because there is no need to have different fractions y_M for different days: the schedule obtained by always using the average fraction of time spent on each matching yields the same recurrence times. We can therefore assume without loss of generality that our schedule is given by $S = S(\{y_M\}_{M\in\mathcal{M}})$, with $y_M \in [0,1]$ and $\sum_{M\in\mathcal{M}} y_M = 1$. S schedules the matchings in some arbitrary fixed order, each day devoting the same y_M fraction of the day to M. Then, recurrence times are simply given by $r_S(e) = 1/\sum_{M\in\mathcal{M}:e\in M} y_M$.

With these simplifications, we can state the fractional relaxation of Optimisation Poly Scheduling instance $\mathcal{P}_o = (P, R, g)$ as an optimisation problem as follows:

$$\begin{aligned}
&\min \quad \bar{h} && (1)\\
&\text{s.t.} \quad \sum_{M\in\mathcal{M}} y_M \le 1 && (2)\\
&\frac{1}{\sum_{M\in\mathcal{M}:e\in M} y_M}\cdot g_e \le \bar{h} \qquad \forall e\in R && (3)\\
&y_M \in [0,1] \qquad \forall M\in\mathcal{M} && (4)
\end{aligned}$$

Substituting $\bar{h} = 1/\ell$, this is equivalent to the following linear program (LP):

$$\begin{aligned}
&\max \quad \ell && (5)\\
&\text{s.t.} \quad \sum_{M\in\mathcal{M}} y_M \le 1 && (6)\\
&\frac{1}{g_e}\sum_{M\in\mathcal{M}:e\in M} y_M \ge \ell \qquad \forall e\in R && (7)\\
&y_M \ge 0 \qquad \forall M\in\mathcal{M} && (8)
\end{aligned}$$

The optimal objective value ℓ^* of this LP gives $\bar{h}^* = 1/\ell^*$, the optimal fractional heat.

▶ **Lemma 16** (Fractional lower bound). *Consider an OPS instance $\mathcal{P}_o = (P, R, g)$ with optimal heat h^* and let $\bar{h}^* = 1/\ell^*$ where ℓ^* is the optimal objective value of the fractional-problem LP from Equation (5). Then $\bar{h}^* \le h^*$.*

Proof. We use the same approach as in [9, §3]: For any schedule S, $h(S)$ is at least the heat $h_T(S)$ obtained during the first T days only, which in turn is at least $\max_e g(e)\cdot\bar{r}(e)$ for $\bar{r}(e)$ the *average* recurrence time of edge e during the first T days. A basic calculation shows that for the fractions y_M of time spent on matching M during the first T days there exists a value $1/\ell = h(S)(1-o(T))$, so that we obtain a feasible solution of the LP (5). Hence $1/\ell^* \le 1/\ell = h(S)(1-o(T))$. Since these inequalities hold simultaneously for all T, taking the limit as $T\to\infty$, we obtain $1/\ell^* = \bar{h}^* \le h(S)$. ◀

The immediate usefulness of Lemma 16 is limited since the number of matchings can be exponential in n.

▶ Remark 17 (Randomized-rounding approximation?). One could try to use this LP as the basis of a randomized-rounding approximation algorithm, but since it is not clear how to obtain an efficient algorithm from that, we do not pursue this route here. The simple route taken in [9] cannot achieve an approximation ratio better than $O(\log n)$, so Theorem 6 already provides an equally good deterministic algorithm.

We therefore proceed to the dual LP of Equation (5):

$$\min \quad x \tag{9}$$

$$\text{s.t.} \quad \sum_{e\in R} z_e \geq 1 \tag{10}$$

$$\sum_{e\in M} \frac{z_e}{g_e} \leq x \quad \forall M \in \mathcal{M} \tag{11}$$

$$z_e \geq 0 \quad \forall e \in R \tag{12}$$

While still exponentially large and thus not easy to solve exactly, the dual LP yields the versatile result from Theorem 7.

Proof of Theorem 7 (page 7). Using the given z_e and $x = \max_{M\in\mathcal{M}} \sum_{e\in M} \frac{z_e}{g(e)}$, we fulfil all constraints of Equation (9). The optimal objective value x^* is hence $x^* \leq x$. By the duality of LPs, we have $x^* \geq \ell^*$ for ℓ^* the optimal objective value of Equation (5). Together with Lemma 16, this means $h^* \geq \bar{h}^* = 1/\ell^* \geq 1/x^* \geq 1/x$. ◀

6.2 Poly Density

Theorem 7 gives a more explicit way to compute the *poly density* $\bar{h}^*$ than the primal LP, but it is unclear whether it can be computed exactly in polynomial time. Given the more intricate global structure of Poly Scheduling, $\bar{h}^*$ is necessarily more complicated than the density of Pinwheel Scheduling. A particularly interesting open problem for Poly Scheduling is whether a sufficiently low poly density implies the existence of a valid (integral) schedule.

Specific choices for z_e in Theorem 7 yield several known bounds:

- Setting $z_e = g_e/G$ for $G = \sum_{e\in R} g_e$ yields Corollary 8.
- Fix any subset $R' \subseteq R$. Now set $z_e = g_e/C$ if $e \in R'$ and 0 otherwise, where $C = \sum_{e\in R'} g_e$. The maximum from Theorem 7 then simplifies to $\frac{1}{C}\max_{M\in\mathcal{M}} |M \cap R'|$, so

$$h^* \geq \frac{\sum_{e\in R'} g_e}{\max_{M\in\mathcal{M}} |M \cap R'|}.$$

- An immediate application of that observation with R' being all edges incident at a person $p \in P$ yields the BGT bound:

  ▶ **Corollary 18** (Bamboo lower bound). *Given an OPS instance (P, R, g) and $p \in P$ with $g_1 \geq \cdots \geq g_d$ the desire growth rates for edges incident at p. Set $G_p = g_1 + \cdots + g_d$. Any periodic schedule for (P, R, g) has heat at least G_p.*

▶ Remark 19 (Better general bounds?). For the general case, it seems challenging to obtain other such simple bounds. The bound of G/m is easy to justify without the linear programs by a "preservation-of-mass argument": Assume a schedule S could achieve a heat $h < G/m$. Every day, the overall polycule's desire grows by G, and S can schedule at most m pairs to meet, whose desire is reset to 0 from some value $\leq h$. Every day, S thus removes only a total of $\leq mh < G$ desire units from the polycule, whereas the overall growth is G, a contradiction to the heat remaining bounded.

Note that the bound of G/m is tight for some instances, so we cannot hope for a strictly lower bound. On the other hand, the example from Figure 4 shows demonstrates that it can also be arbitrarily far from h^*.

Figure 4 shows the tadpole family of instances demonstrating the power of the dual-LP approach and Theorem 7. All DPS tadpoles (as shown in the figure) are infeasible since already the triangle A–B–C does not admit a schedule obeying the given frequencies. The corresponding OPS instances (as given by Lemma 11) with $g(e) = 1/f(e)$ thus have $h^* > 1$;

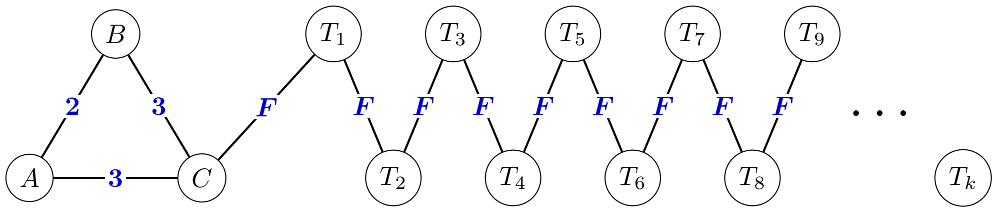


Figure 4 The tadpole family of DPS instances, defined for parameters $k \geq 0$ (tail length) and $F \geq 3$ (tail frequency). The total growth rate is $G = \frac{1}{2} + \frac{2}{3} + k \cdot 1/F = \frac{7}{6} + \frac{k}{F}$ and the size of a maximum matching is $m = 1 + \lfloor (k+1)/2 \rfloor$.

indeed $h^* = 4/3$ if $F \geq 2$. However, the simple lower bounds or local arguments do not detect this: (a) All local Pinwheel Scheduling instances (any person plus their neighbours) are feasible. (b) The mass-preservation bound (Corollary 8) is $G/m < 1$ for $k \geq 1$. Indeed, setting $F = k$ and letting $k \to \infty$, $G/m = O(1/k)$, giving an arbitrarily large gap to h^*. By contrast, consider the LP fractional lower bound. One can show that $\bar{h}^* = \frac{7}{6} > 1$ for any $k \geq 1$ and $F \geq 2$, so Theorem 7 correctly detects the infeasibility in this example.

▶ Remark 20 (Better Pinwheel density via dual LPs?). Since Poly Scheduling is a generalization of Pinwheel Scheduling resp. Bamboo Garden Trimming, we can apply Theorem 7 also to these problems. However, for this special case, the optimal objective value of the dual LPs is *always* $x^* = \ell^* = 1/G$ for G the G the sum of the growth rates, so we only obtain the trivial "biomass" lower bound of G for Bamboo Garden Trimming resp. the density ≤ 1 necessary condition for Pinwheel Scheduling. The more complicated structure of matchings in non-star graphs makes fractional lower bounds in Poly Scheduling much richer and more powerful.

7 Open Problems & Future Directions

This paper opens up several avenues for future work. The most obvious open problem concerns efficient approximation algorithms: We show that finding approximations with a better ratio than 4/3 is NP-hard, and introduce an $O(\log n)$ polynomial-time approximation. Can the gap between these be reduced, or even eliminated?

In the appendix, available online, we conjecture that further analysis of the SAT reduction originally used to prove Theorem 3 may demonstrate better inapproximability results for OPS in the general case. The true lower bound may even be super-constant. However, in light of our Theorem 6, a super-constant hardness of approximation result would have to use Poly Scheduling instances with super-constant degrees. Open Problem 9 will also have clear implications for OPS, as well as being interesting in its own right.

There is also interesting work to be done looking at specific classes of polycules. Bipartite polycules are particularly interesting, both for the likelihood that they will permit better approximations than are possible in the general case and for their applications (e.g., modelling the users and providers of some service).

Polyamorous scheduling has several interesting generalizations including Fungible Polyamorous Scheduling, whose decision version we define as:

▶ **Definition 21** (Fungible Decision Polyamorous Scheduling (FDPS)). *An FDPS instance* $\mathcal{P}_{fd} = (P, R, s, f)$ *(a "(fungible decision)* polycule*") consists of an undirected graph* (P, R) *where the vertices* $P = \{p_1, \ldots, p_n\}$ *are* n classes *of fungible persons and the edges* R *are pairwise relationships between those classes. Classes have integer* sizes $s : P \to \mathbb{N}$ *and relationships have integer* frequencies $f : R \to \mathbb{N}$.

The goal is find an infinite schedule $S : \mathbb{N}_0 \to 2^R$, such that

(1) *(no overflows) for all days $t \in \mathbb{N}_0$, $S(t)$ is a multiset of elements from P such that each node $p \in P$ appears at most $s(p)$ times, and*

(2) *(frequencies) for all $e \in R$ and $t \in \mathbb{N}_0$, we have $e \in S(t) \cup S(t+1) \cup \cdots \cup S(t + f(e) - 1)$;*

or to report that no such schedule exists.

FDPS also has an optimisation version, which again allows each class $p \in P$ to have at most $s(p)$ meetings each day. These problems have clear applications to the scheduling of staff, locations, resources etc. in real-world applications.

Another natural generalisation is Secure Polyamorous scheduling. Suppose that Adam is dating both Brady and Charlie, who are also dating each other. In a DPS or OPS polycule, on any day, Adam must choose to meet with either Brady or Charlie, who each face the same dilemma; but why can't he meet both?[5] The Secure Decision Polyamorous scheduling problem allows this:

▶ **Definition 22** (Secure Decision Polyamorous Scheduling (SDPS)). *An SDPS instance $\mathcal{P}_{sd} = (P, R, f)$ (a "(secure decision)* polycule*") consists of an undirected graph (P, R) where the vertices $P = \{p_1, \ldots, p_n\}$ are n* persons*, and the edges R are pairwise relationships, with integer* frequencies *$f : R \to \mathbb{N}$ for each relationship.*

The goal is find an infinite schedule $S : \mathbb{N}_0 \to 2^R$, such that

(1) *(no third-wheels) for all days $t \in \mathbb{N}_0$, $S(t)$ is a set of meetings between cliques of people in P in which each person appears at most once, and*

(2) *(frequencies) for all $e \in R$ and $t \in \mathbb{N}_0$, we have $e \in S(t) \cup S(t+1) \cup \cdots \cup S(t + f(e) - 1)$;*

or to report that no such schedule exists.

Again, this has a natural optimisation version.

Polyamorous Scheduling also motivates the study of several restricted versions of Pinwheel Scheduling and Bamboo Garden Trimming, including partial scheduling (wherein some portion of the schedule is fixed as part of the problem and the challenge is to find the remainder of the schedule), and fixed holidays (where the fixed part of the schedule consists of periodic gaps).

References

1 Peyman Afshani, Mark de Berg, Kevin Buchin, Jie Gao, Maarten Löffler, Amir Nayyeri, Benjamin Raichel, Rik Sarkar, Haotian Wang, and Hao-Tsung Yang. On cyclic solutions to the min-max latency multi-robot patrolling problem. In *International Symposium on Computational Geometry (SoCG)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPICS.SOCG.2022.2`.

2 Amotz Bar-Noy, Richard E Ladner, and Tami Tamir. Windows scheduling as a restricted version of bin packing. *ACM Transactions on Algorithms*, 3(3):28–es, 2007. `doi:10.1145/1273340.1273344`.

3 Amotz Bar-Noy, Joseph (Seffi) Naor, and Baruch Schieber. Pushing dependent data in clients-providers-servers systems. *Wireless Networks*, 9(5):421–430, 2003. `doi:10.1023/a:1024632031440`.

4 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, *Fun with Algorithms (FUN)*, volume 157 of *Leibniz International Proceedings*

[5] A key part of polyamory [23]!

in Informatics (LIPIcs), pages 5:1–5:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FUN.2021.5.

5 Thomas Bosman, Martijn van Ee, Yang Jiao, Alberto Marchetti-Spaccamela, R. Ravi, and Leen Stougie. Approximation algorithms for replenishment problems with fixed turnover times. *Algorithmica*, 84(9):2597–2621, May 2022. doi:10.1007/s00453-022-00974-4.

6 Mee Yee Chan and Francis Chin. Schedulers for larger classes of pinwheel instances. *Algorithmica*, 9(5):425–462, 1993. doi:10.1007/BF01187034.

7 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.

8 Wun-Tat Chan and Prudence W. H. Wong. On-line windows scheduling of temporary items. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 259–270. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30551-4_24.

9 Serafino Cicerone, Gabriele Di Stefano, Leszek Gasieniec, Tomasz Jurdzinski, Alfredo Navarra, Tomasz Radzik, and Grzegorz Stachowiak. Fair hitting sequence problem: Scheduling activities with varied frequency requirements. In *International Conference on Algorithms and Complexity (CIAC)*, pages 174–186. Springer, 2019. doi:10.1007/978-3-030-17402-6_15.

10 Larry Clemmons, Ken Anderson, and Vance Gerry. Robin hood (movie). *Walt Disney Productions*, 1973.

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

12 Eugene A. Feinberg and Michael T. Curry. Generalized pinwheel problem. *Math. Methods Oper. Res.*, 62(1):99–122, 2005. doi:10.1007/s00186-005-0443-4.

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

14 Leszek Gąsieniec, Benjamin Smith, and Sebastian Wild. Towards the 5/6-density conjecture of pinwheel scheduling. In C. A. Phillips and B. Speckmann, editors, *Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 91–103. SIAM, January 2022. doi:10.1137/1.9781611977042.8.

15 Leszek Gąsieniec, Tomasz Jurdziński, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Perpetual maintenance of machines with different urgency requirements. *Journal of Computer and System Sciences*, 139:103476, February 2024. doi:10.1016/j.jcss.2023.103476.

16 Leszek Gąsieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Min Jie, and Tomasz Radzik. *Bamboo Garden Trimming Problem*, volume 10139 of *Lecture Notes in Computer Science*. Springer, 2017. doi:10.1007/978-3-319-51963-0.

17 C.-C. Han and K.-J. Lin. Scheduling distance-constrained real-time tasks. In *Proceedings Real-Time Systems Symposium*. IEEE Comput. Soc. Press, 1992. doi:10.1109/REAL.1992.242649.

18 Felix Höhne and Rob van Stee. A 10/7-approximation for discrete bamboo garden trimming and continuous trimming on star graphs. In *Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.APPROX/RANDOM.2023.16.

19 Robert Holte, Al Mok, Al Rosier, Igor Tulchinsky, and Igor 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 vol.2, 1989. doi:10.1109/HICSS.1989.48075.

20 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.

21 Ian Holyer. The np-completeness of edge-coloring. *SIAM Journal on computing*, 10(4):718–720, 1981.

22 Tobias Jacobs and Salvatore Longo. A new perspective on the windows scheduling problem. *coRR*, 2014. arXiv:1410.7237.

23 John_Threepwood. Why not both? / Why don't we have both?, August 2011. URL: https://knowyourmeme.com/memes/why-not-both-why-dont-we-have-both.

24 Akitoshi Kawamura. Proof of the density threshold conjecture for pinwheel scheduling. In *Symposium on Theory of Computing (STOC)*, 2024. URL: https://www.kurims.kyoto-u.ac.jp/~kawamura/pinwheel/paper_e.pdf.

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

26 Shun-Shii Lin and Kwei-Jay Lin. A pinwheel scheduler for three distinct numbers with a tight schedulability bound. *Algorithmica*, 19(4):411–426, 1997. doi:10.1007/PL00009181.

27 Jayadev Misra and David Gries. A constructive proof of vizing's theorem. In *Information Processing Letters*. Citeseer, 1992.

28 Martijn van Ee. A 12/7-approximation algorithm for the discrete bamboo garden trimming problem. *Operations Research Letters*, 49(5):645–649, September 2021. doi:10.1016/j.orl.2021.07.001.

29 Vadim G Vizing. The chromatic class of a multigraph. *Cybernetics*, 1(3):32–41, 1965.