



Distance-Based Solution of Patrolling Problems with Individual Waiting Times

Peter Damaschke  

Department of Computer Science and Engineering, Chalmers University, Göteborg, Sweden

Abstract

In patrolling problems, robots (or other vehicles) must perpetually visit certain points without exceeding given individual waiting times. Some obvious applications are monitoring, maintenance, and periodic fetching of resources. We propose a new generic formulation of the problem. As its main advantage, it enables a reduction of the multi-robot case to the one-robot case in a certain graph/hypergraph pair, which also relates the problem to some classic path problems in graphs: NP-hardness is shown by a reduction from the Hamiltonian cycle problem, and on the positive side, the formulation allows solution heuristics using distances in the mentioned graph. We demonstrate this approach for the case of two robots patrolling on a line, a problem whose complexity status is open, apart from approximation results. Specifically, we solve all instances with up to 6 equidistant points, and we find some surprising effects, e.g., critical problem instances (which are feasible instances that become infeasible when any waiting time is diminished) may contain rather large individual waiting times.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases Patrolling, Periodic scheduling, Shortest path, Well-quasi ordering

Digital Object Identifier 10.4230/OASICS.ATMOS.2021.14

Acknowledgements The author would like to thank the master's students Anton Gustafsson and Iman Radjavi for providing experimental results and for critical discussion of an earlier draft.

1 Introduction

Planning of periodically returning complex tasks for an unlimited time horizon and with different request frequencies is an abundant type of problems. For example, traffic companies want to offer clocked connections with different frequencies, and they have to construct timetables to serve these demands. Similarly, vehicles of shipping agencies may have to pick up, transport, and deliver goods from and to certain points periodically, and within prescribed maximum time intervals, and their routes must be planned. However, in the present paper we consider a type of problems that is conceptually somewhat simpler, in that fixed places rather than routes must be served, in a certain sense.

A set of distinguished points is given. Each of them must be perpetually visited by some vehicle, such that at most some prescribed waiting time elapses between any two consecutive visits of this point. These waiting times are individual, that is, they can differ for different points. As an application example, certain important places in some technical installation must be visited for monitoring and maintenance purposes, or for fetching some product or removing garbage. Some places need attention more frequently than others. If several identical vehicles are available, it does not matter which vehicle serves which point. Rather, every point must always be served within the prescribed waiting time by *some* of the vehicles. The problem is to plan a schedule for all these visits.

In a more general setting, a number of such tasks must be perpetually done within prescribed waiting times as described above, but we have some more freedom: Any task can be performed at several alternative places, and we can arbitrarily choose one of them. For instance, a pipe or lead or supply line may be checked at different points; consumable goods that must be renewed periodically may be fetched at various places to choose from, etc.



© Peter Damaschke;
licensed under Creative Commons License CC-BY 4.0

21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021).

Editors: Matthias Müller-Hannemann and Federico Perea; Article No. 14; pp. 14:1–14:14

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

14:2 Distance-Based Solution of Patrolling Problems

Visiting points periodically is called patrolling. In this paper we introduce a generic PATROLLING problem which encompasses some classic path problems in graphs as well as various patrolling problems from the literature [1, 3, 4, 6, 7, 8, 9] where also applications are discussed. The following part is more abstract and technical.

An instance of the PATROLLING problem, as we define it now, consists of an undirected finite graph $G = (V, E)$ named the *position graph*, whose n vertices are called *positions*, furthermore, $m + 1$ subsets $P_i \subset V$ ($i = 0, \dots, m$) called *properties*, and positive integers t_i ($i = 0, \dots, m$), where t_i is called the *waiting time* of property P_i . In other words, a problem instance is a graph and a hypergraph (on the same vertex set) with integer hyperedge weights.

We may say that $v \in V$ “has the property” P_i if $v \in P_i$. Nothing special is assumed about the hyperedges P_i , and a position may have several properties. Note that our indices begin with $i = 0$; the only reason is that this will yield more convenient expressions when we study a specific class of instances later on.

As usual, a *cycle* C in a graph G is a sequence v_1, \dots, v_k of vertices $v_j \in V$ such that $v_j v_{j+1}$ is an edge for every $j \in \{1, \dots, k - 1\}$, and $v_k v_1$ is an edge, too. It is important to notice that C may cross itself, that is, vertices may appear in C multiple times. The start vertex is immaterial, that is, any cyclic shift $v_{j+1}, \dots, v_k, v_1, \dots, v_j$ is the same cycle. We may *walk* a cycle C perpetually, which means to go round C infinitely often. A *round trip* on a path is the cycle obtained by travelling the path back and forth. Now we are ready to specify the problem.

PATROLLING:

Given a graph and a hypergraph on the same vertex set, and an individual waiting time t_i for every hyperedge (property) P_i , find a *solution cycle* C in G , satisfying the following condition for every i : When we walk C , there are never t_i consecutive vertices without property P_i .

Next we connect this formal definition to the above scenario of moving vehicles. From now on we call them “robots” rather than vehicles, to comply with the terminology in earlier literature.

Imagine that some robot can move around in the graph. At every moment, the robot is at some position in V . Time is discrete, and in every time step the robot can move to some adjacent position. (We may also allow it to stay at its current position, but this option has no benefits, in terms of the problem.) For every index i , the robot must visit P_i at least once in every time interval of t_i steps, and in the case $|P_i| > 1$ it is immaterial which vertex with property P_i is chosen. The problem can also be formulated in directed graphs, but in this paper we consider only the undirected case where movements are reversible.

Most importantly, our formulation of PATROLLING also encloses cases where a fleet of $r > 1$ robots in a position graph H must perpetually visit every property P_i with waiting time at most t_i . (Note that it is not prescribed which of the robots visits P_i next. The only demand is that P_i must always be visited by *some* robot within the next t_i time units.) A reduction of this multi-robot version to PATROLLING is quite obvious: We define a position graph G whose vertices are the r -tuples of vertices of H , indicating all robots’ positions. Hence, for constant r , the blow-up is polynomial. Two vertices $(u_1, \dots, u_r) \neq (v_1, \dots, v_r)$ are adjacent if and only if, for every index j , the vertices u_j and v_j are identical or adjacent in H . Moreover, to any r -tuple we assign all properties of its entries, and no further property. Now, the single robot in G represents r robots in H .

In more complicated applications of PATROLLING, vertices of the position graph may model states of a system rather than points in space. For instance, a small factory may want to regularly produce an assortment of diverse products, but it must adapt its machines each time when it switches to another product. Then we may use vertices and edges to represent arrangements of the machines and transitions between them, respectively.

We remark that, in PATROLLING, we are actually looking for an infinite path I in G where, for every index i , no subpath of t_i vertices is disjoint from P_i . But for simple reasons we can aim at a solution cycle instead: If I exists, we can divide I in subpaths of $s := \max_i t_i$ vertices. Since at most n^s different such subpaths can exist, by the pigeonhole principle, some subpath with at most $s(n^s + 1)$ vertices begins and ends with the same subpath of s vertices, which contains at least one vertex of each property. By identifying these two ends we get a solution cycle. This justifies our formulation with a cycle, and it also shows the existence of some solution cycle with length at most sn^s , if (t_0, \dots, t_m) has a solution at all. If G has maximum degree Δ , then similarly we get the existence of a solution cycle of length at most $ns\Delta^s$. A quite different argument yields another upper bound of ns^m . We mention these further bounds here without proof, as we will not further use them.

However, some order-theoretic concepts will be central. For any two vectors V and W of $m + 1$ waiting times we write $V \leq W$ if every waiting time in V is smaller than or equal to the corresponding waiting time in W , and $V < W$ if $V \leq W$ but $V \neq W$.

Given a position graph with a family of $m + 1$ properties, we call an integer vector $T = (t_0, \dots, t_m)$ *feasible* if PATROLLING has a solution cycle with waiting time at most t_i for every property P_i , and we call a feasible vector T *critical* if no vector $T' < T$ is feasible. Also the solution cycle itself is called feasible or critical.

As we noticed in [5], for every fixed hypergraph, the number of critical vectors is finite, since the set of positive integer vectors of fixed length is well-quasi ordered (WQO). Hence also t is always bounded by some constant. However, WQO alone does not hint to specific bounds on the number of critical vectors, the maximum waiting times therein, and the lengths of solution cycles. Time bounds from naive exhaustive search would be prohibitive.

Therefore, the main goal of the present work is to provide heuristics for actually solving certain instances of PATROLLING, i.e., for constructing solution cycles or showing infeasibility. We will see that distances in the position graph are very informative for the problem.

A particularly intriguing case is the problem from [1] where two robots are patrolling on a line. This might appear to be a simple setting at first glance, but it is far from being simple. In [5] we have shown the existence of a PTAS: Any desired approximation ratio $1 + \varepsilon$ for the waiting times can be achieved by solving a discrete problem with $m + 1$ equidistant points, where m only depends on ε . But the weak spot is that just this discrete version is poorly understood, and even its complexity status (polynomial or NP-hard) is still open. (The existence of a PTAS does not hinge on this unknown complexity, as it requires polynomial time only for every fixed ε .) This is amazing, noticing that the one-robot problem on a line is trivial. Feasible two-robots instances can require some well choreographed “pas de deux”. As pointed out in [5], the practical usability of the PTAS depends on exact solutions to discrete problem instances up to some size m . Therefore we use this particular problem as our playground and case study for our approach to PATROLLING, although the ideas are by no means limited to this case. We may also use them for more robots and other topologies (trees, cycles, grids, etc.). Since they contain *two robots on a line* as a special case, it is sensible to start with that. It seems worth considering also for other reasons: By way of contrast, in PATROLLING in a metric space where all distances are 1 (known as PINWHEEL

SCHEDULING), r robots do not add interesting aspects; just the waiting times are scaled by a factor r , because all robots can freely jump. Two robots on a line seems to be the “simplest” nontrivial case where the number of robots matters, due to the underlying metric.

We notice that the PERIODIC LATENCY problem in [2] (where also further related problems with many potential applications are mentioned) is very similar to PATROLLING, and the multiple robots version on the line can be solved in polynomial time by dynamic programming. However, a crucial difference is that every point must be assigned to a unique robot there, whereas in PATROLLING, robots can share their work arbitrarily.

We apply the above simple reduction from two robots to one robot. While it complicates the position graph and the family of properties (see Section 3.1), this is more than compensated by simpler descriptions of solutions, just by cycles traversed by *one* token in *one* graph. Moreover, we can take advantage of distances in the position graph.

Contributions

Our first contribution is the generic formulation of the PATROLLING problem itself, as motivated above. Then we relate it to some classic path problems in graphs, and we provide some simple but powerful distance-based lower bounds on the waiting times. Next we use them to solve instances of the two-robots-on-a-line problem. We can solve them completely and for arbitrary m , when some small waiting times (up to 3) are present. For larger waiting times, the problem becomes considerably more intricate. To our surprise, instances with $t_k = 2$ and $t_{k+1} = 3$ for some index k are already more peculiar, and there exist many critical vectors under this constraint, some with pretty large t_0 and t_m . This tempers our initial hopes that it could be a practical method to enumerate all critical vectors and a solution to each. (Such an enumeration would trivially solve all other instances as well.) But the results show that this innocently looking problem is surprisingly deep and structurally rich. Nevertheless, to show these effects and also the power of the distance approach, we try to enumerate the critical vectors for small sizes m . In the paper we do this completely for $m \leq 5$ (remember that despite WQO this is not trivial even for fixed m) and still partially for $m = 6$. We conclude with the general lessons and directions for further research. In the technical part, readers may skip many of the detailed case inspections without losing track.

2 Distances in the Position Graph

Let $d(u, v)$ denote the distance of the vertices u and v in $G = (V, E)$, or in any related graph when it is clear from context. The distance is the length, i.e., number of edges, of a shortest u - v -path. For $X, Y \subset V$, an X - Y -path is any x - y -path where $x \in X$ and $y \in Y$. We define $d(X, Y) = \min\{d(x, y) \mid x \in X, y \in Y\}$ and abbreviate $d(\{x\}, Y)$ by $d(x, Y)$.

We use $R \subset V$ to denote any set satisfying that every solution cycle must be entirely in $G[R]$, the subgraph of G induced by R , and equipped with the properties $P_i \cap R$. The *range* of a property P_i to be the set $R_i := \{v \in V \mid \exists u \in P_i : d(u, v) \leq \lfloor t_i/2 \rfloor\}$. Obviously, every solution cycle must be entirely in R_i . Thus we may initially set $R := \bigcap_i R_i$. By applying further necessary conditions we might then be able to restrict R further.

Similarly, we use $R' \subseteq E$ to denote any set of edges satisfying that every solution cycle can only traverse edges in R' . Initially we can make R' the set of all edges in $G[R]$. But we might also be able to restrict R' further. For instance, if $t_i = 2$, then edges not incident to P_i can be deleted from R' .

As simple as the following lemma is, it gives powerful lower bounds on the waiting times in feasible instances.

► **Lemma 1.** *Let $t > 0$ be an integer, and P_i and P_j any two properties, where $t_i \leq 2t + 1$. Then every solution cycle satisfies the following:*

- *Every visit of P_j is at the link of a P_i - P_j path and a P_j - P_i path, one of which has length at most t .*
- *If $t = d(P_i, P_j)$ and $t_i = 2t$, then both mentioned paths have length t .*
- *If $t = d(P_i, P_j)$ and $t_i = 2t + 1$, then one of the mentioned paths has length t , and the other path has length t or $t + 1$.*
- *Moreover, we cannot visit any vertex v with $d(v, P_i) \geq t + 1$.*

Proof. Consider the path between the last/first visit of P_i before/after any visit of P_j . Since its length is at most $2t + 1$, one of the two mentioned subpaths has a length at most t . The next two assertions follow instantly from the definition of $d(P_i, P_j)$. Finally, the length of the path between the last/first visit of P_i before/after some visit of v would be at least $2(t + 1)$, which contradicts $t_i \leq 2t + 1$. ◀

Some reformulations and special cases of Lemma 1 are also useful: Applying it to $t := d(P_i, P_j) - 1$ for any two properties P_i and P_j , we get by contradiction that $t_i \geq 2t + 2 = 2d(P_i, P_j)$. Hence

$$t_i \geq 2 \max_j d(P_i, P_j)$$

holds for every fixed j . By setting instead $t := d(P_i, P_j)$ we get assertions for every solution cycle for $t_i \leq 2d(P_i, P_j) + 1$: In this case the mentioned paths of length t must be shortest P_i - P_j -paths. Let D be the vector with components $t_i := 2 \max_j d(P_i, P_j)$ for $i = 0, \dots, m$. The above inequality says that every feasible vector T satisfies $T \geq D$. If D itself is feasible, then D is therefore the only critical vector.

The following theorem is not difficult, however, it may be interesting to notice the connection to some classic path problems.

► **Theorem 2.** *PATROLLING with two properties is equivalent to finding some shortest path between the two hyperedges, and HAMILTONIAN CYCLE is polynomial-time reducible to PATROLLING, which is therefore NP-hard.*

Proof. Consider instances with only two properties X and Y . By Lemma 1, both waiting times must be at least $2d(X, Y)$. Conversely, if both waiting times are at least $2d(X, Y)$, then the round trip on any shortest X - Y -path is a solution cycle.

Next we present a reduction from HAMILTONIAN CYCLE. Given any graph with ν vertices, we declare every single vertex a property, and we set all waiting times equal to ν . Then any Hamiltonian cycle is obviously a solution cycle. Conversely, consider any subpath of ν vertices in a solution cycle for PATROLLING. Due to the waiting times ν , it must contain every vertex, hence it must contain every vertex exactly once. Furthermore, the next vertex in the cycle must equal the first vertex of this path. Thus we have identified a Hamiltonian cycle in the graph. ◀

We remark that membership in NP is unclear, because there may not exist a polynomial bound on the cycle length for PATROLLING in general, such that the standard way of verifying a solution in polynomial time is not available,

For the remainder of the paper we introduce some more terminology. A *constraint* is an inequality or equation of the form $t_k \leq c$ or $t_k = c$, with a constant c , or a conjunction of some of them. Given a graph and a hypergraph of properties, we call a constraint feasible if it can be satisfied by a feasible vector, and infeasible else. If a constraint contains equations, we call a vector *critical under the constraint* if it is feasible, but no waiting time outside the equations can be lowered. Such a vector is not necessarily critical, because it might still be possible to lower some of the waiting times that are fixed by the constraint.

3 Two Robots on a Line

3.1 The Position Graph

In the case of PATROLLING studied in [5], which is a discretized version of the problem from [1], two robots are patrolling on a line. To be precise, the position graph H is the path with $m + 1$ vertices indexed $0, \dots, m$, and the $m + 1$ properties are the single vertices.

We set up the stage where the following study will take place. Let x and y be integer variables for the robots' positions, where $x \geq y$. (Whenever the robots meet, we can swap their roles.) Then the vertex set of the position graph G from the reduction in Section 1 consists of all points with integer coordinates (x, y) , where $0 \leq y \leq m$, $0 \leq x \leq m$, and $x \geq y$. We informally call G “the triangle”, since its convex hull forms a right triangle with one cathetus on the x -axis and the hypotenuse on the line $y = x$. We can identify the hypotenuse with H , as i is mapped to (i, i) . Two vertices $(x, y) \neq (x', y')$ of G are adjacent if and only if both $|x - x'| \leq 1$ and $|y - y'| \leq 1$. Property P_i coming from vertex i of H is the union of the vertical line $x = i$ and the horizontal line $y = i$ which meet at point (i, i) on H . That is, every P_i is Γ -shaped, except for P_0 and P_m which are the catheti of the triangle.

As a detail, vertices on H are never needed in a solution cycle C : Assume that C contains a vertex (i, i) whose two neighbors in C are not in H . Then these two vertices are identical or adjacent. If both have the property P_i , we can simply remove (i, i) from C . If the neighbors are $(i, i - 1)$ and $(i + 1, i - 1)$, we can replace (i, i) in C with $(i + 1, i)$, and similarly in the symmetric case, or if both neighbors are $(i + 1, i - 1)$. If C contains a path of two or more vertices in H , then let $(i - 1, i - 1)$ and (i, i) be its end, hence the next vertex in C is $(i + 1, i)$ or $(i + 1, i - 1)$ or $(i, i - 1)$. In either case we can replace (i, i) in C with $(i, i - 1)$ or remove it. Thus we successively get rid of all vertices in H , hence it suffices to use the triangle of vertices (x, y) with $x > y$.

Defining $a_i := i - \lfloor t_i/2 \rfloor$ and $b_i := i + \lfloor t_i/2 \rfloor$, the range R_i of property P_i is the set of all vertices in the triangle that satisfy $a_i \leq x \leq b_i \vee a_i \leq y \leq b_i$. For the interval lengths we have $b_i - a_i = 2\lfloor t_i/2 \rfloor$, which equals t_i for even t_i , and $t_i - 1$ for odd t_i . Below we will describe the intersection $R := \bigcap_i R_i$ of ranges.

All critical solutions where the intervals $[0, m] \cap [a_i, b_i]$ in H have an empty intersection are obtained as follows [1]: For any fixed $d \in [1, m - 2]$, split H into $[0, d]$ and $[d + 1, m]$, and let one robot zigzag in each of these two parts. We rephrase this known result:

► **Proposition 3.** *For $m \geq 3$, all critical vectors with $\forall i : t_i \geq 2$ and $\bigcap_i [a_i, b_i] = \emptyset$ are given by $t_i = 2 \max\{i, d - i\}$ for all $i \leq d$, and $t_j = 2 \max\{j - d - 1, m - j\}$ for all $j \geq d + 1$, where d is any fixed integer with $1 \leq d \leq m - 2$.*

By the informal notion of a *BB path* (abbreviation of “billiard ball path”) we mean a path consisting of straight line segments with slope $+1$ or -1 that changes direction only by reflection at the border of R . The solution cycles to the critical vectors in Proposition 3 are then exactly the BB paths in $[d + 1, m] \times [0, d]$.

► **Definition 4.** *Let \mathcal{M}_0 denote the set of all critical vectors from Proposition 3.*

From now on we assume for the waiting times that the intervals $[0, m] \cap [a_i, b_i]$ in H have a nonempty intersection, which is denoted $[a, b] := [0, m] \cap \bigcap_i [a_i, b_i]$. With $a' := \max_i a_i$ and $b' := \min_i b_i$ we have $a = \max\{a', 0\}$ and $b = \min\{b', m\}$. These two numbers are cornerstones in the characterization of the intersection R of ranges given below.

R contains the stripes $a \leq x \leq b$ and $a \leq y \leq b$. Furthermore, R cannot contain vertices with $x < a$ or $y > b$, but R may intersect the rectangle $Q := [b+1, m] \times [0, a-1]$. For any vertex $(x, y) \in Q$, the following statements are equivalent: $(x, y) \in R \iff \forall i : (x, y) \in R_i \iff \forall i : y \geq a_i \vee x \leq b_i \iff \nexists i : y < a_i \wedge x > b_i$. Geometrically this means that $Q \cap R$ is obtained from Q by cutting out all quadrants with upper left corner of the form (b_i+1, a_i-1) , hence $Q \cap R$ is the region above some increasing staircase curve. Now we also characterize which of these quadrants intersect Q at all.

► **Lemma 5.** *Every feasible vector satisfies $t_i \geq 2i$ for $i < a$, and $t_j \geq 2(m-j)$ for $j > b$. Furthermore, if all these inequalities are satisfied, then $Q \cap R$ is obtained from Q by cutting out all quadrants with upper left corner of the form (b_k+1, a_k-1) , for all $k \in [a, b]$. In fact, we have $a_i \leq 0$ for $i < a$, and $b_j \geq m$ for $j > b$.*

Proof. As argued above, we must cut out exactly the mentioned quadrants; it remains to show that only the indices $k \in [a, b]$ are needed. Thus, consider any $i < a$. (For $j > b$ we proceed similarly.) From the definition of a_i and b_i we conclude $b_i - a_i + 1 \geq t_i$. Since no vertices (x, y) with $x < a$ are in R , only the horizontal line of P_i crosses R , which implies $2d(P_0, P_i) = 2i$ in R . Lemma 1 implies $t_i \geq 2d(P_0, P_i) = 2i$, or the instance is not feasible. Stacking these inequalities together, we obtain $b_i - a_i + 1 \geq 2i$. Since $b_i - a_i$ is even, this further implies $b_i - a_i \geq 2i$. Since also $(a_i + b_i)/2 = i$, it follows $a_i \leq 0$, thus the quadrant with upper left corner (b_i+1, a_i-1) does not intersect Q . ◀

The following lemma only rephrases some inequalities known from the definition of $[a, b]$ and from Lemma 5, and presents lower bounds on the waiting times for every fixed $[a, b]$.

► **Lemma 6.** *Every feasible vector satisfies $t_i \geq 2i$ and $t_i \geq 2(b-i)$ for all $i < a$, and similarly, $t_j \geq 2(m-j)$ and $t_j \geq 2(j-a)$ for all $j > b$.*

Another general remark is: Due to the inherent symmetry of the problem, every statement about a vector (t_0, \dots, t_m) holds also true for its reversal. To avoid many tiresome repetitions of this fact, from now on, every vector we talk about can also mean its reversal. In other words, we do not distinguish between (t_0, \dots, t_m) and (t_m, \dots, t_0) .

3.2 Short Waiting Times

In this section we study the consequences of the presence of the smallest possible waiting times t_k in a critical vector $(t_0, \dots, t_m) \notin \mathcal{M}_0$. The motivation is twofold. Firstly, R is then mainly composed of two narrow stripes, which should make the solution cycles relatively simple. Secondly, given some vector of waiting times (t_0, \dots, t_m) , even if we only aim at solutions with good approximation ratios rather than exact solutions, the smallest t_k could not be relaxed. The following theorem collects some cases of instances with small waiting times that can be completely solved. Quite surprisingly, the constraint $t_k = 2 \wedge t_{k+1} = 3$ turned out to be a much more subtle case (expect if $k = 1$), therefore it is not listed in the following result.

► **Theorem 7.** *PATROLLING for two robots on a line is solvable in $O(m)$ time when (t_0, \dots, t_m) contains some 1, or two neighbored 2s, or two neighbored 3s, or one 2 neighbored by two 4s. Moreover, each of the following constraints yields exactly one critical vector under the respective constraint:*

- $t_k = 1$ for any m and k ;
- $t_k = t_{k+1} = 2$ for $m \geq 3$ and $1 \leq k < k+1 \leq m-1$;
- $t_1 = 2 \wedge t_2 = 3$ for $m \geq 4$;
- $t_k = t_{k+1} = 3$ for $m \geq 5$ and $2 \leq k < k+1 \leq m-2$.

Furthermore, no critical vector has $t_0 = 3$ or $t_m = 3$, and every critical vector not captured by the above cases satisfies $T \geq (4, 4, 2, \dots, 2, 4, 4)$.

► **Definition 8.** We define \mathcal{M}_1 to be the set of the critical vectors under the constraints in Theorem 7.

The remainder of this section is devoted to the proof of Theorem 7. The scheme is as follows. We consider some constraint with some small waiting time(s) and the resulting position graph with vertex set R and edge set R' as defined in Section 3.1. Recall that R is the union of stripes $a \leq x \leq b$ and $a \leq y \leq b$ plus some subset of Q , and that $d(P_i, P_0) = i$ and $d(P_j, P_m) = m - j$ holds for all $i < a$ and $j > b$, implying $t_i \geq 2i$ and $t_j \geq 2(m - j)$. Note that we will define the position graph and interval $[a, b]$ using the considered constraints only. (They might further shrink due to other waiting times, but this does not affect the following conclusions.) In this position graph we will observe that $d(P_i, P_m) = m - i - c$ and $d(P_0, P_j) = j - c$ holds for some fixed number c and for certain (maybe all) indices $i < a$ and $j > b$. Then Lemma 1 also yields $t_i \geq 2(m - i - c)$ and $t_j \geq 2(j - c)$ for these indices. If we can construct a solution cycle that matches all lower bounds, it follows that the obtained vector of waiting times is the unique critical vector under the constraint.

Let $T = (t_0, \dots, t_m)$ always denote some critical vector. Wildcard symbol $*$ may be used for unspecified coordinates. We will frequently apply Lemma 5 to obtain the position graphs, and Lemma 1 and its consequences to obtain lower bounds, but without explicitly citing the lemmas, for the sake of brevity.

Constraint $t_0 = 1$ implies that R is the line $y = 0$ from $(1, 0)$ to $(m, 0)$. Note that $d(P_1, P_j) = j - 1$ and $d(P_j, P_j) = m - j$ for all j . The round trip on R yields optimal waiting times $t_j = 2 \max\{j - 1, m - j\}$ for all $j \geq 1$.

Constraint $t_k = 1$, for some k with $1 \leq k \leq m - 1$, implies $a = b = k$, hence R consists of the lines $x = k$ and $y = k$. That is, R is merely a path, and we have $c = 1$, as we can skip (k, k) . The round trip on R yields $t_i = 2 \max\{i, m - i - 1\}$ for all $i \leq k - 1$, and $t_j = 2 \max\{j - 1, m - j\}$ for all $j \geq k + 1$.

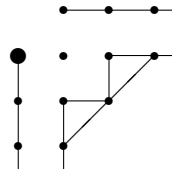
Every further critical vector T satisfies $T \geq (2, \dots, 2)$.

For $m \geq 3$, the set \mathcal{M}_0 contains the feasible vector given by the waiting times $t_0 = t_1 = 2$ and $t_j = 2 \max\{j - 2, m - j\}$ for all $j \geq 2$.

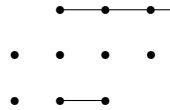
Constraint $t_0 \leq 3$ implies $b = 1$. Hence all P_j with $j \geq 2$ are vertical lines, thus their waiting times cannot be smaller than in the above solution above from \mathcal{M}_0 . It follows that this solution is the only critical vector with $t_0 \leq 3$. In particular, no critical vector with $t_0 = 3$ exists, and similarly for t_m .

Every further critical vector T satisfies $T \geq (4, 2, \dots, 2, 4)$.

Constraint $t_k = t_{k+1} = 2$, for some k with $1 \leq k < k+1 \leq m - 1$, implies $[a, b] = [k, k+1]$ and $c = 2$. See Figure 1. The round trip on the path consisting of the edge $(k+1, k-1)(k+2, k)$ and BB paths in the two stripes yields $t_i = 2 \max\{i, m - i - 2\}$ for all $i \leq k - 1$, $t_k = t_{k+1} = 2$, and $t_j = 2 \max\{j - 2, m - j\}$ for all $j \geq k + 1$. Hence this constraint admits exactly one critical vector.



■ **Figure 1** Position graph for the constraint $t_k = t_{k+1} = 2$. The picture illustrates the vertices in R and, for simplicity, the edges that do *not* belong to R' . We use the same convention also in all subsequent pictures. Vertex (k, k) is highlighted here.



■ **Figure 2** Position graph for the constraint $t_1 = 2 \wedge t_2 = 3$. The vertex in the lower left corner is $(1, 0)$.

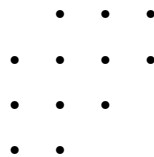
Constraint $t_1 = 2 \wedge t_2 = 3$ implies $[a, b] = [1, 2]$. See Figure 2. Since $d(P_0, P_m) = m - 3$, we have $t_0 \geq 2(m - 3)$.

Constraint $t_1 = 2 \wedge t_2 = 3 \wedge t_0 = 2(m - 3)$ forces every solution cycle to include some shortest P_0 - P_m -path, and thus the edge $(3, 0)(4, 1)$, if $m \geq 4$. Since $t_2 \leq 3$, the previous vertex must be $(2, 1)$, whose distance to P_j ($j \geq 3$) is $j - 2$. This shows $t_j \geq 2(j - 2)$ for every $j \geq 3$. For $m \geq 5$, the round trip on the path that begins with $(2, 1)(3, 0)(4, 1)$ and continues as BB path in the horizontal stripe attains these bounds: $t_0 = 2(m - 3)$, $t_1 = 2$, $t_2 = 3$, and $t_j = 2 \max\{j - 2, m - j\}$ for all $j \geq 3$. This shows that this constraint admits exactly one critical vector.

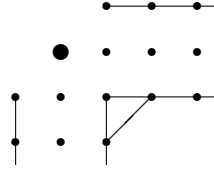
For $t_0 = 2(m - 3) + 1$, the conclusions in the previous paragraph still hold true. Since the vector with $t_0 = 2(m - 2)$, $t_1 = t_2 = 2$, and the same values t_j for all $j \geq 3$ is feasible (as seen earlier), we conclude that a larger t_0 does not yield further critical vectors. Thus, already the constraint $t_1 = 2 \wedge t_2 = 3$ admits only one critical vector.

Constraint $t_1 \leq 3 \wedge t_2 \leq 3$ implies that the solution cycle from $t_1 = 2 \wedge t_2 = 3$ still has optimal waiting times for the same reasons, hence raising t_1 from 2 to 3 does not provide a new critical vector.

Constraint $t_2 = t_3 = 3$ implies $[a, b] = [2, 3]$. See Figure 3. The round trip on $(2, 1)(3, 0)(4, 1)(5, 2) \dots (m, *)$, where the last part is a BB path in the horizontal stripe, achieves $t_0 = 2(m - 3)$, $t_1 = 2(m - 4)$, $t_2 = t_3 = 3$, and $t_j = 2 \max\{j - 2, m - j\}$ for all $j \geq 4$. From $d(P_0, P_m) = m - 3$ and $d(P_1, P_m) = m - 4$ we see that t_0 and t_1 are optimal. We claim that all t_j , $j \geq 4$, are optimal, too. In fact, since $d(P_j, P_0) = j - 3$, every solution cycle with $t_j \leq 2(j - 3) + 1$ would have to contain some shortest P_0 - P_j -path. But every such path contains the edge $(3, 0)(4, 1)$, and since $t_2 = 3$, the previous vertex must have $x = 2$. Hence every solution cycle touches the line $x = 2$ whose distance to P_j is $j - 2$.



■ **Figure 3** Position graph for the constraint $t_1 = 2 \wedge t_2 = 3$. The vertex in the lower left corner is $(2, 0)$. The horizontal stripe can be longer than displayed here.



■ **Figure 4** Position graph for the constraint $t_k = 2$. Vertex (k, k) is highlighted.

Constraint $t_k = t_{k+1} = 3$, for some k with $3 \leq k < k+1 \leq m-3$, implies $[a, b] = [k, k+1]$ and $c = 3$. The round trip on the BB path going with slope $+1$ through $(k+2, k-1)$ achieves $t_i = 2 \max\{i, m-i-3\}$ for all $i \leq k-1$, $t_k = t_{k+1} = 3$, and $t_j = 2 \max\{j-3, m-j\}$ for all $j \geq k+2$, and these waiting times are optimal under the mentioned constraint.

At this point, remember that every further critical vector T satisfies $T \geq (4, 2, \dots, 2, 4)$.

Constraint $t_1 = 2$ alone implies $a \leq b \leq 2$. Thus $t_0 \geq 4$, $t_1 = 2$, $t_2 \geq 4$, and $t_j \geq 2 \max\{j-3, m-j\}$ for all $j \geq 3$. Here, $t_2 \geq 4$ holds since $t_2 \leq 3$ was already treated earlier, and $t_j \geq 2(j-3)$ comes from $d(P_3, P_j) = j-3$. But these waiting times constitute some smaller vector in \mathcal{M}_0 . The conclusion is literally the same for $t_1 = 3$. It follows that every further critical vector T even satisfies $T \geq (4, 4, 2, \dots, 2, 4, 4)$.

Constraint $t_k = 2$, for some k with $2 \leq k \leq m-2$, implies $[a, b] = [k-1, k+1]$ and $c = 2$. See Figure 4. Here is a solution cycle that matches the resulting lower bounds: We traverse the path $Z = (k, k-2)(k+1, k-1)(k+2, k)$, then some BB path to P_m that starts and ends in $(k+2, k)$, then Z backwards, then some BB path to P_0 that starts and ends in $(k, k-2)$, and so forth. Regardless of the parities of k and $m-k$ and of the choice of the BB path in the case of even lengths, the waiting times are $t_i = 2 \max\{i, m-i-2\}$ for all $i \leq k-2$, and $t_j = 2 \max\{j-2, m-j\}$ for all $j \geq k+2$. The values of t_{k-1} and t_{k+1} achieved by this path depend on k and $m-k$, but they are at most 4.

3.3 The Smallest Instances

Using the general results on critical vectors with the smallest waiting times, we can now demonstrate how to solve the smallest instances for all or “most” vectors of waiting times. Cases with $m \leq 4$ are easy to settle and therefore omitted. Case $m = 5$ is still simple:

► **Proposition 9.** *For $m = 5$ there is no critical vector $T \notin \mathcal{M}_0 \cup \mathcal{M}_1$.*

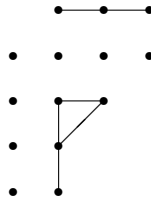
Proof. Let $T = (t_0, \dots, t_5)$. Note that $T \geq (4, 4, 2, 2, 4, 4)$. If $t_2 \leq 3$ then $d(P_0, P_5) \geq 3$, hence $T \geq (6, 4, 2, 2, 4, 6) \in \mathcal{M}_1$. Similarly we can rule out $t_3 \leq 3$. It follows $T \geq (4, 4, 4, 4, 4, 4) > (4, 2, 4, 4, 2, 4) \in \mathcal{M}_0$. ◀

But already the case $m = 6$ reveals the subtlety of the problem.

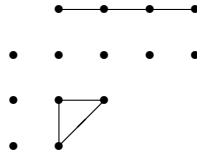
► **Proposition 10.** *For $m = 6$, all critical vectors $T \notin \mathcal{M}_0 \cup \mathcal{M}_1$ are $(6, 4, 4, 3, 4, 4, 6)$ and $(8, 4, 2, 4, 4, 6, 8)$, and several vectors $T \geq (8, 4, 2, 3, 4, 6, 8)$ that are critical under the constraint $t_2 = 2 \wedge t_3 = 3$.*

Proof. Let $T = (t_0, \dots, t_6)$. Note that $T \geq (4, 4, 2, 2, 2, 4, 4)$.

If $t_3 = 2$ then $t_2 \geq 3$ and $t_4 \geq 3$. If now $t_2 \geq 4$ and $t_4 \geq 4$ then (see the end of Section 3.2) we have $T \geq (8, 6, 4, 2, 4, 6, 8) > (8, 6, 2, 2, 4, 6, 8) \in \mathcal{M}_1$. Hence $t_2 = 3$ or $t_4 = 3$. It suffices to consider one of these symmetric cases. Constraint $t_3 = 2 \wedge t_4 = 3$ (see Figure 5) implies, by applying Lemma 1, that $T \geq (8, 6, 4, 2, 3, 6, 8) > (8, 6, 4, 2, 2, 6, 8) \in \mathcal{M}_1$. This excludes $t_3 = 2$ and shows $T \geq (4, 4, 2, 3, 2, 4, 4)$.



■ **Figure 5** Position graph for the constraint $t_3 = 2 \wedge t_4 = 3$. The leftmost line is $x = 3$.



■ **Figure 6** Position graph for $m = 6$ and the constraint $t_2 = 2 \wedge t_3 = 3$. The leftmost line is $x = 2$.

Let $t_3 = 3$. If some of the neighbors equals 3 as well, then we have a constraint from Theorem 7. Hence both neighbors are at least 4, or some equals 2. We also have $2 \leq a \leq b \leq 4$. Now $b = 2$ would imply $t_3 \geq 6$ by Lemma 6. Hence $b \geq 3$, therefore $t_0 \geq 6$, again by Lemma 6. By symmetry this also holds on the other side. Therefore $t_3 = 3$ implies $T \geq (6, 4, 2, 3, 2, 4, 6)$.

Constraint $t_2 = 4 \wedge t_3 = 3 \wedge t_4 = 4$ now yields $T \geq (6, 4, 4, 3, 4, 4, 6)$, which is achieved by the round trip on $(3, 0)(4, 1)(5, 2)(6, 3)$. Hence this is the only critical vector under this constraint.

It remains to study the constraint $t_2 = 2 \wedge t_3 = 3$, and vectors with $t_3 \geq 4$.

Assume that both $t_2 \geq 4$ and $t_4 \geq 4$, in other words, $T \geq (4, 4, 4, 3, 4, 4, 4)$. Assume that also $t_0 \leq 5$. Then $a \leq b \leq 2$. Hence, by Lemma 6, we even have $T \geq (4, 4, 4, 6, 4, 6, 8) > (4, 2, 4, 6, 4, 4, 6) \in \mathcal{M}_0$. This shows $t_0 \geq 6$ and by symmetry also $t_m \geq 6$, hence $T \geq (6, 4, 4, 3, 4, 4, 6)$, which was already feasible. It follows that $t_2 \leq 3$ or $t_4 \leq 3$. By symmetry we can suppose $t_2 \leq 3$.

This yields $1 \leq a \leq 2 \leq b \leq 3$, thus $T \geq (4, 4, 2, 3, 4, 6, 8)$ by Lemma 6. From $d(P_0, P_6) \geq 3$ we also get $t_0 \geq 6$, thus $T \geq (6, 4, 2, 3, 4, 6, 8)$. Now, if $t_2 = 3$ then $T \geq (6, 4, 3, 3, 4, 6, 8) \in \mathcal{M}_1$. This finally shows $t_2 = 2$. Since now the edge $(3, 0)(4, 1)$ is no longer in R' , the bounds are further raised to $d(P_0, P_6) \geq 4$ and $T \geq (8, 4, 2, 3, 4, 6, 8)$. If $t_3 = 4$ then we get the critical vector $(8, 4, 2, 4, 4, 6, 8)$, achieved by the cycle $(2, 0)(3, 1)(4, 2)(5, 3)(6, 2)(5, 1)(4, 2)(3, 1)$. So there only remains the constraint $t_2 = 2 \wedge t_3 = 3$. ◀

Figure 6 shows the position graph for the constraint $t_2 = 2 \wedge t_3 = 3$ that we discuss a bit further now.

Recall that critical vectors T under this constraint satisfy $T \geq (8, 4, 2, 3, 4, 6, 8)$. Since $(8, 6, 2, 2, 4, 6, 8) \in \mathcal{M}_1$, every such critical vector has $t_1 \leq 5$. Hence every solution cycle must contain some P_1 - P_6 path of length 2, and therefore the path $(3, 2)(4, 1)(5, 2)(6, 3)$.

Constraint $t_1 = 4 \wedge t_2 = 2 \wedge t_3 = 3$ requires this path to appear on both sides of $(6, 3)$, which enforces $(3, 2)(4, 1)(5, 2)(6, 3)(5, 2)(4, 1)(3, 2)$.

Constraint $t_2 = 2 \wedge t_3 = 3 \wedge t_4 = 4$ enforces, by similar arguments, every solution cycle to contain a path $(4, 2)(3, 1)(2, 0)(3, 1)(4, 2)$.

Moreover, every visit of P_6 and P_0 , respectively, necessarily happens within such a path.

Constraint $t_2 = 2 \wedge t_3 = 3 \wedge t_4 = 4 \wedge t_5 = 6$ enforces $(5, *) (4, 2)(3, 1)(2, 0)(3, 1)(4, 2)(5, *)$.

Constraint $t_1 = 4 \wedge t_2 = 2 \wedge t_3 = 3 \wedge t_4 = 4 \wedge t_5 = 6$ now obviously enforces both $(3, 2)(4, 1)(5, 2)(6, 3)(5, 2)(4, 1)(3, 2)$ and $(5, *) (4, 2)(3, 1)(2, 0)(3, 1)(4, 2)(5, *)$, and since these paths “disagree”, they must occupy disjoint subpaths of any solution cycle. Finally, consider any “consecutive” visits of P_6 and P_0 , that is, without other such visits in

between. They are surrounded by the mentioned paths, and furthermore, at least some $(4,*)$ must exist between them. However, a single vertex $(4,*)$ is not enough, because then P_1, P_2, P_3 must all be visited by $(4,*)(5,*)$, which is obviously impossible. Hence we must place at least two vertices there. On the other hand, the round trip on, for instance, $(6,3)(5,2)(4,1)(3,2)(3,1)(4,2)(5,3)(4,2)(3,1)(2,0)$ has waiting times $(18, 4, 2, 3, 4, 6, 18)$, and this vector is critical.

This example illustrates two aspects: Small waiting times in the middle can cause very large waiting times at the ends, and the distance lower bounds are strong enough to uniquely identify large parts of the solution cycles, which makes their construction quite efficient. Systematic search with the help of a computer program¹ produced, e.g., as many as 10 critical vectors for $m = 6$ under the constraint $t_2 = 2 \wedge t_3 = 3$, and the waiting time 18 in the example above is the highest one appearing in them. Slightly relaxed waiting times at inner points allow smaller waiting times at the ends, and some combinations of times can be chosen independently, which explains the exploding number of critical vectors. Enumeration for slightly larger m gave a similar picture, after considerably larger computation time, and with growing numbers of critical vectors.

4 Discussion and Conclusions

We have formulated the PATROLLING problem, even with several robots, as a problem dealing with only one vehicle that has to visit “properties” in one so-called position graph. (This plays a bit with the ambiguity of the word property which can also mean an object located somewhere.) The main advantage is that we can use the distances in this graph for solving instances of the problem: They yield simple lower bounds on the waiting times, moreover, waiting times in critical vectors are often equal to (or close to) these lower bounds. Hence their solution cycles must traverse shortest paths between the respective properties, and they are sometimes even unique. Moreover, certain combinations of (e.g., small) waiting times make the resulting position graphs simple. All this facilitates the construction of solution cycles or the verification of infeasible instances.

We have mainly studied the case of two robots patrolling on a line, whose complexity is open. As argued in Section 1, this problem is not as narrow as one might think, rather, it is the natural case to study first. The above ideas are used to determine all critical vectors for the smallest m , partly manually (as shown here), and partly supported by an implementation using further pruning techniques that are hard to summarize here. (However, a small side remark is that, due to the shape of the position graphs, shortest paths can be computed very quickly by a greedy algorithm.)

We found surprisingly many critical vectors, mainly caused by a certain combination of small waiting times in the interior of the line segment. Although pre-computation and plain enumeration of the critical instances is apparently not the method of choice for solving *given* single instances (as initially hoped), this study gives useful pointers to solution techniques. Still, vectors being far from criticality seem to be harder to solve. The ultimate goal would be to generalize these observations, in order to derive either a polynomial algorithm, or at least a better practical approximation algorithm than in [1, 5], or to identify gadgets for an NP-hardness proof. (We remark that unresolved complexities are typical in this field, e.g., the complexity status of PINWHEEL SCHEDULING [6, 8, 9] is notoriously open.)

¹ provided by Anton Gustafsson and Iman Radjavi

Note that a superpolynomial number of critical vectors does not yet rule out a polynomial algorithm. Furthermore, the methods we have demonstrated here to construct *all* critical vectors may similarly be used heuristically to find *some* critical vector being smaller than any given input vector T , or proving T infeasible. Some hope for fast algorithms comes from a certain “stratification” of waiting time vectors: We observed that instances containing some small waiting times behave differently than instances where all waiting times are large in relation to m , and also the size of the intersection $[a, b]$ of ranges plays some role.

Other open problems for two robots on a line, besides the complexity, have arisen: Is the number of critical vectors actually exponential in m ? Are the largest waiting times linearly bounded in m ? How long can their solution cycles be, in the worst case?

Moreover, since two robots on a line is only a special case and a first testbed, we would like to apply our insights also to more robots and more general topologies, such as trees and grids. But it seems to be a reasonable procedure to first aim for a thorough understanding of the “smallest” non-trivial case. The heuristics developed here should extend more or less straightforwardly to more general cases of PATROLLING. But we do not expect to transform results to seemingly similar problems, e.g., as mentioned earlier, PERIODIC LATENCY behaves differently and seems to be simpler from the outset.

References

- 1 Huda Chuangpishit, Jurek Czyzowicz, Leszek Gasieniec, Konstantinos Georgiou, Tomasz Jurdzinski, and Evangelos Kranakis. Patrolling a path connecting a set of points with unbalanced frequencies of visits. In A Min Tjoa, Ladjel Bellatreche, Stefan Biffl, Jan van Leeuwen, and Jirí Wiedermann, editors, *SOFSEM 2018: Theory and Practice of Computer Science – 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29 – February 2, 2018, Proceedings*, volume 10706 of *Lecture Notes in Computer Science*, pages 367–380. Springer, 2018. doi:10.1007/978-3-319-73117-9_26.
- 2 Sofie Coene, Frits C. R. Spijksma, and Gerhard J. Woeginger. Charlemagne’s challenge: The periodic latency problem. *Oper. Res.*, 59(3):674–683, 2011. doi:10.1287/opre.1110.0919.
- 3 Jurek Czyzowicz, Konstantinos Georgiou, and Evangelos Kranakis. Patrolling. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 371–400. Springer, 2019. doi:10.1007/978-3-030-11072-7_15.
- 4 Jurek Czyzowicz, Adrian Kosowski, Evangelos Kranakis, and Najmeh Taleb. Patrolling trees with mobile robots. In Frédéric Cuppens, Lingyu Wang, Nora Cuppens-Boulahia, Nadia Tawbi, and Joaquín García-Alfaro, editors, *Foundations and Practice of Security – 9th International Symposium, FPS 2016, Québec City, QC, Canada, October 24-25, 2016, Revised Selected Papers*, volume 10128 of *Lecture Notes in Computer Science*, pages 331–344. Springer, 2016. doi:10.1007/978-3-319-51966-1_22.
- 5 Peter Damaschke. Two robots patrolling on a line: Integer version and approximability. In Leszek Gasieniec, Ralf Klasing, and Tomasz Radzik, editors, *Combinatorial Algorithms – 31st International Workshop, IWOCA 2020, Bordeaux, France, June 8-10, 2020, Proceedings*, volume 12126 of *Lecture Notes in Computer Science*, pages 211–223. Springer, 2020. doi:10.1007/978-3-030-48966-3_16.
- 6 Peter C. Fishburn and J. C. Lagarias. Pinwheel scheduling: Achievable densities. *Algorithmica*, 34(1):14–38, 2002. doi:10.1007/s00453-002-0938-9.

14:14 Distance-Based Solution of Patrolling Problems

- 7 Leszek Gasieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM 2017: Theory and Practice of Computer Science – 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2017. doi:10.1007/978-3-319-51963-0_18.
- 8 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.
- 9 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.