Well-Separated Multiagent Path Traversal

Gleb Dilman* David Eppstein† Valentin Polishchuk‡ Christiane Schmidt§

Abstract

We consider moving points along a given path, with a fixed speed, so that no two points ever come closer than 1 (in the space into which the path is embedded, not only along the path) while they follow the path (all points traverse the path from start to finish). Since the motion of any point along the path is fully determined as soon as the point enters the path, our only decisions are the times when to send the points at the start of the path. We give algorithmic results for the problem of scheduling as many points as possible, i.e., maximizing the throughput.

1 Introduction

We study the problem of sending entities/agents along a given path so that the agents stay well separated during the motion. Such problem may arise, e.g., in an amusement park where the given path represents a ride followed by circular cabins or any entities which may deviate from the path (the path may live in 3D and the entities may represent 3D cabin volumes). Maximizing the cabin throughput maximizes the profit of the ride owner, and will also maximize the customers adrenaline, as a dense packing of the cabins implies many near misses along the ride. Similar problem appears when putting large items on a conveyor. Last but not least, the separation may be dictated by privacy or safety concerns, e.g., due to the fear of infection spread between two people or making two entities vulnerable to a single point threat/eavesdropper, which affects a certain radius around it.

For the most part, we focus on 2D; however our solutions work in arbitrary dimensions. We use realRAM model of computation – standard in computational geometry.

1.1 Related Work

To our knowledge, the considered problem was not studied before; however, a large body of work on similar questions exist:

- The *Fréchet distance* between two curves is the length of the shortest leash needed for the person on one curve to walk the dog following the other curve; computing the distance is a classical motion coordination problem (in Section 2 below, we make use of the "free space diagram" from the paper [1] that introduced Fréchet distance). More related to our problem is the recent work on *flipped* Fréchet [10], i.e., walking the dog while *maximizing* the person–dog separation. The main difference from our setup is that in both classical and flipped Fréchet settings; the person and the dog are very powerful—they can move with infinite acceleration; on the contrary, our agents move with same speed—the only decision is when to start moving (and even these starting times are not arbitrary, in some versions of our problem).
- Agents (aircraft or trains, modeled by disks and segments, resp.) following each other ducks-in-a-row along given paths were considered in the CCCG paper [16] (and also in [17]); the trains and aircraft are still more powerful than our agents because they have infinite acceleration (but their speed is bounded). Heuristics were given in [11].
- Wire routing and moving a disk through a domain (the former is hard [13] while the latter can be done in polynomial time [6,7]) is also related to our problem. Finding paths for a "snake" (the Minkowski sum of a segment and a disk, i.e., of a train and an aircraft in terms of [16]) was also studied in [13]: the problem is hard for long snakes, but is FPT-tractable w.r.t. the snake length (i.e., the problem is "length-tractable"), which is reminiscent of our results: we show that our problem is hard, but admits a PTAS if the path length is small.
- Finding separated trajectories is a well studied problem in robotics and computational geometry [2, 4, 8, 12, 14, 18, 19]; our work is different in that we do not find the agents' paths (the path—one for all agents—is given in the input, not sought in the output; our problem is purely a scheduling one).
- From non-geometric literature, remotely related to our paper is the work [5] on sets of words, avoiding a set of forbidden Hamming distance subsequences (the solution to our problem will hinge on defining forbidden intervals between the path-following agents). Another non-geometric, scheduling problem is the "pinwheel problem" in which the goal

*School of Pedagogic Skills Center, gleb.dilman@gmail.com

†Computer Science Department, the University of California, Irvine, eppstein@uci.edu

‡Communications and Transport Systems, ITN, Linköping University, valentin.polishchuk@alumni.stonybrook.edu

§Communications and Transport Systems, ITN, Linköping University, christiane.schmidt@liu.se

is to attend to a set of tasks while ensuring that each task is visited with a certain frequency: it was shown in [9] that there always exists a feasible periodic schedule – a result resembling our proof that for our problem a periodic schedule can be arbitrarily close to the optimum.
- Finding (large) gaps between agents in a periodic motion is the subject of the Lonely runner conjecture [20].

1.2 Problem Formulation and Notation

In the problem input we have a polygonal path P (possibly with self-intersections) which we call the *thread*; let n be the number of edges of P and let $\ell \in \mathbb{R}_+$ be the length of the longest edge. We treat the thread as a directed path; let s be its starting point. A *bead* is a radius-1/2 disk whose center moves with unit speed along P (starting from s). A *schedule* is a sequence $S = (t_1, t_2, \dots)$ of beads inter-release times: that is, according to S, the beads are released at s at times $0, t_1, t_1+t_2, t_1+t_2+t_3, \dots$. For convenience, we start the beads numbering from 0 (bead 0 is released at time 0).

A schedule is *feasible* if the beads never collide with each other while following P, i.e., at any time, the distance between the centers of any two beads is at least 1. The goal is to find schedules with high throughput, i.e., long-term average number of sent beads, or equivalently, to minimize the long-term average of the release times, i.e., $\lim_{m\to\infty} \sum_{j=1}^{m} t_j/m$ (the reciprocal of the throughput). We will restrict attention only to feasible schedules for which the limit exists.

Periodic schedules A schedule is *periodic* if it repeats itself, i.e., if for some p we have $t_{j+p} = t_j \ \forall j$; the minimum p for which this holds is called the *period* of the schedule. If the period $p = 1$ (i.e., if the interval between consecutive beads release is constant), the schedule is called *uniform*.

1.3 Results

Section 2, we present an $O(n^3\ell)$-time algorithm for finding optimal uniform schedules (see the paragraph above for the definiton of uniform and periodic scheudles); in Section 3, we extend the algorithm to periodic schedules of period $p = O(1)$ (the runtimes of the algorithms have p in the exponent). In Section 4, we prove that periodic schedules (with long but bounded period) are as good as arbitrary (i.e., possibly aperiodic) schedules. As a corollary we obtain that the optimal (possibly aperiodic) schedule may be approximated arbitrarily well by a periodic one, implying a PTAS for short paths. Finally, in Section 5, we prove hardness of (even approximating) the problem when the period is large.

In summary, we show that our problem is hard in general, but can be solved in polynomial time for short-period schedules; for arbitrary schedules, the problem admits a PTAS if the thread is short.

An applet to play with sending the beads (also along several paths) is available at `https://www.cs.helsinki.fi/group/compgeom/necklacegame/`: to send a bead, click on the bead at the beginning of the path. Figure 1 shows snapshots of the game.

2 Uniform Schedules

Let $t = t_1 = t_2 = \dots$ be the common value for the beads inter-release times in a uniform schedule; our goal is to minimize t. Consider two beads, with the second one following the first one at distance t along P (Fig. 2), and let e_1, e_2 be the edges of the thread on which the beads are situated at some moment in time (we do not assume that P's edges are numbered – the indicies 1 and 2 in e_1, e_2 are not ordinal numbers; in particular, it is possible that $e_1 = e_2$, or that e_1 is farther than e_2 from s). Let F_{e_1,e_2} be the set of "bad" timings t, i.e., the set of values for t that lead to collision of the beads while they are on e_1, e_2.

Lemma 1 *F_{e_1,e_2} is a single interval (possibly empty).*

Proof. Let x_1, x_2 encode the locations of the beads on e_1, e_2 resp. at some moment of time, i.e., bead i is at distance x_i from the starting point of e_i (recall that P and hence its edges are directed). Let $C \subseteq [0, |e_1|] \times [0, |e_2|]$ be the set of (x_1, x_2) pairs for which the distance between the beads is at most 1; C is the *free space* [1] for the Fréchet distance between the edges. It is well known that C is a connected subset of the (x_1, x_2)-plane (in fact, as was proved in [1], C is convex – the convexity of C follows from the convexity of the distance function). Since $t = d - x_2 + x_1$ where d be the distance (along P) from the startpoint of e_2 to the start point of e_1, the beads motion is described by a (45^o-sloped) line in the (x_1, x_2)-plane. The beads do not intersect iff the line does not intersect C, which happens for a contiguous range of t. □

For an interval $I = [a, b] \subset \mathbb{R}$ and a natural number k, let $I/k = [a/k, b/k]$ denote the "scaled down" copy of I. Having t outside F_{e_1,e_2} ensures that two consecutive beads will not collide on e_1, e_2, i.e., that for any j the bead j does not collide with bead $j+1$. To make sure that bead j does not collide with bead $j+2$, the time interval $2t$ between the beads releases should lie outside F_{e_1,e_2}, or equivalently $t \notin F_{e_1,e_2}/2$. Similarly, for the bead j to avoid bead $j+3$, it should hold that $t \notin F_{e_1,e_2}/3$. In general, to ensure no collisions of beads on e_1, e_2 we should have $t \notin F_{e_1,e_2}/k$ for any natural k. Overall, to avoid beads collisions on any pair of edges

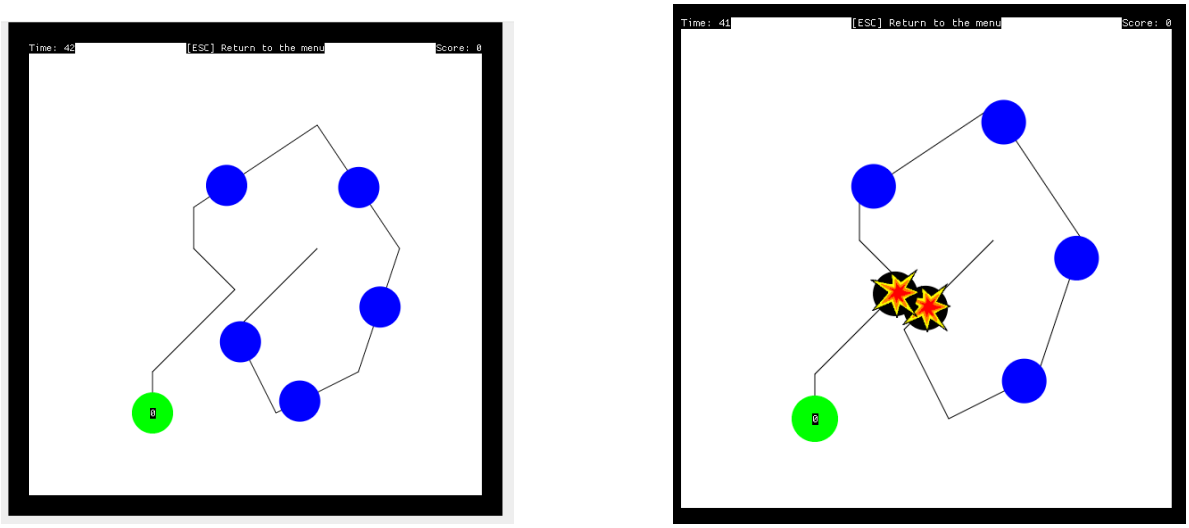

Figure 1: Left: 5 beads (blue) moving along the path. Right: a collision

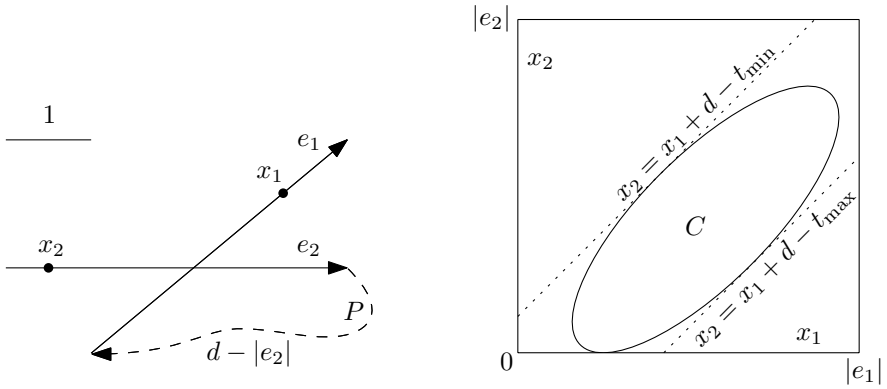

Figure 2: Left: beads at x_1, x_2 on edges e_1, e_2; the part of P between the edges (dashed) has length $d - |e_2|$. Right: $F_{e_1,e_2} = [t_{\min}, t_{\max}]$ is defined by the tangents (dotted) to C (drawn with ipelet [15]).

of the thread, t should be outside all possible forbidden intervals F_σ/k where σ ranges over all pairs of edges of P and $k \in \mathbb{N}$ (Fig. 3).

Theorem 2 *An optimal uniform schedule can be found in $O(n^3\ell)$ time.*

Proof. Let $F_\sigma = [a_\sigma, b_\sigma]$ for a pair σ of edges. It suffices to consider only those k for which $a_\sigma/k \geq 1$, since for a larger k, $[a_\sigma, b_\sigma]/k \subset [0, 1]$ and any $t < 1$ is clearly infeasible. Since the length of the thread is at most $n\ell$, any $t > n\ell$ is feasible, implying $k \leq a_\sigma \leq n\ell$. Thus, the $O(n^3\ell)$ forbidden intervals for t (over all $O(n^2)$ pairs of edges and all $k \leq n\ell$) can be constructed in time $O(n^3\ell)$. The optimal t is the smallest one not covered by the intervals: it is the left endpoint of one of the intervals. □

3 Periodic Schedules

Figure 4 gives a motivation for considering periodic schedules: they can perform arbitrarily better than uniform. We first consider schedules with period 2, defined by the two repeating beads inter-release times t_1, t_2; our goal is to minimize $t_1 + t_2$. The constraint is that no two beads ever collide—on any pair of the thread edges. Therefore, just as with uniform schedules (Section 2), for every pair σ of P's edges we compute the forbidden interval F_σ between two beads on the edges. Let $\mathcal{B} = \bigcup_\sigma F_\sigma$ be the union of all forbidden intervals, treated as a sequence of (maximal) pairwise-disjoint segments on the real line: $\mathcal{B} = [a_1, b_1] \cup [a_2, b_2] \cup \cdots \cup [a_f, b_f] \cup \ldots$ where $a_1 = 0, 1 \leq b_1 < a_2 < b_2 < a_3 \ldots$. The number of the segments is at most the number of edge pairs,

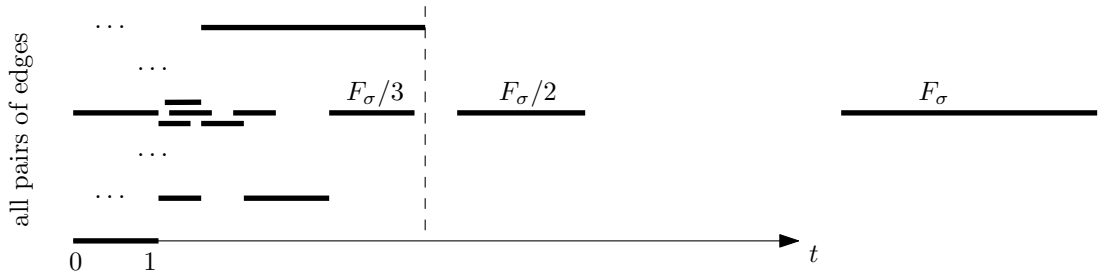

Figure 3: Optimum (dashed) is $\min t : t \notin \bigcup_{\sigma,k} F_\sigma/k$.

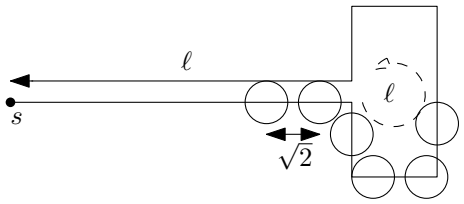

Figure 4: P is shaped as a hammer with thin handle of length $\ell \gg 1$ and the head of perimeter ℓ. A uniform schedule must have $t > \ell$ (so that beads do not collide on the opposite sides of the handle). A periodic schedule can send a length-ℓ train of $\lfloor \ell/\sqrt{2} \rfloor$ beads with inter-release times $\sqrt{2}$ (so the beads do not collide at the 90^o turns of P), wait until the train fully leaves P (time 4ℓ), and repeat. Thus the uniform schedule has throughput $\Theta(1/\ell)$ while the throughput of the periodic schedule is constant (the long-term average of the release times is $> \ell$ for uniform schedules and $\Theta(1)$ for a periodic one).

$O(n^2)$.

In a period-2 schedule, the interval (the distance along the thread) between beads of the same parity is $k(t_1+t_2)$ for an integer $k \geq 1$. The interval between beads of different parity is either $k(t_1 + t_2) + t_1$ (from an even bead to an odd) or $k(t_1 + t_2) + t_2$ (from an odd to an even bead) for $k \geq 0$. Thus for a feasible schedule it is necessary and sufficient that

$$\begin{array}{cccc} (k+1)(t_1+t_2), & k(t_1+t_2)+t_1, \\ k(t_1+t_2)+t_2 & \notin & \mathcal{B} & \forall k = 0, 1, \ldots, n\ell/2 \end{array} \quad (1)$$

(as with uniform schedules, $t_1, t_2 \geq 1$, and it suffices to consider inter-release times that do not exceed the maximum thread length, $n\ell$, i.e., $(k+1)(1+1) \leq n\ell$).

For any one forbidden segment $[a_f, b_f]$ from $\mathcal{B}$ and any fixed k, the inequalities $a_f \leq (k+1)(t_1+t_2) \leq b_f$ define a slab of forbidden pairs in the (t_1, t_2)-plane (Fig. 5). Similarly, the inequalities $a_f \leq k(t_1+t_2)+t_1 \leq b_f$ and $a_f \leq k(t_1+t_2)+t_2 \leq b_f$ each define a slab. Overall, i.e., for segments $[a_f, b_f]$ for all $O(n^2)$ f's and $O(n\ell)$ k's, the requirements (1) define $O(n^3\ell)$ slabs. We build the arrangement of the slabs and find the vertex of the arrangement minimizing $t_1 + t_2$ in $O(n^6\ell^2)$ time

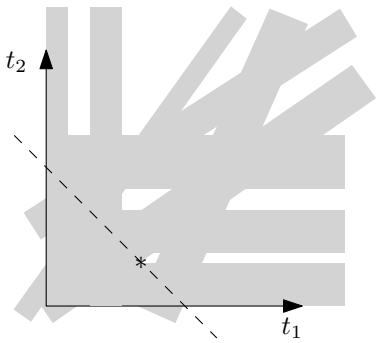

Figure 5: The requirements (1) define slabs of forbidden (t_1, t_2) pairs (gray). The optimal schedule (marked with the asterisk) minimizes $t_1 + t_2$ for points outside the slabs (white).

(by going through all the vertices).

The above algorithm extends to schedules of any length p:

Theorem 3 *An optimal schedule with period p can be found in $O((n^3 p\ell)^p)$ time.*

Proof. Let $b, b' \in \mathbb{Z}_0^+, b < b'$ be two beads, identified with their positions in the schedule (recall that we start numbering the beads from 0). If $b \equiv b' \bmod p$, then in a period-p schedule the interval (the distance along the thread) between the beads is $(k+1)(t_1 + t_2 + \cdots + t_p)$ for an integer $k \geq 0$. More generally, if $b \equiv r \bmod p$ and $b' \equiv r' \bmod p$, the distance is

$$\begin{array}{c} D(r.r', k) = t_{r'+1} + t_{r'+2} + \cdots + t_p + \\ +k(t_1 + \cdots + t_p) + t_1 + t_2 + \cdots + t_r \end{array} \quad (2)$$

Thus for a feasible schedule it is necessary and sufficient that

$$\begin{array}{cc} D(r, r', k) \notin \mathcal{B} \\ \forall k = 0, 1, \ldots, n\ell/p, \quad \forall r, r' = 0, 1, \ldots p-1 \end{array} \quad (3)$$

For any one segment $[a_f, b_f]$ from $\mathcal{B}$ and any fixed r, r', k the inequalities $a_f \leq D(r, r', k) \leq b_f$ define a slab of forbidden schedules in the $(t_1, t_2, \ldots, t_p)$-space—overall, i.e., for segments $[a_f, b_f]$ for all $O(n^2)$ f's,

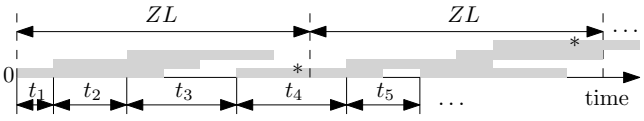

Figure 6: Length-L segments (gray) correspond to beads in the maximum-throughput schedule; we will remove the segments marked with asterisks.

$O(n\ell/p)$ k's and $O(p^2)$ pairs (r, r'), the requirements (1) define $O(n^3 p\ell)$ slabs in the p-dimensional space. We build the arrangement of the slabs and find the vertex of the arrangement minimizing $t_1 + t_2 + \cdots + t_p$. □

4 Arbitrary Schedules

Let τ be the value of optimal throughput over arbitrary, possibly aperiodic schedules. We show that periodic schedules can achieve a throughput arbitrarily close to τ:

Theorem 4 *For any $\varepsilon > 0$ there exists a periodic schedule whose throughput is at least $\tau - \varepsilon$.*

Proof. Let L denote the length of P. Identify each bead with the length-L segment on the real line, spanning the time during which the (center of) the bead traverses P (Fig. 6). Consider the segments in the optimal (possibly aperiodic) schedule S. Choose an integer $Z > 2L\tau^2/\varepsilon$ and draw vertical lines $t = ZL, t = 2ZL, \ldots$ at the regular spacing ZL. We claim that no such line intersects the interior of more than $x = 2L\tau$ segments. Indeed, the segments intersecting any one line cover at most $2L$ of the time. If there existed a line intersecting a set of more than x segments, we could have laid such sets one after another, obtaining a (periodic) schedule with throughput at least $x/2L > \tau$, contradicting the global optimality of S.

Now remove from S the segments whose interior is intersected by one of the drawn vertical lines. By the above, the fraction of the removed segments is less than x/Z, implying that the throughput of the schedule with the remaining segments is greater than $\tau - \tau x/Z > \tau - \varepsilon$. It follows that at least one set of segments between the lines has throughput $> \tau = \varepsilon$; (in fact, such sets should appear infinitely often, but for us it is enough to) take one such set and repeat it – this results in a periodic schedule with throughput $> \tau - \varepsilon$. □

It follows from the proof of Theorem 4 that to get a schedule with throughput $\tau(1 - x/Z) > \tau(1 - \varepsilon/\tau)$ it is enough to find an optimal schedule in a length-ZL interval. Since the number of beads sent during the interval is $O(ZL) = O(L^2\tau^2/\varepsilon)$, it suffices to consider schedules with period $O(L^2\tau^2/\varepsilon)$. Taking $\delta = \varepsilon/\tau$, we conclude that a $(1-\delta)$-approximation to the maximum throughput can be obtained by considering schedules with period $p = O(L^2\tau/\delta) = O(L^3/\delta)$ since $\tau < L$. From Theorem 3,

Theorem 5 *A $(1-\delta)$-approximation to the maximum throughput can be found in $O((nL/\delta)^{O(L^3/\delta)})$ time.*

In particular, for threads with constant length, our problem has a PTAS.

5 Hardness of Approximating Arbitrary Schedules

Our result is based on a known inapproximability results for maximum clique by Arora et al. [3]:

Theorem 6 (Arora et al. [3]) *There is a constant $c > 0$, such that approximating the maximum clique size in an N-vertex graph to within a factor N^c is NP-hard.*

We combine this result for the maximum clique problem with an approximation-preserving reduction from maximum clique to our problem, that is, the problem of determining the optimal schedule of beads. Hence, from a given graph G in which we aim to find a maximum clique we will construct a thread P, such that the optimal release times for P correspond to a maximum clique in G. Thus, the approximation ratio for the optimal schedule of beads for P cannot be better than the approximation ratio for maximum clique in G (given by Theorem 6).

For the construction of P from G, we define a set $\mathcal{I}$ of maximal intervals of the timeline such that, if a bead is released at time 0, it is safe to release another bead within one of these intervals (and inter-release times $\sum_{j=i}^{i+k} t_j \notin \mathcal{I}, i = 1, 2, \ldots; k = 0, 1, \ldots$ are infeasible). We call $\mathcal{I}$ the set of *safe* intervals. Assuming that we can construct $\mathcal{I}$ as desired, we describe the approximation-preserving reduction from maximum clique in Section 5.1, and we detail the construction of $\mathcal{I}$ in Section 5.2.

5.1 Reduction from Maximum Clique

Let G be the graph in which we want to solve the maximum clique problem, let $|V(G)| = N$, and let the vertices be labeled $1, \ldots, N$. We use a greedy algorithm to construct a set $U = \{u_1, u_2, \ldots u_N, u_{N+1}\}$ of integers, such that $u_{j1} \pm u_{j2} \pm u_{j3} \pm u_{j4} \pm u_{j5} \pm u_{j6} \neq 0$ for any 6 indicies $j_1, \ldots, j_6 \in \{1, \ldots, N+1\}$. We start with $u_1 = 2$ (see Section 5.2 on why we do not start with 1) and consider integers of increasing value, adding them to U whenever they do not yield an infeasible linear combination with the numbers previously added to U. For integers $1, \ldots, k$, we need to pick at least $k^{1/5}$ integers for U in order to exclude the other numbers due to infeasible linear combinations. Thus, $u_N \in O(N^5)$.

We construct a thread P with safe intervals around $\{u_i \mid 1 \le i \le N\} \cup \{u_i - u_j \mid i > j \text{ and } ij \in E(G)\}$ and a semi-infinite interval starting at u_{N+1}. That is

$$\mathcal{I} = \bigcup_{i=1}^{N} (u_i - \varepsilon, u_i + \varepsilon) \cup \bigcup_{ij \in E(G); i>j} (u_i - u_j - \varepsilon, u_i - u_j + \varepsilon) \cup [u_{N+1}, \infty), \quad \varepsilon > 0 \tag{4}$$

Any optimal schedule with release times smaller than u_{N+1} can be repeated at intervals of time u_{N+1}. Hence, we obtain a finite problem: Find the largest subset of the finite parts of $\mathcal{I}$ all of whose differences are in $\mathcal{I}$.

Let $K = \{v_{k_1}, v_{k_2}, \ldots, v_{k_{|K|}}\}$ be the set of vertices in a clique with $k_1 < k_2 < ... < k_{|K|}$, then release times $0, t_1, t_1 + t_2, \ldots$ with

$$\sum_{i=1}^{j} t_i = u_{k_j}, j = 1, \ldots, |K| \tag{5}$$

yield a feasible schedule of $|K| + 1$ release times: the inter-release times (for $j_1 < j_2$) are $(t_1 + t_2 + \ldots + t_{j_2}) - (t_1 + t_2 + \ldots + t_{j_1}) = u_{k_{j_2}} - u_{k_{j_1}} \in \mathcal{I}$, because $(v_{k_{j_1}}, v_{k_{j_2}}) \in E(G)$. Moreover, any feasible schedule for P for which all release times are 0 or elements of U must be of this form.

Additionally, for a clique K in G the set $\{0, u_{k_{|K|}}\} \cup_{j=1,\ldots,k_{|K|-1}} \{u_{k_{|K|}} - u_j\}$ is a feasible set of release times. There also exist feasible schedules with release times of the form $u_{k_i} - u_{k_j}$. The release time $u_{k_i} - u_{k_j}$ is compatible with u_{k_i} (because $u_{k_i} - (u_{k_i} - u_{k_j}) = u_{k_j} \in \mathcal{I}$) and with a single other time $u_{k_j} - u_{k_{j^*}}$ (because $(u_{k_i} - u_{k_j}) - (u_{k_j} - u_{k_{j^*}}) = u_{k_i} - u_{k_{j^*}} \in \mathcal{I}$ if $ij^* \in E(G)$), but not with more release times of this form. Hence, the schedules that do not stem from cliques in G have bounded size. Thus, if we aim for cliques larger than that, the release times in an optimal schedule must stem from a clique in G.

Using Section 5.2, we construct a thread P with $n_P = 3 + 7 \cdot N + 7 \cdot |E(G)|$ edges with $|\mathcal{I}| = f(n_P) = N + |E(G)| + 1$ safe intervals. With Theorem 6 we yield:

Theorem 7 *There exists a constant $c > 0$, such that approximating the optimal schedule of beads in a thread with n_P edges to within a factor $n_P^{1/2-c}$ is NP-hard.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that can schedule beads optimally within a factor $\rho \le n_P^{1/2-c}$. Then we can construct an algorithm $\mathcal{A}'$ for the maximum clique problem in G:

1. Use Lemma 8 to construct P from the given graph G (the input for the maximum clique problem).
2. Use algorithm $\mathcal{A}$ to schedule beads on P.
3. Construct a maximum clique for G from the release times for P.

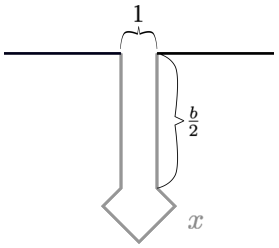

Figure 7: Gadget to exclude the interval $[x-b, x]$. The black lines are the horizontal edges of P adjacent to the gadget. The total length of the (gray) gadget is x.

Then the approximation ratio for $\mathcal{A}'$ for the maximum clique problem is the same as for algorithm $\mathcal{A}$ for scheduling beads (ρ). Hence, we have:

$$\rho \le n_P^{1/2-c} = O((N+M)^{1/2-c}) = (N^{1-2c}) \tag{6}$$

which yields a contradiction to Theorem 6. □

5.2 Construction of the Set of Safe Intervals

We aim to exclude all but the set of safe intervals given in Equation (4). We use a long horizontal path to which we add several gadgets to exclude certain intervals. Between each pair of consecutive gadgets we have a horizontal edge of length u_{N+1} in P:

- To exclude $(0, u_1 - \varepsilon]$ use a path of length $u_1 - \varepsilon$ that runs on itself, i.e., $\frac{u_1-\varepsilon}{2}$ up and $\frac{u_1-\varepsilon}{2}$ down.
- To exclude intervals of the form $[x - b, x]$ we use the gadget shown in Figure 7: The total length of the gadget (shown in gray) is x, we have two vertical edges of length $\frac{b}{2}$ each within a distance of 1, and part of a slanted square with four edges, total length $x - b$, and a distance > 1 between parallel edges. For example, if we aim to exclude the interval $[u_i + \varepsilon, u_{i+1} - \varepsilon]$ from the safe intervals, we choose $x = u_{i+1} - \varepsilon$, $b/2 = u_i - \varepsilon$, which excludes the interval $[x-b, x]$ with $x - b = u_{i+1} - \varepsilon - 2*(u_i - \varepsilon) = u_i + \varepsilon$. Each of these gadgets also excludes the interval $(0, \sqrt{2}]$, thus, we choose $u_1 > 1$.

Each gadget except for the first has one horizontal edge and six edges within the gadget. The gadget excluding the interval $(0, u_1 - \varepsilon]$ has three edges. Hence, for a graph G with N vertices, constructing the set $\mathcal{I}$ of safe intervals from Equation (4), that is, excluding all "unsafe" intervals, we use $n = 3 + 7 \cdot N + 7 \cdot |E(G)|$ edges. This yields:

Lemma 8 *Given a graph G with $|V(G)| = N, |E(G)| = M$, we can construct in polynomial time a thread P with $n_P = 3 + 7 \cdot N + 7 \cdot |E(G)|$ vertices, and $C(G) \le S(P) \le C(G) + 1$, where $C(G)$ is the maximum clique size in G, and $S(P)$ the length of an optimum finite schedule for P.*

Acknowledgements We thank the anonymous reviewers for their helpful comments. This research is partially supported by the Swedish Transport Administration and the Swedish Research Council.

6 Conclusion

We gave pseudopolynomial-time algorithms and hardness results for scheduling uniform motion of well separated agents along a given path; our algorithms extend to the case of agents following multiple (constant number of) paths. An open problem is the existence of a polynomial-time solution.

References

[1] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[2] E. M. Arkin, J. S. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 20–27, 2008.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

[4] A. T. Becker, S. P. Fekete, P. Keldenich, M. Konitzny, L. Lin, and C. Scheffer. Coordinated motion planning: The video (multimedia exposition). In *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2018.

[5] V. D. Blondel, R. Jungers, and V. Protasov. On the complexity of computing the capacity of codes that avoid forbidden difference patterns. *IEEE Transactions on Information Theory*, 52(11):5122–5127, 2006.

[6] D. Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. *ACM Transactions on Algorithms (TALG)*, 11(4):1–46, 2015.

[7] L. P. Chew. Planning the shortest path for a disc in o (n 2log n) time. In *Proceedings of the first annual symposium on Computational geometry*, pages 214–220, 1985.

[8] E. D. Demaine, S. P. Fekete, P. Keldenich, H. Meijer, and C. Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM Journal on Computing*, 48(6):1727–1762, 2019.

[9] E. A. Feinberg and M. T. Curry. Generalized pinwheel problem. *Mathematical Methods of Operations Research*, 62:99–122, 2005.

[10] O. Filtser, M. Goswami, J. S. Mitchell, and V. Polishchuk. On Flipping the Fréchet distance. In *ITCS - Innovations in Theoretical Computer Science*, 2023.

[11] J. Kim, A. Kröller, and J. Mitchell. Scheduling aircraft to reduce controller workload. In *9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'09)(2009)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2009.

[12] Y. Kobayashi and K. Otsuki. Max-flow min-cut theorem and faster algorithms in a circular disk failure model. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1635–1643. IEEE, 2014.

[13] I. Kostitsyna and V. Polishchuk. Simple wriggling is hard unless you are a fat hippo. *Theory of Computing Systems*, 50(1):93–110, 2012. Special issue on FUN'10.

[14] S. Neumayer, A. Efrat, and E. Modiano. Geographic max-flow and min-cut under a circular disk failure model. *Computer Networks*, 77:117–127, 2015.

[15] G. Röte. Free-space diagram ipelet. `https://www.mi.fu-berlin.de/inf/groups/ag-ti/software/ipelets.html`.

[16] C. Scheffer. Scheduling Three Trains is NP-Complete. In *CCCG*, pages 87–93, 2020.

[17] C. Scheffer. Train scheduling: Hardness and algorithms. In *WALCOM: Algorithms and Computation: 14th International Conference, WALCOM 2020, Singapore, Singapore, March 31–April 2, 2020, Proceedings 14*, pages 342–347. Springer, 2020.

[18] A. Sen, S. Murthy, and S. Banerjee. Region-based connectivity-a new paradigm for design of fault-tolerant networks. In *2009 International Conference on High Performance Switching and Routing*, pages 1–7. IEEE, 2009.

[19] A. Sen, B. H. Shen, L. Zhou, and B. Hao. Fault-tolerance in sensor networks: A new evaluation metric. In *INFOCOM 2006: 25th IEEE International Conference on Computer Communications*, page 4146923, 2006.