Two Robots Patrolling on a Line: Integer Version and Approximability

Peter Damaschke(✉)

Department of Computer Science and Engineering,
Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se

Abstract. Suppose that two robots can move at unit speed on a line and must visit certain points called stations infinitely often. Every station allows some maximal waiting time between two visits. The problem is to construct an optimal schedule for the robots. While the one-robot problem is easy to solve in linear time, already for two robots the complexity is open. Chuangpishit, Czyzowicz, Gasieniec, Georgiou, Jurdzinski, and Kranakis (SOFSEM 2018) found a $\sqrt{3}$-approximation algorithm. Here we provide a PTAS, accomplished by rounding and (perhaps more surprisingly) by using the well-quasi ordering of vectors of positive integers. The result is not very practical in the present form, but further investigation of the integer version may make it more usable.

Keywords: Patrolling · Approximation scheme · Well-quasi ordering

1 Introduction

Patrolling problems where mobile robots must visit certain points at least with prescribed frequencies are interesting for monitoring and maintenance. Various cases and aspects have been studied: environments with different topologies, unreliable robots, with equal and different speeds, etc. [3–6]. See also a recent survey in [2]. Pinwheel scheduling [8,10,12] is also a special case of patrolling where all points have equal pairwise distances. More recently, patrolling problems received new attention by observing that different individual frequencies make them difficult, even on the simplest topologies [1,9].

The problem called PUF (patrolling with unbalanced frequencies) in [1] is the following (with somewhat changed notation): n stations are deployed at fixed points s_i $(i = 1, \ldots, n)$ on the real line L. For every station i, a duration $t_i > 0$ is also specified. Two identical robots move on L, at some given maximum speed. We say that station i is *visited* at some moment if at least one robot is at s_i. The problem is to construct a *schedule*, i.e., a pair of trajectories of two robots such that, during an unlimited period of time, every station i gets repeatedly visited, and the time between two consecutive visits never exceeds t_i. (But it does not matter which robots visit the station.) Of course, the same problem

L. Gąsieniec et al. (Eds.): IWOCA 2020, LNCS 12126, pp. 211–223, 2020.
https://doi.org/10.1007/978-3-030-48966-3_16

may be defined for any number of robots and for other topologies. Unless said otherwise, in the present paper PUF refers to the case of two robots on a line.

For any number $c \geq 1$, an instance of PUF is called *c-feasible* if it has a c-feasible schedule, i.e., for every i, the time between two consecutive visits of station i never exceeds ct_i. A 1-feasible instance or solution is simply called *feasible*. One may also view PUF as an optimization problem with the goal to find a schedule with minimum c. A c-approximation algorithm is one that outputs a c-feasible schedule when a feasible schedule exists.

While PUF for one robot on a line is easy to solve in $O(n)$ time, PUF becomes surprisingly difficult already for two robots: Only a $\sqrt{3}$-approximation is achieved in [1]. To our best knowledge, this is the state-of-the-art polynomial-time approximation, and the complexity status of PUF is open (both for deciding existence of feasible solutions and minimizing the times). It is far from obvious how one should divide the visits of certain stations among the two robots.

Overview. In Sect. 2 we introduce the integer version called IntPUF, with $m+1$ stations at points $0, \ldots, m$, and the notion of instance vectors that encode the instances. Since fixed-length vectors of positive integers are well-quasi ordered (WQO), only finitely many minimal feasible vectors exist for any fixed m. Using this fact plus some elementary graph theory, we can solve IntPUF in a time depending on m only, but not on the t_i. (However, the time as a function of m remains open.) In Sect. 3 we switch from m to the parameter $k := \min_i t_i$. The reason is that IntPUF can be approximated with a ratio arbitrarily close to 1, in a time depending on k only: If k/m is small, this is simple, and otherwise the result of Sect. 2 is applied. This insight is used to construct a PTAS in Sect. 4, where k is a discretization parameter. As far as we know, this might be the first example of using WQO in a PTAS. Next, rounding of the s_i and t_i to integers causes yet another approximation error. (The issue is that a station shifted to the next integer point can escape a turning point of a robot's trajectory.) Larger k yield better approximations but also higher time complexity. We remark that several ideas on the way, especially the WQO argument, can be generalized to more robots and to other topologies, but we keep the focus on two robots on a line.

2 The Integer Version of PUF

We introduce a variant of PUF that we name IntPUF. We define it precisely as PUF (see Sect. 1) but with the following additional demands:

- All s_i and t_i are integer (and $t_i > 0$).
- Every robot is, at any time, in one of two possible modes: either it stays at some point with integer coordinate or it moves at unit speed.
- Every robot can change its speed or its moving direction only at integer times and at integer points.

We remark that PUF allows real values, and this might lead to subtle effects for irrational numbers. Thus, algorithms for IntPUF may not completely solve

PUF. However, we have not further considered such questions. Note that, in practice, input numbers are usually rational.

Lemma 1. *Let r and s be real numbers and t an integer, with $0 \leq s - r \leq t$. Then $0 \leq \lfloor s \rfloor - \lfloor r \rfloor \leq t$.*

Proof. Non-negativity is obvious. To see $\lfloor s \rfloor - \lfloor r \rfloor \leq t$, we write the numbers as $r = \lfloor r \rfloor + r'$ and $s = \lfloor s \rfloor + s'$, Observe that $\lfloor s \rfloor - \lfloor r \rfloor = (s - s') - (r - r') = (s - r) + (r' - s')$. If $r' \leq s'$ then $s - r \leq t$ yields the assertion. If $r' > s'$ then $(\lfloor s \rfloor + s') - (\lfloor r \rfloor + r') \leq t$ implies that still $(\lfloor s \rfloor + s') - (\lfloor r \rfloor + s') \leq t$, since t is integer and $r' - s' < 1$. Again the assertion follows. □

Theorem 1. *Every feasible instance of PUF, where all s_i and t_i are integers, is also a feasible instance of IntPUF, and vice versa.*

Proof. As the converse is trivial, we only need to consider a feasible solution to an instance of PUF, and transform it into a feasible solution, for the same numbers s_i and t_i, that enjoys the additional properties of an IntPUF solution.

Whenever a robot leaves a station i and moves back to i without visiting another station, it can just stay at station i. Whenever a robot moves from a station i to a neighboring station j, it can just move first at unit speed and then stay at station j for the remaining time.

Now we can partition the trajectory of each robot into 0-epochs and 1-epochs where the robot has speed 0 and 1, respectively. During a 1-epoch, a robot may change its moving direction at stations.

Let t be the start time of any 1-epoch. If t is not integer, we let the epoch start already at time $\lfloor t \rfloor$. That is, we move the entire 1-epoch back in time by $t - \lfloor t \rfloor$ time units. Because of the unit speed, every arrival and departure time of the robot at any station during the whole 1-epoch is rounded to the next smaller integer, too. This modification is done for all 1-epochs independently.

Applying Lemma 1 to the arrival and departure times at any station i we see that the solution remains valid (i.e., no robot departs before it arrives), and some waiting times between consecutive visits may increase, but they does not exceed the given integer bound t_i. □

For formal reasons we assume from now on that we have $m + 1$ stations, at the integer points $0, 1, \ldots, m$. That is, s_i simply becomes i. If there is no station at point i, we formally set $t_i := \infty$, however, t_0 and t_m are finite. Thus an instance of IntPUF is characterized by an *instance vector* $\mathbf{t} = (t_0, \ldots, t_m)$ whose $m + 1$ entries are positive integers or ∞ symbols. We may use the terms instance vector and instance interchangeably.

Two instance vectors $\mathbf{x} = (x_0, \ldots, x_m)$ and $\mathbf{y} = (y_0, \ldots, y_m)$ are in relation $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for all i. We then say that $\mathbf{x}$ is smaller than $\mathbf{y}$, and $\mathbf{y}$ is larger than $\mathbf{x}$, in the non-strict sense. We call $\mathbf{x}$ strictly smaller than $\mathbf{y}$, and $\mathbf{y}$ strictly larger than $\mathbf{x}$, if $\mathbf{x} \leq \mathbf{y}$ but $\mathbf{x} \neq \mathbf{y}$. Trivially, if $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x}$ is feasible, then $\mathbf{y}$ is feasible, too. We call a feasible instance vector with only finite entries a *minimal feasible vector* if no strictly smaller vector is feasible. These

concepts can be defined in literally the same way for IntPUF with one robot. The following theorem is merely the known solution to the one-robot case [1] adapted to IntPUF. A *zigzag route* between two points i and j is a trajectory that perpetually goes from i to j and back, at unit speed.

Theorem 2. *[1] The only minimal feasible instance vector of IntPUF with one robot is given by* $t_i = \max\{2i, 2(m-i)\}$ *for all* i*. Moreover, if an instance is feasible, then a zigzag route between* 0 *and* m *is a feasible (and optimal) solution.*

Proof. The robot must sometimes visit point 0, say at time t. Then the last visit of point i was at time $t - i$ or earlier, and the next visit of point i will be at time $t + i$ time or later. Hence $t_i \geq 2i$ is necessary for feasibility. By symmetry we also get $t_i \geq 2(m-i)$. Conversely, in the mentioned zigzag route, the maximum waiting time between two consecutive visits of any point i equals $\max\{2i, 2(m-i)\}$. □

We denote the vector in Theorem 2 by $F(m)$. This means: $F(1) = (2, 2)$, $F(2) = (4, 2, 4)$, $F(3) = (6, 4, 4, 6)$, $F(4) = (8, 6, 4, 6, 8)$, $F(5) = (10, 8, 6, 6, 8, 10)$, and so on. Characterizing the minimal feasible vectors for IntPUF with two robots appears to be far more complicated. However, suppose for the moment that, for some fixed size m, we know the list of all these minimal feasible vectors, along with a feasible solution for each of them. In fact, this list is finite, as a consequence of the famous Dickson's lemma, as explained below.

Vectors $\mathbf{x}$ and $\mathbf{y}$ are *incomparable* if neither $\mathbf{x} \leq \mathbf{y}$ nor $\mathbf{y} \leq \mathbf{x}$. Dickson's lemma (attributed to Dickson due to some result in [7]) states that every set of pairwise incomparable vectors of some fixed length (an antichain), with positive integers as entries, is finite. In other words, these vectors form a well-quasi ordering (WQO). Dickson's lemma has later been generalized, leading to a rich theory of WQO; see [11] for a historical note.

We could now solve any instance of IntPUF of a fixed size m as follows. First check whether there exists a solution where the two robots move in disjoint intervals $[0, v]$ and $[u, m]$, where $v < u$. For every fixed u and v, these are just two independent instances of the one-robot problem solved in Theorem 2. Even a naive implementation takes only $O(m^3)$ time. In all other cases, the intervals visited by the two robots intersect in some nonempty *shared interval* $[u, v]$, $u \leq v$. Note that every solution with a shared interval visits all $i \in [0, m]$. Hence, every instance vector $\mathbf{t}$ that admits a solution with a shared interval is larger than some feasible instance vector $\mathbf{t}'$ where all entries are finite, and trivially, $\mathbf{t}'$ is larger than some minimal feasible vector $\mathbf{t}''$. Thus it remains to check for the given instance vector $\mathbf{t}$ and every minimal feasible vector $\mathbf{t}''$ whether $\mathbf{t} \geq \mathbf{t}''$, and in the positive case, take a solution for $\mathbf{t}''$.

Not only the list of minimal partial solutions $\mathbf{t} = (t_0, \ldots, t_m)$ is finite, but each of them also has a solution with a finite description. The argument is as follows. Let us describe the situation at any integer time by the *state vector* $\mathbf{p} = (p_0, \ldots, p_m)$, where $p_i < t_i$ is the (integer) time that has passed since the last visit of i. Note that there is some robot at i if and only if $p_i = 0$. Since all t_i are finite, the number of state vectors is finite, too. Define the *state graph* of

$\mathbf{t}$ as the directed graph where the vertices are the state vectors, and a directed edge from $\mathbf{p}$ to $\mathbf{q}$ indicates that $\mathbf{q}$ is reachable from $\mathbf{p}$ in one time unit. The possible solutions to $\mathbf{t}$ are exactly the infinite directed paths in the state graph. But since the graph is finite, every solution contains some simple directed cycle (i.e., without repeated vertices). Conversely, every simple directed cycle is a solution. Thus we can choose any simple directed cycle, and such a solution is periodic. (A remark is that these matters are not so clear for PUF, as irrational numbers might have bizarre effects.) We have arrived at the following result, where we presume that arithmetic operations with integers, e.g., comparisons, cost constant time:

Lemma 2. *For any fixed m, once we know all minimal feasible instance vectors, we can solve every instance of IntPUF on $[0, m]$ in constant time. Moreover, every feasible instance admits a periodic solution, with a period bounded by some constant. (That is, time and period length are bounded by functions of m only.)*

We may effectively solve any instance $\mathbf{t} = (t_0, \ldots, t_m)$ also in the following way: Construct the state graph from $\mathbf{t}$ and find a simple directed cycle, or recognize that $\mathbf{t}$ is not feasible otherwise. However, the size of the state graph depends on $\mathbf{t}$. Therefore, Lemma 2 that moves some work to a preprocessing phase is a step of progress. (As a side remark, also in a logarithmic cost model, comparing $\mathbf{t}$ against a fixed finite list is much faster than using the state graph.) It remains the question whether we can construct the list of all minimal feasible instances effectively. The WQO argument yields only its finiteness, but in fact, we can provide an effective algorithm. However, we will not care about its running time as a function of m, which is a separate (and apparently difficult) matter.

Lemma 3. *There exists an algorithm that effectively constructs all minimal feasible instance vectors for any given m, along with some periodic solution for each of them, in a time that depends on m only.*

Proof. Trivially, some minimal feasible vector exists, and since there are only finitely many of them, there is some finite upper bound on all entries in them. Thus, if we try all vectors $(t, \ldots, t)$ for $t = 0, 1, 2, \ldots$, we will eventually find some feasible vector $\mathbf{t}$. Recall that every fixed vector can be tested for feasibility via its state graph. We test all vectors being smaller than our $\mathbf{t}$, thus identifying at least one minimal feasible vector. But we cannot stop here, as there may exist further minimal feasible vectors being incomparable to $\mathbf{t}$.

Next, assume that we have some nonempty set M of minimal feasible vectors, and we want to decide whether we have already found them all. Assume that $\mathbf{u} = (u_0, \ldots, u_m) \notin M$ is some further, yet unknown minimal feasible vector. Then, for every $\mathbf{t} = (t_0, \ldots, t_m) \in M$ there must exist some i with $u_i < t_i$. This observation suggests the following procedure: For every $\mathbf{t} \in M$ we select some i and set $v_i := t_i - 1$. If the same i is selected several times, we take the minimum of these v_i. If i is never selected, we set $v_i := \infty$. There exist only finitely many such selections, generating finitely many different vectors $\mathbf{v} = (v_0, \ldots, v_m)$. Now,

every minimal feasible vector $\mathbf{u} \in M$ has the following property: There exists some $\mathbf{v}$ such that $u_i \leq v_i$ holds for all i.

It remains to test for every vector $\mathbf{v}$ whether it has a feasible solution such that the waiting times of all stations i with finite v_i are actually bounded by these v_i, and the waiting times of all stations i with infinite v_i are bounded by some finite (but unspecified) integers. (Note carefully that the latter condition is also needed to obtain some minimal feasible vector smaller than $\mathbf{v}$; it is not enough to demand and test feasibility of $\mathbf{v}$, because this entails no condition on the points with $v_i = \infty$.) More compactly, the property to be tested is that $\mathbf{v}$ is larger than some feasible vector with only finite entries.

Therefore, we proceed similarly as earlier in the feasibility test for vectors with only finite entries, but we must generalize our notion of state graph; see the details below. If we find such a solution to some $\mathbf{v}$, we also examine all smaller vectors and obtain a new minimal feasible vector that we add to M. If no vector $\mathbf{v}$ has such a solution, we know that M was already complete.

Now we give the details of testing a vector $\mathbf{v}$. We define a state by the passed times $p_i < v_i$ for all i with finite v_i, and by the positions of the two robots. (The passed times for points i with $v_i = \infty$ are not recorded, and the robots' positions are now given explicitly.) As earlier, a directed edge from one state to another one indicates reachability in one time unit. As seen above, every instance that admits a solution, with finite bounds on the waiting times of all stations i, also has a solution with a finite period. In our generalized state graph, such periodic solutions correspond exactly to directed cycles C (not necessarily simple!) such that every i appears as a robot position on some vertex of C.

Let V_i denote the set of states where some robot is visiting i. Then, deciding the existence of a periodic solution boils down to the following graph problem: Given a directed graph and a family of subsets V_i of vertices, find some directed cycle that intersects every V_i. However, this is an easy problem: Such a cycle exists if and only if some strongly connected component of the graph intersects all V_i. The "only if" direction holds because every directed cycle is entirely in some strongly connected component, The "if" direction holds because we can freely navigate in a strongly connected component, and thus connect some vertices from every V_i to some directed cycle. In conclusion, we only need to compute the strongly connected components of the generalized state graph and check their intersections with all V_i. □

From Lemma 2 and 3 it follows:

Theorem 3. *There exists an algorithm solving IntPUF in a time that depends on m only. Moreover, every feasible instance admits a periodic solution, with a period bounded by some function of m.*

3 Short Waiting Times

The presence of some station j with a small t_j drastically restricts the robots' possible movements, as we will discuss now. Fix some j and define $h := \lfloor t_j/2 \rfloor$,

that is, $t_j = 2h$ or $t_j = 2h + 1$. Also define the interval $J := [j - h, j + h]$. At any integer time, some robot must be present in J. (Otherwise, the last visit of j was more than h time units ago, and the next visit will be more than h time units from now, such that j would have to wait at least $2h + 2 > t_j$ time units between two visits.) In other words, at any integer time, at most one robot can be outside J. We call it the *outer robot*, whereas the robot in J is called the *inner robot*. When both robots are in J, the assignment of these two roles is arbitrary, and we can swap the roles of the two robots if we want or need.

Suppose furthermore that $0 \notin J$ and $m \notin J$. Since the outer robot must repeatedly visit stations on both sides of J, it must repeatedly cross J in both directions. In the best case, the outer robot needs one time unit to skip J. In detail: At time t, the outer robot is at point $j - h - 1$, and at time $t + 1$, the outer robot (actually now the other one) is at point $j + h + 1$, or vice versa. It is equivalent to say that, even in this best case, the outer robot must solve the one-robot instance $(t_0, \ldots, t_{j-h-1}, t_{j+h+1}, \ldots, t_m)$ obtained by cutting out J. Now Theorem 2 implies that this vector must be larger than $F(m - 2h - 1)$. If $m \in J$ (or similarly, if $0 \in J$), we have $(t_0, \ldots, t_{j-h-1}) \geq F(j - h - 1)$ by a simpler argument: only the outer robot can visit the stations to the left of J.

We are ready to solve another special case of IntPUF to optimality:

Theorem 4. *For every m and j there exists exactly one minimal feasible vector with $t_j = 1$, which is $F(m - 1)$ with a 1 inserted after the first j entries.*

Proof. If $t_0 = 1$ then the inner robot must stay at point 0 all the time, while the outer robot must solve the one-robot instance $(t_1, \ldots, t_m)$, and the assertion follows from Theorem 2. The argument for $t_m = 1$ is similar. If $t_j = 1$ for some j with $0 < j < m$, we have the lower bound $(t_0, \ldots, t_{j-1}, t_{j+1}, \ldots, t_m) \geq F(m-1)$ for every feasible vector, as shown above. To show that the claimed vector is feasible, note that the outer robot can zigzag between 0 and m and always skip $J = \{j\}$ in only one time unit. Now the maximal waiting time of every station outside J is equal to the corresponding entry of $F(m-1)$. Hence, together with $t_j = 1$, these integers form a feasible vector matching the lower bound. □

Similarly it should be possible to characterize the minimal feasible vectors with some $t_j = k$ also for any fixed $k > 1$, but it turns out that we run into many case distinctions regarding the times needed to skip J and the lengths of the outer robot's trajectories. However, the above observations still lead to the approximation result below. (Note that we assume that the instance is already given in "compact" form, as the list of all n finite values t_i.)

Theorem 5. *There exists an algorithm for IntPUF which, for every feasible instance with $n \leq m + 1$ stations (with finite t_i) and with $k = \min_i t_i \leq m/4$, outputs some $(1 + O(k/m))$-feasible solution in $O(n)$ time.*

Proof. Fix some j with $t_j = k$, and define $h := \lfloor k/2 \rfloor$ and $J := [j - h, j + h]$ as before (even in the case when $0 \in J$ or $m \in J$). As we have seen, the instance vector after cutting out J must be feasible for one robot. With Theorem 2 we get $t_i \geq m - k$ for all $i \notin J$.

Let m' denote the distance between J and the farthest station (either 0 or m), that means, $m' := \max\{j - h, m - (j + h)\}$. Note that $m' \geq (m - k)/2$. Since the outer robot must visit this farthest station sometimes, the outer robot cannot be in J during some time interval I of duration at least $2m'$.

Next, let J' be the set of all $i \in J$ with $t_i \leq 2m' - 2k$. Since $4k \leq m$, we have $3k \leq m - k \leq 2m'$, hence $t_j = k \leq 2m' - 2k$, thus $j \in J'$, that is, $J' \neq \emptyset$. Let u and v be the leftmost and rightmost point, respectively, in J'. Finally, we define I' to be the time interval I truncated by k time units at both ends.

Since the length of I' is at least $2m' - 2k$, the inner robot must visit every station in J' at least once during I'. In particular, it must visit u during I'. Now assume for some $i \in J'$ that $t_i < 2(i - u)$. For the inner robot it is then impossible to visit i before u and again after u, within t_i time units. Since $i - u \leq 2h \leq k$, the two mentioned visits of i must still happen during I. But since the outer robot is not in J during I, it cannot do any of these visits either. This contradiction to feasibility shows $t_i \geq 2(i - u)$ for all $i \in J'$. Similarly we can prove $t_i \geq 2(v - i)$ for all $i \in J'$. Hence, if we simply let the inner robot zigzag in $[u, v]$, it visits all stations in J' frequently enough.

Our solution is now: Let the inner and outer robot zigzag in $[u, v]$ and $[0, m]$, respectively. It remains to analyze the waiting times of stations outside J'.

Consider any station $i \notin J$. Since the instance vector after cutting out J is feasible for one robot, the outer robot would always return to i within t_i time units if it could skip J. But in reality it may need $2k$ additional time units to traverse J twice. Since $t_i \geq m - k$, the waiting time is at most $t_i + 2k = (1 + 2k/t_i)t_i \leq (1 + 2k/(m - k))t_i$.

Consider any station $i \in J \setminus J'$. By definition we have $t_i > 2m' - 2k$. Also remember that $2m' \geq m - k$. The outer robot returns to i in a time at most $2m' + 2k = (2m' - 2k) + 4k < t_i + 4k = (1 + 4k/t_i)t_i < (1 + 4k/(2m' - 2k))t_i < (1 + 4k/(m - 3k))t_i$.

Altogether, the solution is $(1 + O(k/m))$-feasible. The time $O(n)$ is obvious: We must only find the smallest t_i and construct J' for determining u and v. □

For notational convenience we have formulated Theorem 5 for IntPUF, but the proof does not really use integrality, hence we also have immediately:

Theorem 6. *There exists an algorithm for PUF which, for every feasible instance with n stations, distance m between the outermost stations, and $k = \min_i t_i \leq m/4$, outputs some $(1 + O(k/m))$-feasible solution in $O(n)$ time.*

4 Rounding the Coordinates

A natural idea for solving PUF approximately is now to round all s_i and t_i to integers and apply the results for IntPUF. Given an instance P of PUF and an integer parameter k, we scale the time axis such that $k = \min_i t_i$. In other words, $\min_i t_i/k$ becomes the unit of time. The length unit on the line L is chosen such that the given maximum speed of robots is the unit speed. This setting is assumed throughout this section.

Before we discuss rounding, we study more generally what happens if the stations are slightly shifted. Let Q be an instance of PUF, obtained from P by moving every station by less than half the length unit. That is, n and the t_i are preserved, but the stations in Q are at points s'_i where $|s'_i - s_i| \leq 1/2$ for all i.

Let S be any feasible solution to the instance P. In general, S is not feasible for Q: Besides small delays we may even completely miss some visits, since a robot's trajectory may change its direction at some station, but the station may have been moved away from the turning point. Therefore we would like to modify S so as to construct a solution that is c-feasible for Q, with some "small" $c > 1$. We will modify the two robot trajectories independently, that is, in the following we only consider the trajectory of any single robot.

Lemma 4. *Let I be any time interval of duration r, let $J \subset L$ be any interval of length at most $r/2$, and let $a, b \in J$ be any points therein. Then a robot can move during I such that its trajectory starts in a, ends in b, and visits all of J.*

Proof. Assume that $a \leq b$. (The other case is symmetric.) We simply travel from a to the left end of J, then to the right end of J, and finally to b. Obviously, the robot can manage this in at most $2(r/2) = r$ time units. □

Some more special definitions will be needed; note that they refer to real (not integer) intervals: For a given time interval I, let $J(I) \subset L$ denote the interval of points visited by the considered robot during I. For any interval $J = [u, v] \subset L$ we define $J^+ := [u - 1/2, v + 1/2]$. In the following we temporarily allow robots to be faster than the unit speed.

Lemma 5. *Let I be a time interval of duration r, $J \subset L$ an interval of length at least $(r - 2)/2$, and $a, b \in J$. Assume that a robot can move such that its trajectory during I starts in a, ends in b, and visits all of J. Then there also exists a trajectory during I that starts in a, ends in b, visits all of J^+, and has a speed at most $1 + 4/(r - 4)$.*

Proof. Let u and v be the ends of J, that is, $J = [u, v]$, and let T be the assumed trajectory. Since T visits all of J, it must contain a sub-trajectory T_2 going from u to v (or vice versa, but this case is symmetric). Hence we can partition T into three sub-trajectories: T_1 going from a to u, T_2 going from u to v, and T_3 going from v to b. Note that T_1 and T_3 may be empty, if $a = u$ and $v = b$, respectively.

We modify T as follows. Immediately after T_1 we insert a piece going from u to $u - 1/2$ in $1/2$ time units, and immediately before T_3 we insert a piece going from $v+1/2$ to v in $1/2$ time units. Finally we adjust T_2 such that (i) it goes from $u - 1/2$ to $v + 1/2$ (to connect to the extended T_1 and T_3) and (ii) it needs one time unit less (to be used for the additional $1/2+1/2$ time units). We achieve (i) by stretching T_2 parallel to L, and we achieve (ii) by shrinking T_2 parallel to the time axis. Since J has a length at least $(r-2)/2$, the robot following the original T_2 has to travel a distance at least $(r-2)/2$, and it also needs at least $(r-2)/2$ time units. Travelling one length unit more in one time unit less increases the speed by a factor at most $((r-2)/2+1)/((r-2)/2-1) = 1+4/(r-4)$. □

Lemma 6. *Let P be an instance of PUF specified by s_i and t_i $(i = 1, \ldots, n)$, and let Q be an instance of PUF with the same size n and the same durations t_i but with station positions s'_i such that $|s'_i - s_i| \leq 1/2$ for all i. If P is c-feasible, then Q is $(1 + 4\sqrt{2}/\sqrt{k} + O(1/k))c$-feasible.*

Proof. We partition the time axis into intervals of some length r that we fix later. Consider any time interval I in this partitioning, and the trajectory T of either one of the robots, in some feasible solution to P.

If the length of $J(I)$ is at most $(r-2)/2$, then the length of $J(I)^+$ is at most $r/2$. In this case we apply Lemma 4 with $J := J(I)^+$. If $J(I)$ is longer than $(r-2)/2$, then we apply Lemma 5 with $J := J(I)$. Due to the Lemmas, in both cases we can replace the sub-path of the given trajectory T during I with a path that begins and ends in the same points as in T, but visits all of $J(I)^+$. However, in the second case we increase the speed by a factor up to $1 + 4/(r-4)$.

We do the described change independently in all intervals I of our partitioning, and we observe: (i) Since the robot's positions at the start and end moments of these time intervals have not changed, the modified trajectories form together a new trajectory T' overall. (ii) If T visits a station i during some time interval I of the partitioning, then $s_i \in J(I)$, hence $s'_i \in J(I)^+$, hence also T' visits the station i during I. Moreover, since I has duration r, this visit is by at most r time units earlier or later than in T.

The described changes are done independently for both robots. From the above property (ii) it follows that the waiting time between two consecutive visits of any station i increases by $2r$ time units in the worst case. Remembering that $k = \min_i t_i$ and $c \geq 1$, this implies that the waiting time is always at most $(1+2r/k)ct_i$. In other words, the modified solution would have been $(1+2r/k)c$-feasible if it had respected the speed limit.

In order to get unit speed again, we finally stretch the trajectories along the time axis by a factor $1+4/(r-4)$. This yields a valid $(1+2r/k)(1+4/(r-4))c$-feasible solution. Choosing $r := \sqrt{2k} + 4$ gives the assertion. □

Now we can describe an algorithm to solve any feasible instance P of PUF approximately. Note that it is not known in advance whether P is feasible; we must discuss this point later, as well as the choice of parameter k:

We decide on an integer k and choose time and length unit such that $k = \min_i t_i$ and the robots have unit speed, as already explained. Recall that m denotes the distance of the outermost stations.

In the following we distinguish two cases regarding k and m. The exact cut-off point is not that important, but we must decide on some suitable one.

If $m > 4k\sqrt{k}$, then we run the algorithm from Theorem 6 to solve P approximately. (Note that its prerequisites are satisfied here.) It yields some $(1 + O(k/m))$-feasible solution, hence some $(1 + O(1/\sqrt{k}))$-feasible solution to P, in $O(n)$ time.

If $m \leq 4k\sqrt{k}$, then we proceed as follows. We round every s_i to the closest integer. Ties are broken arbitrarily if s_i is an integer plus $1/2$. If several stations i end up on the same point, we only keep one of these stations with the smallest t_i and "mask" the others.

Due to Lemma 6, the obtained instance Q is c-feasible, for some $c = 1 + O(1/\sqrt{k})$. Next we replace every t_i with $t'_i := \lceil ct_i \rceil$. The obtained instance R is feasible, and by Theorem 1, R is also a feasible instance of IntPUF. Defining $k' := \min_i t'_i$ we also note that $k' = \Theta(k)$.

We run the algorithm from Theorem 3 to solve R exactly, in a time that depends on m only, and thus on k only. The computed feasible solution to R, which we denote S, is $(1 + O(1/\sqrt{k}))$-feasible for Q.

Finally we move all stations i (also the masked ones) back to their original positions s_i and apply Lemma 6 again in the opposite direction, to translate S into a solution to P. Since $(1 + O(1/\sqrt{k}))^2 = 1 + O(1/\sqrt{k})$, this solution to P is $(1 + O(1/\sqrt{k}))$-feasible, too.

It is crucial that this last step can be done effectively. By Theorem 3, we can always take a periodic solution S, with a period bounded by some function of m. Furthermore, the proof of Lemma 6 does not only show the existence of a $(1+O(1/\sqrt{k}))$-feasible solution but also describes a construction of this solution from the given one (here: from S). The approximation ratio $1+O(k/m)$ remains true if we choose $r := \lfloor \sqrt{2k} + 4 \rfloor$ (to have an integer value r). Then it suffices to modify the trajectories on some time interval of finite duration (the least common multiple of r and the period of S) and then to repeat this solution infinitely on the time axis. That is, our approximate solution to P is periodic, too. For any desired $\varepsilon > 0$ we may choose $k = \Theta(1/\varepsilon^2)$ with some suitable constant factor. Altogether this shows:

Theorem 7. *There exists an algorithm that outputs, for any feasible instance of PUF with n stations and for any prescribed $\varepsilon > 0$, some $(1+\varepsilon)$-feasible solution, in time $O(\max\{n, g(\varepsilon)\})$, where g is some function that depends on ε only.*

One can trivially check afterwards whether a solution is $(1+\varepsilon)$-feasible. If the algorithm failed to find such a solution, we know that the given instance P was not feasible. In that case we consider instances P_c obtained from P by replacing all t_i with ct_i. We may choose any factor $c > 1$ and apply the same algorithm to P_c. Either we get a $(1 + \varepsilon)$-feasible solution to P_c, or c was too small. Once our c is within a factor $1 + \varepsilon$ of the minimal c^* that makes P_{c^*} feasible, we get a $(1 + O(\varepsilon))$-approximate solution to P.

It remains to find such a near-optimal c efficiently. Trivially, P_c is feasible when $c = 2m/k$. Hence, if $k/m = \Omega(\varepsilon)$, then $O(\log(1/\varepsilon))$ steps of binary search are enough. The case of smaller k/m is more peculiar, but the concepts of Sect. 3 enable us to first find some c within a constant factor of c^* without binary search, in $O(n)$ time: Let J_c be the interval of length ck, having some station with minimum t_i in the center. As we have seen in Sect. 3, P_c is feasible only if the instance P_c after cutting out J_c is feasible for one robot. A necessary condition is that $ct_i \geq m - ck$ for all $i \notin J_c$. Hence we can pick any $i \notin J_c$ with $ct_i < m - ck$ and raise c until either $ct_i \geq m - ck$ or $i \in J_c$. (Calculation details are straightforward.) As c only grows in this process, we successively get rid of all stations $i \notin J_c$ with a too small ct_i. For the final value c' we have that no instance $P_{c'-\delta}$, $\delta > 0$, is feasible, hence $c^* \geq c'$. Assume that still $c'k/m = O(\varepsilon)$;

otherwise we have already reached the former case. Furthermore, $t_i \geq k$ holds for all i by definition. In particular, $c't_i \geq c'k$ holds for all $i \in J_{c'}$. Hence, if we generously set $c := 3c'$ and let the robots zigzag in $[0, m]$ and $J_{c'}$, respectively, we obtain a feasible solution to our current P_c. It follows $1 \leq c/c^* \leq 3$. Now we have also overcome the restriction that P must be feasible, and we arrive at:

Theorem 8. *PUF admits a polynomial-time approximation scheme.*

Concluding Remarks. Our PTAS is not yet practical. We have not bounded the time as a function of $1/\varepsilon$, and large k may be needed to beat the known $\sqrt{3}$-approximation [1]. However, we believe that our approach paves the way. To achieve practicality, we must *efficiently* solve IntPUF instances up to certain values of k and m, using the structure of cycles in the state graph. That is, we need an efficient version of the algorithm from Lemma 3.

References

1. Chuangpishit, H., Czyzowicz, J., Gąsieniec, L., Georgiou, K., Jurdziński, T., Kranakis, E.: Patrolling a path connecting a set of points with unbalanced frequencies of visits. In: Tjoa, A.M., Bellatreche, L., Biffl, S., van Leeuwen, J., Wiedermann, J. (eds.) SOFSEM 2018. LNCS, vol. 10706, pp. 367–380. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73117-9_26
2. Czyzowicz, J., Georgiou, K., Kranakis, E.: Patrolling. In: Flocchini, P., Prencipe, G., Santoro, N. (eds.) Distributed Computing by Mobile Entities, Current Research in Moving and Computing. LNCS, vol. 11340, pp. 371–400. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11072-7_15
3. Czyzowicz, J., Gąsieniec, L., Kosowski, A., Kranakis, E.: Boundary patrolling by mobile agents with distinct maximal speeds. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 701–712. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23719-5_59
4. Czyzowicz, J., Godon, M., Kranakis, E., Labourel, A., Markou, E.: Exploring graphs with time constraints by unreliable collections of mobile robots. In: Min Tjoa, A., Bellatreche, L., Biffl, S., van Leeuwen, J., Wiedermann, J. (eds.) SOFSEM 2018. LNCS, vol. 10706, pp. 381–395. Springer, Cham (2018)
5. Czyzowicz, J., Kosowski, A., Kranakis, E., Taleb, N.: Patrolling trees with mobile robots. In: Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., Garcia-Alfaro, J. (eds.) FPS 2016. LNCS, vol. 10128, pp. 331–344. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51966-1_22
6. Das, S., Di Luna, G.A., Gasieniec, L.A.: Patrolling on dynamic ring networks. In: Catania, B., Královič, R., Nawrocki, J., Pighizzini, G. (eds.) SOFSEM 2019. LNCS, vol. 11376, pp. 150–163. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10801-4_13
7. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Am. J. Math. **35**, 413–422 (1913)
8. Fishburn, P.C., Lagarias, J.C.: Pinwheel scheduling: achievable densities. Algorithmica **34**, 14–38 (2002)

9. Gąsieniec, L., Klasing, R., Levcopoulos, C., Lingas, A., Min, J., Radzik, T.: Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In: Steffen, B., Baier, C., van den Brand, M., Eder, J., Hinchey, M., Margaria, T. (eds.) SOFSEM 2017. LNCS, vol. 10139, pp. 229–240. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51963-0_18
10. Holte, R., Rosier, L.E., Tulchinsky, I., Varvel, D.A.: Pinwheel scheduling with two distinct numbers. Theor. Comput. Sci. **100**, 105–135 (1992)
11. Kruskal, J.B.: The theory of well-quasi-ordering: a frequently discovered concept. J. Comb. Theory A **13**, 297–305 (1972)
12. Lin, S.S., Lin, K.J.: A pinwheel scheduler for three distinct numbers with a tight schedulability bound. Algorithmica **19**, 411–426 (1997)