On Cyclic Solutions to the Min-Max Latency Multi-Robot Patrolling Problem

Peyman Afshani
Department of Computer Science, Aarhus University, Denmark

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands

Kevin Buchin
Department of Computer Science, TU Dortmund, Germany

Jie Gao
Department of Computer Science, Rutgers University; New Brunswick, NJ 08854, USA

Maarten Löffler
Department of Information and Computing Sciences, Utrecht University, the Netherlands

Amir Nayyeri
School of Electrical Engineering and Computer Science, Oregon State University, OR 97330, USA

Benjamin Raichel
Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA

Rik Sarkar
School of Informatics, University of Edinburgh, Edinburgh, U.K.

Haotian Wang
Department of Computer Science, Rutgers University; New Brunswick, NJ 08854, USA

Hao-Tsung Yang
School of Informatics, University of Edinburgh, Edinburgh, U.K.

Abstract

We consider the following surveillance problem: Given a set P of n sites in a metric space and a set R of k robots with the same maximum speed, compute a *patrol schedule* of minimum latency for the robots. Here a patrol schedule specifies for each robot an infinite sequence of sites to visit (in the given order) and the latency L of a schedule is the maximum latency of any site, where the latency of a site s is the supremum of the lengths of the time intervals between consecutive visits to s.

When $k = 1$ the problem is equivalent to the travelling salesman problem (TSP) and thus it is NP-hard. For $k \geqslant 2$ (which is the version we are interested in) the problem becomes even more challenging; for example, it is not even clear if the decision version of the problem is decidable, in particular in the Euclidean case.

We have two main results. We consider *cyclic solutions* in which the set of sites must be partitioned into ℓ groups, for some $\ell \leqslant k$, and each group is assigned a subset of the robots that move along the travelling salesman tour of the group at equal distance from each other. Our first main result is that approximating the optimal latency of the class of cyclic solutions can be reduced to approximating the optimal travelling salesman tour on some input, with only a $1 + \varepsilon$ factor loss in the approximation factor and an $O\left((k/\varepsilon)^k\right)$ factor loss in the runtime, for any $\varepsilon > 0$. Our second main result shows that an optimal cyclic solution is a $2(1 - 1/k)$-approximation of the overall optimal solution. Note that for $k = 2$ this implies that an optimal cyclic solution is optimal overall. We conjecture that this is true for $k \geqslant 3$ as well.

The results have a number of consequences. For the Euclidean version of the problem, for instance, combining our results with known results on Euclidean TSP, yields a PTAS for approximating an optimal cyclic solution, and it yields a $(2(1 - 1/k) + \varepsilon)$-approximation of the optimal unrestricted (not necessarily cyclic) solution. If the conjecture mentioned above is true, then our algorithm is actually a PTAS for the general problem in the Euclidean setting. Similar results can be obtained by combining our results with other known TSP algorithms in non-Euclidean metrics.

38th International Symposium on Computational Geometry (SoCG 2022).
Editors: Xavier Goaoc and Michael Kerber; Article No. XX; pp. XX:1–XX:24

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures → Computational geometry

Keywords and phrases Approximation, Motion Planning, Scheduling

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.XX

Funding *Mark de Berg*: Supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.
Jie Gao: This work is supported by NSF OAC-1939459, CCF-2118953 and CCF-1934924.
Benjamin Raichel: Partially supported by NSF CAREER Award 1750780.

1 Introduction

We study the following problem, motivated by the problem of monitoring a fixed set of locations using autonomous robots: We are given a set $P = \{s_1, \cdots, s_n\}$ of n sites in a metric space as well as a set $R = \{r_1, \cdots, r_k\}$ of k robots. We assume the robots have the same maximum speed, called the *unit speed*, and their task is to repeatedly visit (i.e., survey) the sites such that the maximum time during which any site is left unmonitored is minimized. More precisely, we wish to compute a *patrol schedule*; that is, an infinite sequence of sites to visit for each robot, of minimum *latency*. Here the latency of a site s_i is the supremum of the length of the time intervals between consecutive visits of s_i, and the latency of the patrol schedule is the maximum latency over all the sites.

Related Work. For $k = 1$, the problem reduces to the Traveling Salesman Problem. To see this, consider the time interval $[0, 3L]$, where L is the optimal latency, and observe that every site is visited at least twice by the robot in this time interval. Let $L' \leqslant L$ be the maximum length of time between two consecutive visits of a site. Then there exists a site that is visited at times t_0 and $t_0 + L'$ and all other sites are visited at least once in the time interval $(t_0, t_0 + L')$. Hence, if an optimal solution has latency L, there is a TSP tour of length at most L. The converse is clearly true as well—by repeatedly traversing a TSP tour of length L we obtain a patrol schedule of latency L—and so the TSP problem is equivalent to the patrol problem for a single robot. Since TSP is NP-hard even in the Euclidean case [16] we will focus on approximation algorithms. There are efficient approximation algorithms for TSP and, hence, for the patrolling problem for $k = 1$. In particular, there is a $(3/2)$-approximation for metric TSP [5] (which was slightly improved very recently [10]) and a PTAS for Euclidean TSP [4, 15]. However, it seems difficult to generalize these solutions to the case $k \geqslant 2$, because it seems non-trivial to get a grip on the structure of optimal solutions in this case. We will mention some of the major challenges shortly.

There has been a lot of work on such surveillance problems in the robotics community [7, 9, 14, 22, 17, 18]. Most previous work, however, focused on either practical settings or aspects of the problems other than finding the best approximation factor. There are two papers that provide theoretical guarantees for the weighted version of the problem, where sites of higher weight require more frequent patrols. Alamdari *et al.* [2] provided a $O(\log n)$-approximation algorithm for the weighted problem for $k = 1$. (Due to existence of weights, a TSP tour may no longer be optimal for $k = 1$.) Afshani *et al.* [1] studied the problem for $k \geqslant 1$ and they present an $O(k^2 \log \frac{w_{\max}}{w_{\min}})$-approximation algorithm, where $w_{\max}$ and $w_{\min}$ are the maximum and the minimum weights of the sites.

Related Problems. As already mentioned, the TSP problem can be viewed as a special case of the problem for unweighted sites and for $k = 1$. Another related problem is the *k-path cover*

problem where we want to find k paths that cover the vertices of an edge-weighted graph such that the maximum length of the paths is minimized. This problem has a 4-approximation algorithm [3]. Another problem is the problem of covering all the sites with k trees that minimize the maximum length of the trees; this problem is known as the *min-max tree cover* problem and it has constant-factor approximation algorithms [3, 13] with 8/3 being the current record [21]. The *k-cycle cover* problem is similar, except that we want to use k cycles (instead of paths or trees); again constant-factor approximation algorithms are known, with 16/3 being the current record [21]. If the goal is to minimize the sum of all cycle lengths, there is a 2-approximation for the metric setting and a PTAS in the Euclidean setting [11, 12]. Our problem is also related to (but different from) the *vehicle routing problem* (VRP) [6], which asks for k tours, starting from a given depot, that minimize the total transportation cost under various constraints; see the surveys by Golden *et al.* [8] or Tóth and Vigo [20].

Our results. All covering problems mentioned above are obviously decidable. The question of decidability for the patrolling problem seems non-trivial. However, since patrol schedules are infinite sequences and thus it is not even clear how to guess a solution[1]. To tackle this issue, we consider the class of *cyclic solutions*. In a cyclic solution the set P of sites is partitioned into $\ell \leqslant k$ subsets $P_1, \cdots, P_\ell$, and each subset P_i is assigned k_i robots, where $\sum_{i=1}^{\ell} k_i = k$. The k_i robots are then distributed evenly along a TSP tour of P_i, and they traverse the tour at maximum speed. Thus, the latency of the sites in P_i equals $\|T_i\|/k_i$, where $\|T_i\|$ is the length of the TSP tour of P_i.

The significance of this definition is that in Section 3 we prove that (in any metric space) the best cyclic solution is a $2(1 - 1/k)$-approximation of the optimal solution in terms of maximum latency. We do this by transforming an optimal solution to a cyclic one, with only a $2(1 - 1/k)$ factor loss in the approximation ratio. This proof is highly non-trivial and involves a number of graph-theoretic arguments and carefully inspecting the coordinated motion of the k robots, cutting them up at proper locations, and re-gluing the pieces together to form a cyclic solution. In combination with this, in Section 4 we prove that, given a γ-approximation algorithm for TSP, for any fixed k and $\varepsilon > 0$, we can obtain a $(1 + \varepsilon)\gamma$-approximation of the best cyclic schedule in polynomial time. Therefore, in the Euclidean setting, we can use a known PTAS to obtain a $(1 + \varepsilon)$-approximation to the *best cyclic* solution and in the general metric setting, we can use known approximation algorithms for TSP [10] to get a 1.5-approximation to the *best cyclic* solution. Together with the results in Section 3 these lead to a $(2 - 2/k + \varepsilon)$-approximation algorithm for the Euclidean case, and a $(3 - 3/k)$-approximation for general metrics.

We conjecture that the best cyclic solution is in fact the best overall solution. If this is true, then our algorithm in Section 4 already gives a PTAS in the Euclidean setting. Observe that a corollary of our result in Section 3 is that the conjecture holds for $k = 2$. We remark that there is an easy proof showing the existence of a cyclic 2-approximation solution (See Section 2.2). Our new bound $2(1 - 1/k)$ is a significant improvement when k is a small constant. For example, for $k = 3$, we get that a cyclic 4/3 approximate solution exists, and for $k = 2$ –as mentioned above– that there is a cyclic optimal solution.

[1] If we assume that all distances are integers and we want to decide whether the latency is at most a given integer ℓ, then we can guess a solution as laid out in Appendix B. These assumptions, however, do not hold in the Euclidean case, even if the coordinates of sites are rational.

2 Challenges, Notation, and Problem Statement

2.1 Notation and Problem Statement

Let (P, d) be a metric space on a set P of n sites, where the distance between two sites $s_i, s_j \in P$ is denoted by $d(s_i, s_j)$. Following Afshani *et al.* [1], we model the metric space in the following way. We take the undirected complete graph $G = (P, P \times P)$, and we view each edge $(s_i, s_j) \in P \times P$ as an interval (that is, a continuous 1-dimensional space) of length $d(s_i, s_j)$ in which the robot can travel. This transforms the discrete matric space (P, d) into a continuous metric space $C(P, d)$. From now on, and with a slight abuse of terminology, when we talk about the metric space (P, d) we actually mean the continuous metric space $C(P, d)$.

We allow the robots to "stay" on a site for any amount of time. This implies it never helps if a robot moves slower than the maximum speed: indeed, the robot may as well move at maximum speed towards the next site and stay a bit longer at that site. Also, it does not help to have a robot start at time $t = 0$ "in the middle" of an edge, so we can assume all robots start at some sites at the beginning. A *schedule* of a robot r_j is defined as a continuous function $f_j : \mathbb{R}^{\geqslant 0} \to C(P, d)$, where $f_j(t)$ specifies the position of r_j at time t. The unit-speed constraint implies that a valid schedule must satisfy $d(f_j(t_1), f_j(t_2)) \leqslant |t_1 - t_2|$ for all t_1, t_2. A *schedule for the collection R of robots*, denoted by $\sigma(R)$, is a collection of schedules f_j, one for each robot $r_j \in R$. Note that we allow robots to be at the same location at the same time.

We say that a site $s_i \in P$ is *visited* at time t if $f_j(t) = s_i$ for some robot r_j. Given a schedule $\sigma(R)$, the *latency* L_i of a site s_i is defined as follows.

$$L_i = \sup_{0 \leqslant t_1 < t_2} \{|t_2 - t_1| : s_i \text{ is not visited during the time interval } (t_1, t_2)\}$$

We only consider schedules where the latency of each site is finite. Clearly such schedules exist; e.g., a robot can repeatedly traverse a TSP tour of the sites. Given a metric space (P, d) and a collection R of k robots, the *(multi-robot) patrol-scheduling problem* is to find a schedule $\sigma(R)$ minimizing the latency $L := \max_i L_i$, the maximum latency of any site.

2.2 Challenges

The problem of scheduling multiple robots is quite challenging and involves several subtleties, caused by the fact that patrol schedules are infinite sequences. For example, the time intervals between consecutive visits of any given site might increase continuously, and so we have to define the latency of a site using the notion of supremum rather than maximum. Moreover, for $k > 1$, it is not even clear if the problem is decidable: Given a set of n points in the Euclidean plane, an integer $k > 1$, and a value L, is it decidable if there exists a patrol schedule for the k robots such that the maximum latency is bounded by L? As already mentioned, a corollary of our results in Section 3 is that for $k = 2$ there exists an optimal cyclic solution and thus for $k = 2$ the answer to the above question is yes.

A severe challenge is that, since patrol schedules are infinite sequences, it is difficult to rule out chaotic solutions where the robots visit the sites in a way that avoids any sort of repeated pattern. Indeed, optimal solutions can behave so chaotically that they require an infinite sequence of bits to describe. For instance, consider the left situation in Figure 1, where we have three robots and four points A, B, C, D that are the vertices of a thin rectangle. To obtain the optimal latency, it suffices that r_1 moves back and forth between A and B, and r_2 moves back and forth between C and D. Since r_3 cannot be used to decrease the

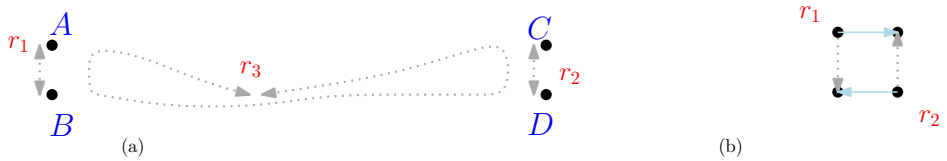


Figure 1 (a) Four points A, B, C, and D form a short and wide rectangle. Robots r_1, r_2, and r_3 can have infinite "unpredictable" optimal patrol schedules. (b) Robots r_1 and r_2 can move to the other diagonal in two different ways.

latency—it will take r_3 too much time to go from A, B to C, D—it can behave as chaotically as it wants, thus causing the description of the patrol schedule to be arbitrarily complicated. This is even possible using only two robots: Consider four sites that form a unit square and two robots placed on opposite corners of the square; see the right situation in Figure 1. An optimal schedule is then an infinite sequence of steps, where in each step both robots move counterclockwise or both move clockwise. Such a schedule need not be cyclic and, hence, may require an infinite sequence of bits to describe. Of course, in both cases we know optimal cyclic solutions exist, and such solutions can be described using finitely many bits. We conjecture that this should be true in general:

▶ **Conjecture 1.** *For the k-robot patrolling problem with min-max latency, there is a cyclic solution that is optimal.*

It is easy to see that there exists a cyclic solution that is a 2-approximation: take an optimal schedule with latency L, and at time L move the robots back to their respective starting positions at time 0, and repeat. The challenge lies in getting an approximation factor smaller than 2, which we achieve in Section 3 where we show that there is a cyclic solution that is a $2(1 - 1/k)$ approximation.

3 Turning an Optimal Solution into a Cyclic Solution

The main goal of this section is to prove the following theorem.

▶ **Theorem 2.** *Let L be the latency in an optimal solution to the k-robot patrol-scheduling problem in a metric space (P, d). There is a cyclic solution with latency at most $2(1 - 1/k)L$.*

We prove the theorem by considering an optimal (potentially "chaotic") solution and turning it into a cyclic solution. This is done by first identifying a certain set of "bottleneck" sites within a time interval of length L, then cutting the schedules into smaller pieces, and then gluing them together to obtain the final cyclic solution. This will require some graph-theoretic tools (Appendix A.1) as well as several new ideas (Appendices A.2–A.4).

Below we sketch the main ideas of the proof; the full proof can be found in Appendix A. We also require some graph theoretic arguments that are presented in Appendix A.1.

3.1 Bidirectional sweep to find "bottleneck" sites.

Consider an optimal patrol schedule with latency L, and consider a time interval $\mathcal{I} := [t_0, t_0 + L]$ for an arbitrary $t_0 > 2L$. By our assumptions, every site is visited at least once within this time interval. We assign a time interval $I_i \subseteq \mathcal{I}$ to every robot r_i. Initially $I_i = \mathcal{I}$. To identify the important sites that are visited by the robots, using a process that we will describe shortly, we will *shrink* each I_i. Shrinking is done by moving the left and right

endpoints of I_i "inward" at the same speed. This will be done in multiple stages and at the end of each stage, an endpoint of some intervals could become *fixed*; a fixed endpoint does not move anymore during the following stages. When both endpoints of I_i are fixed, we have found the final shrunken interval for r_i. Initially, all the endpoints are *unfixed*. For an interval $I_i = [t_i, t'_i]$, shrinking I_i by some value $\varepsilon \geqslant 0$ yields the interval $[t_i + \varepsilon\varphi_1, t'_i - \varepsilon\varphi_2]$ where φ_1 (resp. φ_2) is 1 if the left (resp. right) endpoint of I_i is unfixed, otherwise it is 0. Note that an interval $[x, y]$ with $x > y$ is considered *empty* (i.e., an empty set) and thus shrinking an interval by a large enough value will yield an empty interval (assuming at least one endpoint is unfixed).

The invariant. We maintain the invariant that at the beginning of each stage of the shrinking process, all the sites are visited during the shrunken intervals, i.e., for every site $s \in P$, there exists a robot r_i, and a value $t \in I_i$ such that $f_i(t) = s$. Observe that the invariant holds at the beginning of the first stage of the shrinking process.

The shrinking process. Consider the j-th stage of the shrinking process. Let $\varepsilon_j \geqslant 0$ be the largest (supremum) number such that shrinking all the intervals by ε_j respects the invariant. If ε_j is unbounded, then this is the last stage; every interval I_i that has an unfixed endpoint is reduced to an *empty interval* and we are done with shrinking, meaning, the shrinking process has yielded some $k' \leqslant k$ intervals with both endpoints fixed, and $k - k'$ empty intervals. Otherwise, ε_j is bounded and well-defined as the invariant holds for $\varepsilon_j = 0$. With a slight abuse of the notation, let $I_1, \cdots, I_k$ be the intervals shrunken by ε_j. See Figure 2(left).

Since ε_j is the largest value that respects our invariant, it follows that there must be at least one interval I_{i_j} and at least one of its endpoints t_j such that at time t_j, the robot r_{i_j} visited the site $f_{i_j}(t_j)$ and this site is not visited by any other robot in the interior of their time intervals. Now this endpoint of I_{i_j} is marked as fixed and we continue to the next stage.

For a fixed endpoint A, let $\ell(A)$ be the distance of A to the corresponding boundary of the unshrunk interval. More precisely, if A is a left endpoint then the position of A on the time axis is $t_0 + \ell(A)$, and if A is a right endpoint then this position is $t_0 + L - \ell(A)$. With our notation, if A was discovered at stage j, then $\ell(A) = \varepsilon_1 + \cdots + \varepsilon_j$.

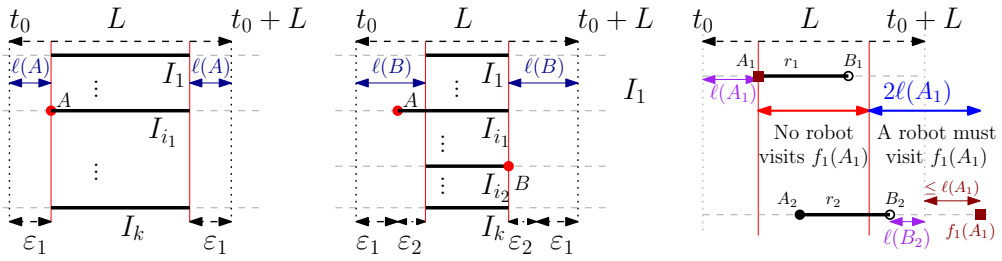


Figure 2 (left) A is fixed at stage 1. (middle) B is fixed at stage 2. (right) By the property of the shrinking process, the site visited at A_1 is not visited by any robot within the red time interval but since the site has latency at most L, it must be visited by some robot in the blue interval.

3.2 Patrol graph, shortcut graph, and bag graph

Shortcutting idea. Figure 2 (right) explains the crucial property of our shrinking process: Robot r_1 visits the site $p = f_1(A_1)$ at time A_1 (which corresponds to the left endpoint of the interval I_1) but to keep the latency of p at most L, p must be visited by another robot, say r_2, sometime in the interval $[t_0 + L - \ell(A_1), t_0 + L + \ell(A_1)]$, shown in blue in the figure. For

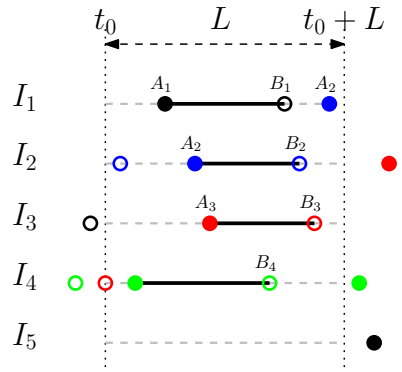
the moment, assume the right endpoint of the interval of r_2 is a fixed point B_2 and r_2 visits a site $p' = f_2(B_2)$ at time B_2. This implies that the distance between p and p' is at most $\ell(A_1) + \ell(B_2)$. Now observe that we can view this as a "shortcut" between endpoints p and p': for example, r_2 can follow its own route from A_2 to B_2, then take the shortcut to A_1, and then follow r_1's route to B_1. The extra cost of taking the shortcut, which is $\ell(A_1) + \ell(B_2)$, can also be charged to the two "shrunken" pieces of the two intervals (the purple intervals in the picture). Our main challenge is to show that these shortcuts can be used to create a cyclic solution with only a small increase in the latency.

To do that, we will define a number of graphs associated with the shrunken intervals. We define a *patrol graph* $\mathcal{P}$, a *bag graph* $\mathcal{B}$ and a *shortcut graph* $\mathcal{S}$. The first two are multigraphs, whereas the shortcut graph is a simple graph. For examples see Figure 3 on page 8 and its discussion on page 7.

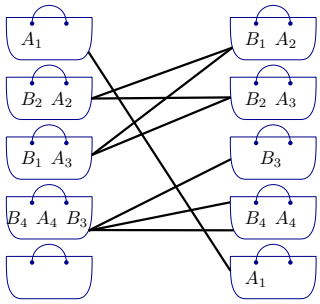
We start with the bag graph and the shortcut graph. We first shrink the intervals as described previously. To define these graphs, consider $2k$ conceptual *bags*, two for each interval (including the empty intervals). More precisely, for each interval we have one *left bag* and one *right bag*. The bags are the vertices of the bag graph $\mathcal{B}$ (Figure 3(b)). The vertices of the shortcut graph $\mathcal{S}$ are the endpoints of the non-empty intervals. To define the edges of the two graphs, we use *placements*. We will present the details below but basically, every endpoint of a non-empty interval will be placed in two bags, one in some left bag β_1 and another time in some right bag β_2. After this placement, we add an edge in the bag graph between β_1 and β_2. Once all the endpoints have been placed, we add edges in the shortcut graph between every two endpoints that have been placed in the same bag (Figure 3(c)).

An example of a bag and shortcut graphs. An example is shown in Figure 3. In part (a), we have four non-empty intervals $I_1 = [A_1, B_1], I_2 = [A_2, B_2], I_3 = [A_3, B_3], I_4 = [A_4, B_4]$ and an empty interval I_5 (we will later explain the second appearance of each endpoint in this picture and for now the reader can ignore the "floating" endpoints). An example of a bag graph is shown in Figure 3(b): Every endpoint is placed twice (once in some left bag and one is some right bag). E.g., A_1 is placed in the top-left bag and the bottom-right bag and thus the two bags are connected in the bag graph. Similarly, B_1 is placed in two bags, once at the top-right bag and the other time at the mid-left bag. In part (c) of the figure, one can see the shortcut graph in which two endpoints are connected if and only if they are placed in the same bag. Also, this is a simple graph and despite the fact that A_4 and B_4 are placed together in two different bags, they are still connected once in the shortcut graph.

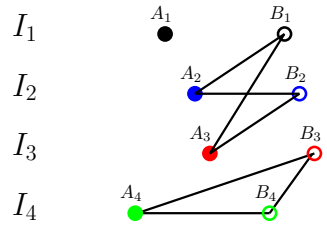
Initially, all the bags are empty. For every non-empty interval $I_1 = [A_1, B_1]$, we place the left endpoint of I_1 in its own left bag and the right endpoint of I_1 in its own right bag. This is the first placement. For the second placement, consider a non-empty interval I_{i_1} and its left endpoint A. The position of A on the time interval is $t = t_0 + \ell(A)$. See Figure 2(right). By our assumptions, the robot r_{i_1} visits the site $p = f_{i_1}(t)$ at time t. Consider the stage of our shrinking process when A gets fixed. For this to happen, the site p cannot be visited by any robot in the time interval $(t_0 + \ell(A), t_0 + L - \ell(A))$ (the red interval in Figure 2(right)), as otherwise, we could either shrink all the intervals by an infinitesimal additional amount or some other endpoint would have been fixed. On the other hand, this site has latency at most L, so it must be visited by another robot in the time interval $(t, t + L] = (t_0 + \ell(A), t_0 + \ell(A) + L]$. This means that the robot r_j that visits p earliest in this interval must do so within the time interval $[t_0 + L - \ell(A), t_0 + L + \ell(A)]$ (the blue interval in Figure 2(right)). Note that r_j could be any of the robots, including r_{i_1} itself. We now place A in the right bag of I_j. A very similar strategy is applied to the right end



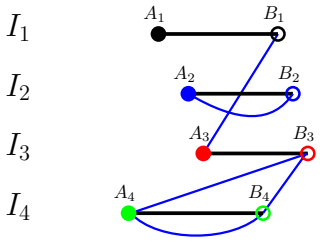
(a) After the shrinking process



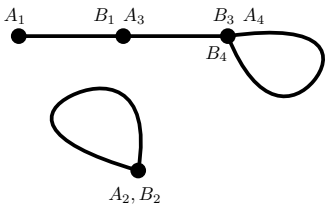
(b) The bag graph



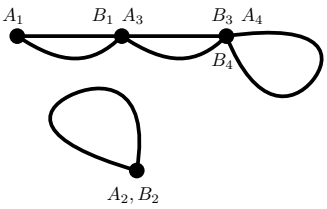
(c) The shortcut graph



(d) The patrol graph



(e) After contracting the blue edges



(f) Eulerized

Figure 3 Example of bag, shortcut and patrol graph

point of I_1; for details see Section A where we also prove the following properties.

▶ **Lemma 3.** *The bag graph $\mathcal{B}$ and the shortcut graph $\mathcal{S}$ have the following properties.*

(a) *$\mathcal{B}$ is a bipartite graph.*

(b) *$\mathcal{S}$ is isomorphic to the line graph of $\mathcal{B}$*

(c) *Let $\mathcal{B}'$ be a connected component of $\mathcal{B}$. If none of the vertices of $\mathcal{B}'$ belong to empty-intervals, then the number of vertices of $\mathcal{B}'$ is equal to its number of edges.*

(d) *Let $\mathcal{S}'$ be a connected component of $\mathcal{S}$. If $\mathcal{S}'$ has a vertex v of degree one, then it must be the case that v corresponds to an endpoint of a non-empty interval that has been placed (alone) in a bag of an empty interval.*

The patrol graph. The above lemma, combined with the graph theoretical tools that we outline in Appendix A.1 allows us to define the patrol graph $\mathcal{P}$. Here, we only give an outline, and for the full details see Appendix A. An example of a patrol graph is shown in Figure 3(d). Initially, the patrol graph, $\mathcal{P}$, consists of k' isolated black edges, one for each non-empty interval. Observe that both $\mathcal{P}$ and the shortcut graph $\mathcal{S}$ have the same vertex set (endpoints of the non-empty intervals). We add a subset of the edges of the shortcut graph to $\mathcal{P}$. Let us consider an "easy" case to illustrate the main idea.

An easy case. Assume $\mathcal{B}$ is connected and that it has an even number of edges. In this case, we can in fact prove that an optimal cyclic solution exists. Recall that $\mathcal{S}$ is the line graph of $\mathcal{B}$ and it is known (see Theorem 13) that the line graph of a connected graph with even number of edges, has a perfect matching. Thus, we can find a perfect matching M as a subset of edges of $\mathcal{S}$. Add M to $\mathcal{P}$ as "blue" edges. Now, every vertex of $\mathcal{P}$ is adjacent to a blue and a black edge and thus $\mathcal{P}$ decomposes into a set of "bichromatic" cycles, i.e., cycles with alternating black-blue edges. With a careful accounting argument, we can show that this indeed yields a cyclic solution without increasing the latency of any of the sites. We have already mentioned the main idea under the "shortcutting idea" paragraph, at the beginning of the section. Specifically, we will use the following lemma.

▶ **Lemma 4.** *Consider two adjacent vertices v and w in the shortcut graph. This means that there are two non-empty intervals I_1 and I_2 such that v corresponds to an endpoint A of I_1 and w corresponds to an end point B of I_2 and A and B are placed in the same bag. Let s_1 be the site visited at A during I_1 and s_2 be the site visited at B on I_2. Then, we have $d(s_1, s_2) \leqslant \ell(A) + \ell(B)$.*

Black edges represent the routes of the robots, and blue edges are the shortcuts that connect one route to another. So in this easy case, once the patrol graph has decomposed into bichromatic cycles, we turn each cycle into one closed route (i.e., cycle) using the shortcuts. All the robots that correspond to the black edges are placed evenly on this cycle. Since by our invariant all the sites are visited at some time on the black edges, it follows that the robots visit all the sites. A careful accounting argument using the "missing" pieces $\ell(\cdot)$, then shows that the latency does not increase at all.

Unfortunately $\mathcal{B}$ can have connected components with odd number of edges. Nonetheless, in all cases we can build a particular patrol graph, $\mathcal{P}$, with the following properties.

▶ **Lemma 5.** *The patrol graph $\mathcal{P}$ consists of k' pairwise non-adjacent black edges and a number of blue edges. Any blue edge (v, w) in $\mathcal{P}$ corresponds to an edge in the shortcut graph $\mathcal{S}$. Furthermore, the set of blue edges can be decomposed into a matching and a number of triangles. In addition, any vertex of $\mathcal{P}$ that is not adjacent to a blue edge can be charged to a bag of an empty interval.*

The idea covered in the above "easy case" works because it covers the black edges of $\mathcal{P}$ with bichromatic edge-disjoint cycles and each cycle becomes a cyclic route. Unfortunately, in general $\mathcal{P}$ might not have the structure that would allow us to do this. Here, we only outline the steps we need to overcome this: we consider each connected component, $\mathcal{P}_i$, of $\mathcal{P}$. We first contract blue edges of $\mathcal{P}_i$ to obtain a contracted patrol graph, $\mathcal{P}_i^c$, (Figure 3(e)) then we Eulerize it (Figure 3(f)), meaning, we duplicate a number of black edges such that the resulting graph is Eulerian.

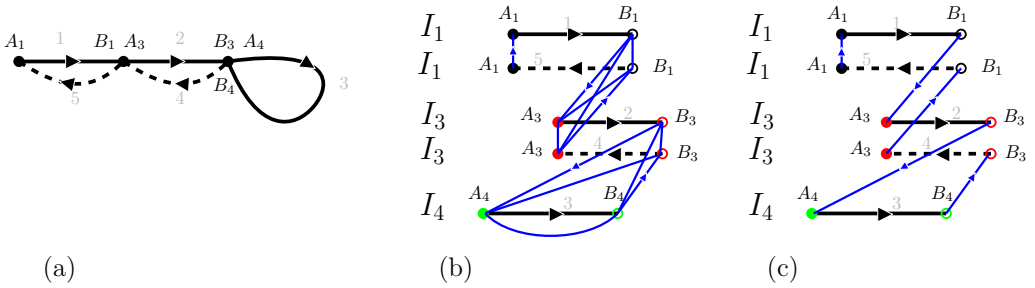


Figure 4 (a) An Eularized contracted patrol graph (ECPG). Duplicated edges are drawn with dashed lines. The edges are directed and numbered according to an Euler tour. (b) The same ECPG after "uncontracting" the blue edges. The duplicated routes (i.e., the routes that will be traversed twice) are shown with dashed lines. The corresponding Euler tour is marked and numbered. The shortcuts are taken in the correct direction. (c) The bichromatic cycle gives us a cyclic route. It shows how the robots can travel along it.

This yields us an Eularized contracted patrol graph, $\mathcal{P}_i^{Ec}$ (Figure 4(a)). Next, we put the contracted blue edges back in $\mathcal{P}_i^{Ec}$ which gives us the final patrol graph (Figure 4(b)). In this final graph, we can show that we can cover the black edges using bichromatic edge disjoint cycles where each connected component of the final graph turns into one cycle (Figure 4(c)); this yields us a cyclic solution. However, the duplicated black edges represent routes of robots that need to be traversed twice to obtain the cyclic solution. This leads us to the final challenge: how to allocate the robots to the resulting cycles to minimize the latency. With some careful accounting and considering a few cases, we can show that this can be done in such a way that the resulting cyclic solution has latency at most $2L(1 - 1/k)$. We do this in Appendix A, proving Theorem 2.

4 Cyclic Solutions

In this section we show how to approximate an optimal cyclic solution to the patrol scheduling problem for k robots in a metric space (P, d). We start with some notation and basic observations.

For a subset $Q \subseteq P$, let $\mathrm{TSP}(Q)$ denote an optimal TSP tour of Q and let $tsp(Q)$ denote its total length. Let $\mathrm{MST}(Q)$ denote a minimum spanning tree of Q. Now consider a partition $\Pi = \{P_1, \ldots, P_t\}$ of P, where each subset P_i is assigned k_i robots such that $\sum_{i=1}^{t} k_i = k$. A *cyclic solution* for this partition and distribution of robots is defined as follows. For each P_i there is a cycle C_i such that the k_i robots assigned to P_i start evenly spaced along C_i and then traverse C_i at maximum speed in the same direction. Hence, the latency L of such a cyclic solution satisfies $L \geqslant \max_i(tsp(P_i)/k_i)$, with equality if $C_i = \mathrm{TSP}(P_i)$ for all i.

To prove the main theorem of this section we need several helper lemmas. Let $\Pi = (P_1, \ldots, P_t)$ be a partition of P and let $E \subseteq P \times P$ be a set of edges. The *coarsening of* Π

with respect to E is the partition Π' of P given by the connected components of the graph $(\bigcup_i \mathrm{MST}(P_i)) \cup E$.

▶ **Lemma 6.** *Let S be a cyclic solution with partition $\Pi = (P_1, \ldots, P_t)$ and latency L. Let $\Pi' = (P'_1, \ldots, P'_{t'})$ be the coarsening of Π with respect to an edge set E of total length ℓ. Then there is a cyclic solution S' with partition Π' and latency L' such that $L' \leqslant L + \ell$.*

Proof. Let $C_1, \ldots, C_t$ be the cycles used in S. Consider a subset $P'_i \in \Pi'$, and assume without loss of generality that P'_i is the union of the subsets $P_1, \ldots, P_s$ from Π. Then there is a set $E_i \subseteq E$ of $k-1$ edges such that $\left(\bigcup_{j=1}^{s} C_j\right) \cup E_i$ is connected. Moreover, there is a cycle C'_i covering all sites in P'_i traversing the edges of each C_j once and the edges of E_i twice. Hence,

$$\|C'_i\| = \sum_{j=1}^{s} \|C_j\| + 2 \cdot \|E_i\|,$$

where $\|\cdot\|$ denotes the total length of a set of edges. Since the latency in S is L, we know that $\|C_j\| \leqslant k_j L$. Hence, using $\sum_{j=1}^{s} k_j \geqslant 2$ robots for the cycle C'_i, the latency for the sites in P'_i is at most

$$\frac{\|C'_i\|}{\sum_{j=1}^{s} k_j} = \frac{\sum_{j=1}^{s} \|C_j\| + 2 \cdot \|E_i\|}{\sum_{j=1}^{s} k_j} \leqslant \frac{\sum_{j=1}^{s} k_j L + 2 \cdot \|E_i\|}{\sum_{j=1}^{s} k_j} \leqslant L + \|E_i\| \leqslant L + \ell.$$

Thus the latency for any subset $P'_i \in \Pi'$ is at most $L + \ell$. ◀

▶ **Lemma 7.** *Let L^* be the latency of an optimal cyclic solution. For any $\varepsilon > 0$, there exists a cyclic solution with partition $\Pi = (P_1, \ldots, P_t)$ and latency $L < (1+\varepsilon)L^*$ such that for any pair $i \neq j$ we have $d(P_i, P_j) > \varepsilon \cdot L^*/k$, where $d(P_i, P_j) := \min\{d(x, y) : x \in P_i \text{ and } y \in P_j\}$.*

Proof. Let S^* be an optimal solution with partition $\Pi^* = (P^*_1, \ldots, P^*_q)$, where $q \leqslant k$. Let E_{short} be the set of all edges of the complete graph of the metric space with length at most $\varepsilon L^*/k$. Let $\Pi = (P_1, \ldots, P_t)$ be the partition obtained by coarsening Π^* with respect to E_{short}, and let $E^* \subseteq E_{\text{short}}$ be a minimal subset such that coarsening Π^* with E^* gives the same partition Π. Observe that as $q \leqslant k$, we have $|E^*| \leqslant k-1$. Lemma 6 implies that there is a cyclic solution S with partition Π and latency at most

$$L^* + |E^*| \cdot (\varepsilon L^*/k) < (1+\varepsilon)L^*.$$

Moreover, since Π is a coarsening of Π^* with respect to E_{short}, the pairwise distance between any two sets of Π is larger than $\varepsilon L^*/k$. ◀

▶ **Lemma 8.** *Suppose there is a cyclic solution of latency L for a given metric space (P, d) and k robots. Then $\mathrm{MST}(P)$ has fewer than $k(1+1/\alpha)$ edges of length more than αL, for any $0 < \alpha \leqslant 1$.*

Proof. Let $C_1, \ldots, C_q$ be the cycles in the given cyclic solution of latency L, let k_i denote the number of robots assigned to C_i, and let $P_i \subset P$ be the sites in C_i. Let E be a subset of $q-1 \leqslant k-1$ edges from $\mathrm{MST}(P)$ such that $(\bigcup_{i=1}^{q} C_i) \cup E$ is connected. Then

$$\sum_{i=1}^{q} \|C_i\| > \|\mathrm{MST}(P)\| - \|E\| = \|MST(P) \setminus E\|$$

Since $\|C_i\| \leqslant k_i L$, we have $\sum_{i=1}^{q} \|C_i\| \leqslant kL$. Hence, $\|\mathrm{MST}(P) \setminus E\| < kL$, which implies that $\mathrm{MST}(P) \setminus E$ contains less than k/α edges of length more than αL. Including the edges in E, we thus know that $\mathrm{MST}(P)$ has less than $k(1+1/\alpha)$ edges of length more than αL. ◀

I Theorem 9. Suppose we have a-approximation algorithm for TSP in a metric space $(P; d)$, with running time (n), and an algorithm for computing an MST that runs in time $T^0(n)$. Then there is a $(1 + ")$ -approximation algorithm for nding a minimum-latency cyclic patrol schedule withk robots that runs in $T^0(n) + (O(k="))^k$ (n) time.

Proof. Let L be the latency in an optimal cyclic solution. By Lemma 7 there is a solution$S$ with latency $(1 + ")L$ and partition $= f P_1; : : : ; P_t g$ such that $d(P_i ; P_j) > "L =k$ for all $i 6 j$. Let E be the set of edges of$MST(P)$ with length more than $"L =k$, and let $T_1; : : : ; T_z$ be the forest obtained from$MST(P)$ by removing E. Let $V(T_j)$ denote the sites inT_j. For any j we have$V(T_j)$ P_i for some$i$. Otherwise, there would exist two sites$p; q 2 V(T_j)$ that are neighbors in T_j but stay in di erent sets in . This would lead to a contradiction: the former implies $d(p; q) 6 "L =k$ while the later implies $d(p; q) > "L =k$. Thus is a coarsening off $V(T_1); : : : ; V(T_z)g$ with respect to some subset of$E$.

By Lemma 8, the number of edges of$MST(P)$ longer than $"L =k$ is at most $k 1 + \frac{k}{"}$. That is, the heaviest $k 1 + \frac{k}{"}$ edges of$MST(P)$ are a superset of the set$E$ from above. Thus we can nd the partition from above by rst computing $MST(P)$, removing the heaviest $k 1 + \frac{k}{"}$ edges, and then trying all coarsenings determined by subsets of the removed edges. Given a -approximation for TSP, below we argue how to get a -approximation to the optimal cyclic solution for a given partition . Running this subroutine for each of the above determined partitions and taking the best solution found will thus give latency at most $(1 + ") L$.

Observe that the optimal cyclic solution on a given partition $= f P_1; : : : ; P_t g$ uses cycles determined by$TSP(P_i)$, and choosesk_i, the number of robots assigned toP_i, so as to minimize $\max_i tsp(P_i)=k_i$. Thus we can compute a -approximation of the optimal solution on by rst computing a -approximation to $TSP(P_i)$ for all i, where$\overline{tsp}(P_i)$ denotes its corresponding value, and then selecting$k_1^0; : : : ; k_t^0$ so as to minimize$\max_i \overline{tsp}(P_i)=k_i^0$. The latter step of determining the k_i^0 can be done in$O(k \log k)$ time by initially assigning one robot to each P_i, and then iteratively assigning each next robot to whichever set of the partition currently has the largest ratio. The latency of the solution we nd for is thus

$$\max_i \frac{\overline{tsp}(P_i)}{k_i^0} \quad 6 \max_i \frac{\overline{tsp}(P_i)}{k_i} \quad 6 \max_i \frac{tsp(P_i)}{k_i} \quad = \quad [\text{optimal cyclic latency for }];$$

where the last inequality follows from the fact that $\overline{tsp}(P_i) 6$ $tsp(P_i)$ for all i.

It remains to bound the running time. For each partition we approximate $TSP(P_i)$ for all i, and then run an $O(k \log k)$ time algorithm to determine the robot assignment. Thus the time per partition is bounded by (n), where$n$ is the total number of sites. Here we assume that $(n) = (n \log n)$ and that (n) upper bounds the time for the initial $MST(P)$ computation.

The number of partitions we consider is determined by the number of subsets of size at most k of the longest$k(1 + k=")$ edges of$MST(P)$, which is bounded by

$$\binom{k(1 + k=")}{k} \quad 2^k;$$

as the rst term bounds the number of subsets of size exactly$k$, and for each subset the second term accounts for the number of ways in which we can pick at most$k$ edges from that subset. We have the following standard upper bound on binomial coe cients.

$$\binom{N}{K} \quad 6 \quad \left(\frac{N \ e}{K}\right)^K :$$

Therefore, the total number of partitions we consider is at most

$$\binom{k(1+k/\varepsilon)}{k} \cdot 2^k \leqslant \left(\frac{k(1+k/\varepsilon)\cdot e}{k}\right)^k \cdot 2^k = (2e(1+k/\varepsilon))^k = (O(k/\varepsilon))^k .$$

Thus the total running time is $(O(k/\varepsilon))^k \cdot \tau_\gamma(n)$ as claimed. ◀

Recently, Karlin *et al.* [10] presented a $(3/2-\delta)$-approximation algorithm for metric TSP, where $\delta > 10^{-36}$ is a constant, thus slightly improving the classic $(3/2)$-approximation by Christofides [5]. Furthermore TSP in $\mathbb{R}^d$ admits a PTAS [4, 15]. Thus we have the following.

▶ **Corollary 10.** *For any fixed k, there is polynomial-time $(3/2)$-approximation algorithm for finding a minimum-latency cyclic patrol schedule with k robots in arbitrary metric spaces, and there is a PTAS in $\mathbb{R}^d$ for any fixed constant d.*

Theorem 2 in Section 3 and Corollary 10 together imply the following.

▶ **Theorem 11.** *For any fixed k and $\varepsilon > 0$, there is a polynomial-time $(3(1-1/k)+\varepsilon)$-approximation algorithm for the k-robot patrol-scheduling problem in arbitrary metric spaces, and a polynomial-time $(2(1-1/k)+\varepsilon)$-approximation algorithm in $\mathbb{R}^d$ (for fixed d).*

5 Conclusion and Future Work

This is the first paper that presents rigorous analysis and approximation algorithms for multi-robot patrol scheduling problem in general metric spaces. There are several challenging open problems. The first and foremost is to prove or disprove the conjecture that there is always a cyclic solution that is optimal overall. Proving this conjecture will immediately provide a PTAS for the Euclidean multi-robot patrol-scheduling problem. It would also imply that the decision problem is decidable. Another direction for future research is to extend the results to the weighted setting. As has been shown for the 1-dimensional problem [1], the weighted setting is considerably harder.

References

1. Peyman Afshani, Mark de Berg, Kevin Buchin, Jie Gao, Maarten Löffler, Amir Nayyeri, Benjamin Raichel, Rik Sarkar, Haotian Wang, and Hao-Tsung Yang. Approximation algorithms for multi-robot patrol-scheduling with min-max latency. In *Algorithmic Foundations of Robotics XIV*, pages 107–123, 2021.
2. Soroush Alamdari, Elaheh Fata, and Stephen L Smith. Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research*, 33(1):138–154, 2014.
3. Esther M Arkin, Refael Hassin, and Asaf Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.
4. Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.
5. Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon Univ., Pittsburgh, 1976.
6. George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.
7. Yehuda Elmaliach, Asaf Shiloni, and Gal A. Kaminka. A realistic model of frequency-based multi-robot polyline patrolling. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, pages 63–70, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

8 Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

9 L. Iocchi, L. Marchetti, and D. Nardi. Multi-robot patrolling with coordinated behaviours in realistic environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2796–2801, Sept 2011. doi:10.1109/IROS.2011.6094844.

10 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, page 32–45, 2021.

11 M Yu Khachai and ED Neznakhina. A polynomial-time approximation scheme for the Euclidean problem on a cycle cover of a graph. *Proceedings of the Steklov Institute of Mathematics*, 289(1):111–125, 2015.

12 Michael Khachay and Katherine Neznakhina. Polynomial time approximation scheme for the minimum-weight k-size cycle cover problem in Euclidean space of an arbitrary fixed dimension. *IFAC-PapersOnLine*, 49(12):6–10, 2016.

13 M Reza Khani and Mohammad R Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69(2):443–460, 2014.

14 Kin Sum Liu, Tyler Mayer, Hao-Tsung Yang, Esther Arkin, Jie Gao, Mayank Goswami, Matthew P. Johnson, Nirman Kumar, and Shan Lin. Joint sensing duty cycle scheduling for heterogeneous coverage guarantee. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

15 Joseph SB Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on computing*, 28(4):1298–1309, 1999.

16 Christos H Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical computer science*, 4(3):237–244, 1977.

17 D. Portugal and R. P. Rocha. On the performance and scalability of multi-robot patrolling algorithms. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 50–55, Nov 2011. doi:10.1109/SSRR.2011.6106761.

18 E. Stump and N. Michael. Multi-robot persistent surveillance planning as a vehicle routing problem. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 569–575, Aug 2011. doi:10.1109/CASE.2011.6042503.

19 David P. Sumner. Graphs with 1-factors. *Proc. Amer. Math. Soc.*, 42(1):8–12, 1974.

20 Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

21 Wenzheng Xu, Weifa Liang, and Xiaola Lin. Approximation algorithms for min-max cycle cover problems. *IEEE Transactions on Computers*, 64(3):600–613, 2013.

22 Hao-Tsung Yang, Shih-Yu Tsai, Kin Sum Liu, Shan Lin, and Jie Gao. Patrol scheduling against adversaries with varying attack durations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1179–1188, 2019.

A Turning an Optimal Solution into a Cyclic Solution

The main goal of this section is to prove the following theorem.

▶ **Theorem 2.** *Let L be the latency in an optimal solution to the k-robot patrol-scheduling problem in a metric space (P, d). There is a cyclic solution with latency at most $2(1-1/k)L$.*

We prove the theorem by considering an optimal (potentially "chaotic") solution and turning it into a cyclic solution. This is done by first identifying a certain set of "bottleneck" sites within a time interval of length L, then cutting the schedules into smaller pieces, and then gluing them together to obtain the final cyclic solution. This will require some graph-theoretic tools (Appendix A.1) as well as several new ideas (Appendices A.2–A.4).

A.1 Graph Theoretic Preliminaries

Here, we use the term graph to include multigraphs (i.e., those with multiple edges between the same vertices and with loops). A simple graph is one that does not have multiple edges or loops. A connected graph is Eulerian if all of its vertices have even degree. One of the earliest results in graph theory is that an Eulerian graph has an Euler tour, i.e., a closed walk that visits every edge exactly once. Given a graph G, Eulerizing G is the problem of duplicating the minimum number of edges of G until the resulting graph is Eulerian. We will use the following lemma.

▶ **Lemma 12.** *Let G be a connected graph with E edges.*

1. *G can always be Eulerized by duplicating all E edges.*
2. *If G has exactly one vertex of degree 1, then we can Eulerize G by duplicating at most $E-1$ edges.*
3. *If G has no vertex of degree 1, then we can Eulerize G by duplicating at most $E-2$ edges.*

Proof. The first claim is trivial. For the other two parts, let T be a spanning tree of G. To make G Eulerian, we need to make all the degrees even. Let O be the number of vertices with odd degree in T. First observe that O is an even number. Pair the vertices of odd degree into $O/2$ pair v_i, u_i, $1 \leqslant i \leqslant O/2$ and consider the path π_i in T that connects v_i to u_i. Consider a pairing that minimizes the total length of the paths π_i.

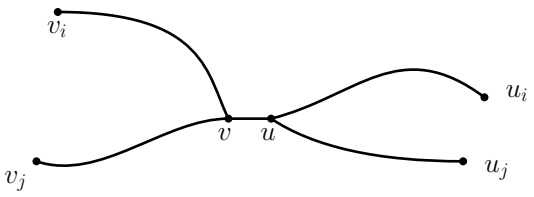


Figure 5 Two paths cannot share an edge, if the total path length is minimized.

Observe that the paths $\pi_1, \cdots, \pi_{O/2}$ will be edge disjoint. To see this, assume π_i (connecting v_i to u_i) and π_j (connecting v_j to u_j) share an edge $e = (v, u)$. W.l.o.g, we can assume that on the path π_i (resp. π_j) v is closer to v_i (resp. v_j) than u. See Figure 5. Let $\pi_i(v_i, v)$ denote the subpath of π_i from v_i to v, and define $\pi_i(u, u_i)$, $\pi_j(v, v_j)$, and $\pi_j(u_j, u)$ similarly. Then the total length of the paths $\pi_i(v_i, v) \circ \pi_j(v, v_j)$ and $\pi_j(u_j, u) \circ \pi_i(u, u_i)$ is one smaller than the total length of π_i and π_j, which contradicts the choice of the pairing. Thus, the pairing that minimizes the total length of the paths consists of edge-disjoint paths.

After considering such a pairing, we simply duplicate all the edges on the paths π_i. Note that each edge in tree T is duplicated at most once. Now, the second claim in the lemma easily follows: as G has exactly one vertex of degree 1, it is not a tree which means T has at most $E-1$ edges and thus we duplicate at most $E-1$ edges to Eulerize G.

For the third claim, we need some further case analysis. Observe that if T has at most $E-2$ edges, then we are done. Thus, assume T has $E-1$ edges, so G is the union of T and one edge e. But by our assumptions, G does not have a degree 1 vertex which means T has at most two leaves; but this means that T is a tree with at most 2 leaves and thus T can only be a path with $E-1$ edges and furthermore, both of these leaves must be connected by e in G. Consequently, this implies that G is a cycle but in that case, G is already Eulerian and thus nothing needs to be duplicated in this case. ◀

A 2-path is the graph of two adjacent edges. A K_3 or a triangle is the clique of three vertices and a claw is the complete bipartite graph $K_{1,3}$. For a graph G, the line graph of G, denoted by $\overline{G}$, is a graph whose vertices are the edges of G and two vertices in $\overline{G}$ are connected if and only if the two corresponding edges in G are adjacent. Observe that a perfect matching M in $\overline{G}$ corresponds to a partition of edges of G into 2-paths and vice versa. The following theorems are known about the line graphs.

▶ **Theorem 13.** *If G is connected and it has an even number of edges, then $\overline{G}$ has a perfect matching M. Consequently, M yields a partition of edges of G into a number of 2-paths.*

However, we need something a bit more general about line graphs.

▶ **Theorem 14.** *Consider a connected graph G with odd number of edges. For any vertex v in G, we can find a partition of edges of G into a number of 2-paths plus an edge adjacent to v.*

Proof. The proof basically follows from Sumner's proof [19]. Pick any vertex v in G. We use induction, meaning, we assume that the claim holds for any connected subgraph of G that includes the vertex v. Let T be the BFS traversal of G with root v. If there is a vertex u such that u is adjacent to two edges e_1 and e_2 with $e_1, e_2 \notin T$, then we can remove e_1 and e_2 as a 2-path and the remaining graph will stay connected (through T). And thus our claim follows by induction. So in the rest of this proof assume that every vertex is adjacent to at most one edge that is not in T.

Let u be a leaf in T that is farthest away from v and let e_1 be the edge in T adjacent to u. We now consider a few cases:

- case (i). Assume u is adjacent to another edge e_2. Here, by the above assumption, u is not adjacent to any other edge (in G). Now, we remove e_1 and e_2 as a 2-path, leaving us with an isolated vertex u while the rest of the graph is still connected (through T) and thus our claim follows by induction. Thus assume, case (i) does not hold, which means u has degree 1 in G. Consider the parent w of u.
- case (ii). Assume w is adjacent to an edge e_2 that is not in T. In this case, we can remove e_1 and e_2 as a 2-path. This leaves u as an isolated vertex and the rest of the graph will still stay connected and thus the claim follows by induction.
- case (iii). Assume w is not adjacent to any edge that is not in T. Let e_1 be the edge that connects w to u. This case has three additional subcases.
  - case (iii)a. Assume u is the only child of w and $w = v$. In this case, we are done, since we have only one edge left which is adjacent to v.

- case (iii)b. Assume u is the only child of w and $w \neq v$. In this case, w must have a parent and thus let e_2 be the edge that connects w to its parent. We can remove e_1 and e_2 as a 2-path. This leaves both w and u as two isolated vertices but the rest of the graph still stays connected and thus our claim follows by induction.
- case (iii)c. Assume w has another child u' and let e_2 be the edge that connects w to u'. Here, we can remove e_1 and e_2 as a 2-path and this leaves both u and u' as two isolated vertices but the rest of the graph still stays connected and thus our claim follows by induction.

◀

We now prove the one graph theoretic lemma that we will later use.

▶ **Lemma 15.** *Let G be a connected graph that contains an even cycle. Let $\overline{G}$ be the line graph of G. Then, either $\overline{G}$ has a perfect matching or its vertices can be partitioned into a matching and a triangle. Consequently, we can partition the edges of G into a number of 2-paths or into a number of 2-paths and a claw.*

Proof. Observe that we only need to consider the case when G has an odd number of edges as the other case is already covered by Theorem 13.

Let C be an even cycle in G, and let $G_1, \cdots, G_m$ be the connected components obtained after removing the edges of C. Since G has an odd number of edges and C has an even number of edges, there is a component G_i with an odd number of edges. Let v be a vertex of G_i that lies on C; such a vertex exists because G is connected, so every component resulting from the removal of the edges of C must contain at least one vertex from C. By Theorem 14, we can partition the edges of G_i into a number of 2-paths and at most one isolated edge e adjacent to v. Let e' and e'' be the edges of C incident to v. Note that $\{e, e', e''\}$ is a claw. Now consider the graph G again (including the cycle C), and remove the claw $\{e, e', e''\}$ and the 2-paths comprising $G_i \setminus \{e\}$ from G. The resulting graph G^* consist of the path $C \setminus \{e', e''\}$ plus various components G_j attached to this path, and so G^* is connected. Moreover, G^* has an even number of edges, since it was obtained by removing an odd number of edges from G. Hence, the edges from G^* can be decomposed into a number of 2-paths by Theorem 13, thus finishing the proof. ◀

A.2 Bidirectional sweep to find "bottleneck" sites.

Consider an optimal patrol schedule with latency L, and consider a time interval $\mathcal{I} := [t_0, t_0 + L]$ for an arbitrary $t_0 > 2L$. By our assumptions, every site is visited at least once within this time interval. We assign a time interval $I_i \subseteq \mathcal{I}$ to every robot r_i. Initially $I_i = \mathcal{I}$. To identify the important sites that are visited by the robots, using a process that we will describe shortly, we will *shrink* each I_i. Shrinking is done by moving the left and right endpoints of I_i "inward" at the same speed. This will be done in multiple stages and at the end of each stage, an endpoint of some intervals could become *fixed*; a fixed endpoint does not move anymore during the following stages. When both endpoints of I_i are fixed, we have found the final shrunken interval for r_i. Initially, all the endpoints are *unfixed*. For an interval $I_i = [t_i, t'_i]$, shrinking I_i by some value $\varepsilon \geqslant 0$ yields the interval $[t_i + \varepsilon\varphi_1, t'_i - \varepsilon\varphi_2]$ where φ_1 (resp. φ_2) is 1 if the left (resp. right) endpoint of I_i is unfixed, otherwise it is 0. Note that an interval $[x, y]$ with $x > y$ is considered *empty* (i.e., an empty set) and thus shrinking an interval by a large enough value will yield an empty interval (assuming at least one endpoint is unfixed).

The invariant. We maintain the invariant that at the beginning of each stage of the shrinking process, all the sites are visited during the shrunken intervals, i.e., for every site $s \in P$, there exists a robot r_i, and a value $t \in I_i$ such that $f_i(t) = s$. Observe that the invariant holds at the beginning of the first stage of the shrinking process.

The shrinking process. Consider the j-th stage of the shrinking process. Let $\varepsilon_j \geqslant 0$ be the largest (supremum) number such that shrinking all the intervals by ε_j respects the invariant. If ε_j is unbounded, then this is the last stage; every interval I_i that has an unfixed endpoint is reduced to an *empty interval* and we are done with shrinking, meaning, the shrinking process has yielded some $k' \leqslant k$ intervals with both endpoints fixed, and $k - k'$ empty intervals. Otherwise, ε_j is bounded and well-defined as the invariant holds for $\varepsilon_j = 0$. With a slight abuse of the notation, let $I_1, \cdots, I_k$ be the intervals shrunken by ε_j. See Figure 6 (left).

Since ε_j is the largest value that respects our invariant, it follows that there must be at least one interval I_{i_j} and at least one of its endpoints t_j such that at time t_j, the robot r_{i_j} visited the site $f_{i_j}(t_j)$ and this site is not visited by any other robot in the interior of their time intervals. Now this endpoint of I_{i_j} is marked as fixed and we continue to the next stage.

For a fixed endpoint A, let $\ell(A)$ be the distance of A to the corresponding boundary of the unshrunk interval. More precisely, if A is a left endpoint then the position of A on the time axis is $t_0 + \ell(A)$, and if A is a right endpoint then this position is $t_0 + L - \ell(A)$. With our notation, if A was discovered at stage j, then $\ell(A) = \varepsilon_1 + \cdots + \varepsilon_j$.

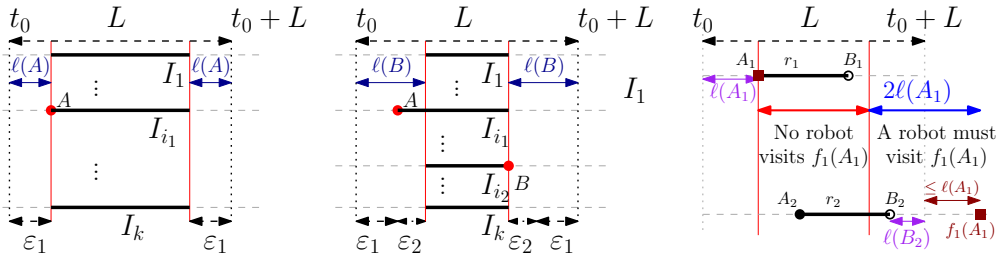


Figure 6 (left) A is fixed at stage 1. (middle) B is fixed at stage 2. (right) By the property of the shrinking process, the site visited at A_1 is not visited by any robot within the red time interval but since the site has latency at most L, it must be visited by some robot in the blue interval.

A.3 Patrol graph, shortcut graph, and bag graph

Shortcutting idea. Figure 6(right) explains the crucial property of our shrinking process: Robot r_1 visits the site $p = f_1(A_1)$ at time A_1 (which corresponds to the left endpoint of the interval I_1) but to keep the latency of p at most L, p must be visited by another robot, say r_2, sometime in the interval $[t_0 + L - \ell(A_1), t_0 + L + \ell(A_1)]$, shown in blue in the figure. For the moment, assume the right endpoint of the interval of r_2 is a fixed point B_2 and r_2 visits a site $p' = f_2(B_2)$ at time B_2. This implies that the distance between p and p' is at most $\ell(A_1) + \ell(B_2)$. Now observe that we can view this as a "shortcut" between endpoints p and p': for example, r_2 can follow its own route from A_2 to B_2, then take the shortcut to A_1, and then follow r_1's route to B_1. The extra cost of taking the shortcut, which is $\ell(A_1) + \ell(B_2)$, can also be charged to the two "shrunken" pieces of the two intervals (the purple intervals in the picture). Our main challenge is to show that these shortcuts can be used to create a cyclic solution with only a small increase in the latency.

To do that, we will define a number of graphs associated with the shrunken intervals. We define a *patrol graph* $\mathcal{P}$, a *bag graph* $\mathcal{B}$ and a *shortcut graph* $\mathcal{S}$. The first two are multigraphs,

whereas the shortcut graph is a simple graph. We will describe particular examples of these graphs below.

We start with the bag graph and the shortcut graph. We first shrink the intervals as described previously. To define these graphs, consider $2k$ conceptual *bags*, two for each interval (including the empty intervals). More precisely, for each interval we have one *left bag* and one *right bag*. The bags are the vertices of the bag graph $\mathcal{B}$ (Figure 3(b)). The vertices of the shortcut graph $\mathcal{S}$ are the endpoints of the non-empty intervals. To define the edges of the two graphs, we use *placements*. We will present the details below but basically, every endpoint of a non-empty interval will be placed in two bags, one in some left bag β_1 and another time in some right bag β_2. After this placement, we add an edge in the bag graph between β_1 and β_2. Once all the endpoints have been placed, we add edges in the shortcut graph between every two endpoints that have been placed in the same bag (Figure 3(c)).

Initially, all the bags are empty. For every non-empty interval $I_1 = [A_1, B_1]$, we place the left endpoint of I_1 in its own left bag and the right endpoint of I_1 in its own right bag. This is the first placement. For the second placement, consider a non-empty interval I_{i_1} and its left endpoint A. The position of A on the time interval is $t = t_0 + \ell(A)$. See Figure 6(right). By our assumptions, the robot r_{i_1} visits the site $p = f_{i_1}(t)$ at time t. Consider the stage of our shrinking process when A gets fixed. For this to happen, the site p cannot be visited by any robot in the time interval $(t_0 + \ell(A), t_0 + L - \ell(A))$ (the red interval in Figure 6(right)), as otherwise, we could either shrink all the intervals by an infinitesimal additional amount or some other endpoint would have been fixed. On the other hand, this site has latency at most L, so it must be visited by another robot in the time interval $(t, t + L] = (t_0 + \ell(A), t_0 + \ell(A) + L]$. This means that the robot r_j that visits p earliest in this interval must do so within the time interval $[t_0 + L - \ell(A), t_0 + L + \ell(A)]$ (the blue interval in Figure 6(right)). Note that r_j could be any of the robots, including r_{i_1} itself. We now place A in the right bag of I_j.

A very similar strategy is applied to the right endpoint B of I_{i_1}; the location of B on the time axis is $t' = t_0 + L - \ell(B)$, the site $p' = f_{i_1}(t')$ cannot be visited by any robot within the interval $(t_0 + \ell(B), t_0 + L - \ell(B))$ whereas it should be visited by some robot in the region $[t' - L, t') = [t_0 - \ell(B), t')$ and thus it should visited by some robot in the time interval $[t_0 - \ell(B), t_0 + \ell(B)]$. Take r_m to be the robot that visits p' the latest in this time interval and then the right endpoint of I_{i_1} is placed in the left bag of I_m. This concludes the second placement of the endpoints of the non-empty intervals.

Thus, for the sites p and p' defined above, we consider visits of them by the robots; intuitively, we consider once when are visited on the "left" side of the interval $\mathcal{I}$ and another time on the "right". In Figure 3(a), the two visits of the sites are denoted by symbols of the same shape and color. E.g., r_5 is the first robot that visits the site $f_1(A_1)$ (colloquially speaking, the left endpoint of I_1) during the interval $[t_0 + L - \ell(A_1), t_0 + L + \ell(A_1)]$. We now prove the following crucial lemma.

▶ **Lemma 16.** *The bag graph $\mathcal{B}$ and the shortcut graph $\mathcal{S}$ have the following properties.*

(a) *$\mathcal{B}$ is a bipartite graph.*

(b) *$\mathcal{S}$ is isomorphic to the line graph of $\mathcal{B}$*

(c) *Let $\mathcal{B}'$ be a connected component of $\mathcal{B}$. If none of the vertices of $\mathcal{B}'$ belong to empty-intervals, then the number of vertices of $\mathcal{B}'$ is equal to its number of edges.*

(d) *Let $\mathcal{S}'$ be a connected component of $\mathcal{S}$. If $\mathcal{S}'$ has a vertex v of degree one, then it must be the case that v corresponds to an endpoint of a non-empty interval that has been placed (alone) in a bag of an empty interval.*

Proof. **(a)** Every endpoint of a non-empty interval is placed twice, once in a left bag and once in a right bag. Hence, all edges of the bag graph are between left bags and right bags.

(b) Observe that by the definition of the two graphs, an edge e in the bag graph between bags β and β' is formed because of the placement of an endpoint p (of a non-empty interval) in the two bags. Recall that p is a vertex in the shortcut graph $\mathcal{S}$. We claim the shortcut graph $\mathcal{S}$ is isomorphic to the line graph of $\mathcal{B}$ using the bijection that maps e (a vertex in $\overline{\mathcal{B}}$) to the vertex p in $\mathcal{S}$. We now show that this is an isomorphism.
Consider two edges e_1 and e_2 in the bag graph in which e_i is between bags β_i and β_i', and it is formed because of the placement of an endpoint p_i of a non-empty interval in the corresponding bags, for $i = 1, 2$. We consider two cases. First, assume, e_1 and e_2 are adjacent, and so w.l.o.g, assume $\beta_1 = \beta_2$. For this to happen, p_1 and p_2 must be placed in the same bag $\beta_1 = \beta_2$. Thus, by the definition of the shortcut graph, they are connected in $\mathcal{S}$. The second case is when e_1 and e_2 are not adjacent which implies all the bags $\beta_1, \beta_1', \beta_2$, and β_2' are distinct. In this case observe that p_1 and p_2 are not placed in the same bag ever, since each of them is placed only twice, and that no other bag can contain them. As a result, they are not connected with an edge in the shortcut graph.

(c) Let $\beta_1, \ldots, \beta_r$ be the vertices of $\mathcal{B}'$. By assumption, none of the bags belong to an empty interval. However, recall that for a bag β_i, $1 \leqslant i \leqslant r$, the endpoint of its corresponding non-empty interval is placed in its own bag β_i, and then once more in some other bag β'. But since $\beta_1, \ldots, \beta_r$ form a connected component, it must be the β' should be one of the bags $\beta_1, \ldots, \beta_r$, other than β_i. As a result, we can charge the edge between β' and β_i to β_i and it is clear that every bag will be charged exactly once, meaning, there would be exactly r edges between the vertices $\beta_1, \ldots, \beta_r$.

(d) Observe that if during the second placement, the left (resp. right) endpoint p_1 of a non-empty interval is placed in the right (resp. left) bag β of a non-empty interval, then by our first placement, β contains the right (resp. left) endpoint p_2 of some interval but crucially, $p_1 \neq p_2$ and thus p_1 will be connected to at least one other endpoint in the shortcut graph. As a result, the only possible way for p_1 to end up with degree one is that p_1 is placed in a bag of an empty interval, and also no other endpoint is placed in that bag.

◀

The patrol graph. We are now ready to define the patrol graph $\mathcal{P}$ (an example is shown in Figure 3(d)). Recall that k' is the number of non-empty intervals. Initially, $\mathcal{P}$ consists of k' isolated edges, one for each non-empty interval. Let us color these initial edges black. We now define a natural bijection J between the vertices of $\mathcal{P}$ and the vertices of $\mathcal{S}$: a vertex v of $\mathcal{S}$ is an endpoint of a non-empty interval and $J(v)$ maps it to the vertex in $\mathcal{P}$ that represents the same endpoint. In a process that we will describe shortly, we will add a subset of edges of $\mathcal{S}$ to $\mathcal{P}$ and color them blue. To add an edge $e = (u, v)$ of $\mathcal{S}$ to $\mathcal{P}$, we simply add an edge between $J(u)$ and $J(v)$. To describe the set of blue edges that we add, consider a connected component $\mathcal{S}'$ of $\mathcal{S}$ and let $\mathcal{B}'$ be the connected component of $\mathcal{B}$ where $\mathcal{S}' = \overline{\mathcal{B}'}$. We now add a number of blue edges, using the following cases:

- (case i) $\mathcal{B}'$ has an even number of edges. Then by Theorem 13 and Lemma 3(b), $\mathcal{S}'$ has a perfect matching M. We add all edges of M to $\mathcal{P}$.
  For example, in Figure 3, we add the perfect matching $\{A_2B_2, B_1A_3\}$ in the central connected component to the patrol graph.
- (case ii) If case (i) does not hold, but none of the vertices of $\mathcal{B}'$ belong to empty intervals, then by Lemma 3(c), the graph $\mathcal{B}'$ has at least one cycle C. However, by Lemma 3(a),

the graph $\mathcal{B}'$ is bipartite meaning, C is an even cycle. Consequently, by Lemma 15, we can decompose $\mathcal{B}'$ into a number of 2-paths and a claw. The line graph of a claw is a triangle and thus it follows that the vertices of $\mathcal{S}'$ can be decomposed into a matching M and a triangle Δ. We add the edges of M and Δ to $\mathcal{P}$.
For example, in Figure 3 we have added the triangle formed by A_4, B_4, and B_3 to the patrol graph (here the matching part is empty).

- (case iii) If none of the cases (i) or (ii) hold, then at least one of the vertices of $\mathcal{B}'$ belongs to an empty interval. In this case, we simply decompose $\mathcal{S}'$ into a matching and an isolated vertex, since by Theorem 14, $\mathcal{B}'$ can be partitioned into a number of 2-paths and one edge. We add the edges in the matching to $\mathcal{P}$ (in Figure 3 we have A_1 as an isolated vertex).

▶ **Lemma 17.** *The patrol graph $\mathcal{P}$ consists of k' pairwise non-adjacent black edges and a number of blue edges. Any blue edge (v, w) in $\mathcal{P}$ corresponds to an edge in the shortcut graph $\mathcal{S}$. Furthermore, the set of blue edges can be decomposed into a matching and a number of triangles. In addition, any vertex of $\mathcal{P}$ that is not adjacent to a blue edge can be charged to a bag of an empty interval.*

A.4 Obtaining a Cyclic Solution

The main and the most crucial consequence of our sweeping strategy is the following lemma.

▶ **Lemma 18.** *Consider two adjacent vertices v and w in the shortcut graph. This means that there are two non-empty intervals I_1 and I_2 such that v corresponds to an endpoint A of I_1 and w corresponds to an end point B of I_2 and A and B are placed in the same bag. Let s_1 be the site visited at A during I_1 and s_2 be the site visited at B on I_2. Then, we have $d(s_1, s_2) \leqslant \ell(A) + \ell(B)$.*

Proof. We have two cases: either both A and B are left or right endpoints, or one of them is a right endpoint while the other one is a left endpoint. By symmetry, it suffices to consider the following two cases.

- A and B are both left endpoints. For A and B to be placed in the same bag, they must be placed in the right bag of an interval I_3 (of a robot r_3) during the second placement of A and B. Note that I_3 could be equal to both I_2 or I_1 but since A and B are both left endpoints, I_1 and I_2 are distinct intervals. As we argued during the second placement, r_3 visits s_1 in the time interval $[t_0 + L - \ell(A), t_0 + L + \ell(A)]$ and also visits s_2 in the time interval $[t_0 + L - \ell(B), t_0 + L + \ell(B)]$. As r_3 travels at unit speed, it thus follows that the distance between s_1 and s_2 cannot be larger than the maximum distance between the points of the two intervals $[t_0 + L - \ell(A), t_0 + \ell(A) + L]$, $[t_0 + L - \ell(B), t_0 + \ell(B) + L]$. Both of these intervals are centered at the point $t_0 + L$, one has diameter $2\ell(A)$ and the other the diameter $2\ell(B)$ and thus the maximum distance between them is $\ell(A) + \ell(B)$.
- A is a left endpoint and B is a right endpoint. Let r_1 and r_2 be the robots that correspond to I_1 and I_2. In this case, in order for A and B to be placed in the same bag, the second placement of A should be placed in the same bag as the first placement of B or vice versa. W.l.o.g, assume the former, which implies s_1 must be visited by robot r_2 during the time interval $[t_0 + L - \ell(A), t_0 + L - \ell(A)]$; however, observe that s_2 is visited by r_2 at time $t_0 + L - \ell(B)$ since this is the position of the endpoint B on the time axis. Once again, it follows that the distance between A and B is at most the maximum distance between the point $t_0 + L - \ell(B)$ and the interval $[t_0 + L - \ell(A), t_0 + L + \ell(A)]$ which equals $\ell(A) + \ell(B)$.

◀

Using the patrol graph defined previously, we can obtain an efficient cyclic solution. Let $\mathcal{P}_i$ be a connected component of $\mathcal{P}$. We turn $\mathcal{P}_i$ into one cycle. Let x be the number of black edges of $\mathcal{P}_i$; it follows that $\mathcal{P}_i$ has $2x$ vertices. Let $\mathcal{P}_i^c$ be the graph obtained by contracting all the blue edges in $\mathcal{P}_i$ (see Figure 3(e)). We then use Lemma 12 to duplicate a number of (black) edges of $\mathcal{P}_i^c$ such that the resulting graph, denoted by $\mathcal{P}_i^{Ec}$, is Eulerian (see Figure 3(f)). We call $\mathcal{P}_i^{Ec}$ an Eularized contracted patrol graph (ECPG). We now look back at $\mathcal{P}_i$. For every edge that is duplicated in $\mathcal{P}_i^{Ec}$, we duplicate the edge as well as its two endpoints; this basically corresponds to adding another copy of the same interval to the picture. In addition, we add a blue edge between the two copies of the same vertex as well as between any two vertices that are connected in the shortcut graph. See Figure 4 (b). Let $\mathcal{P}_i^f$ be the resulting graph. The collection of all such graphs will form the *final patrol graph* and thus $\mathcal{P}_i^f$ is a connected component of the final patrol graph.

▶ **Observation 19.** *Let $\mathcal{P}_i^f$ be a connected component of the final graph. It has the following properties. $\mathcal{P}_i^f$ consists of a number of pairwise non-adjacent black edges and a number of blue edges. Contracting all the blue edges of $\mathcal{P}_i^f$ results in an Eulerian graph. The blue edges can be partitioned into a number of vertex disjoint cliques. For every blue edge (v_1, v_2), the vertices v_1 and v_2 represent two endpoints A_1 and A_2 of two non-empty intervals I_1 and I_2 such that for the sites s_i visited at A_i, $i = 1, 2$, we have $d(s_1, s_2) \leqslant \ell(A_1) + \ell(A_2)$.*

Proof. The first two claims are obvious due to our constructions and definitions. For the third claim, observe that by Lemma 5, the blue edges of a patrol graph can be decomposed into (vertex disjoint) a number of K_2's and K_3's. However, the duplication process also duplicates the blue edges, meaning, if a vertex of a blue clique K_i is duplicated, it is still connected to all the other vertices of K_i as well as it duplicate copy. Thus, a blue clique K_i can turn into a blue clique $K_{i'}$ where $i' \leqslant 2i$. For the last property, if the edge (v_i, v_2) is between the two copies of the same interval, then it follows that $I_i = I_2$ and thus $s_i = s_2$ and thus $d(s_i, s_2) = 0$. Otherwise, zero, one or both vertices of v_i, v_2 could be copies of another vertex. Assume v_i' and v_2' are the original vertices. By construction it holds that (v_i', v_2') is a blue edge in a patrol graph and thus the last property holds by Lemma 4. ◀

A *bichromatic cycle* is a cycle of even size that is made of edges of alternating colors of black and blue. We can use the above observation to obtain the following result.

▶ **Lemma 20.** *Let $\mathcal{P}_i^f$ be a connected component of the final patrol graph with x black edges (the duplicates of the black edges are also counted). We can find a bichromatic cycle in $\mathcal{P}_i^f$ that covers all the black edges. Consequently, it yields a cyclic route of length at most xL that visits all the sites visited by the intervals that correspond to the black edges of $\mathcal{P}_i$.*

Proof. Let $\mathcal{P}_i^{Ec}$ be the ECPG associated with $\mathcal{P}_i^f$. Fix one Euler tour of $\mathcal{P}_i^{Ec}$, i.e., an ordering $e_1, \cdots, e_x$ of the edges of $\mathcal{P}_i^{Ec}$ that forms a closed walk. See Figure 4(a). The first part of the lemma is easy: for every j, $1 \leqslant j \leqslant x$, e_j and e_{j+1} (where $e_{x+1} = e_1$) are connected with a blue edge by Observation 19 and thus the Euler tour in $\mathcal{P}_i^{Ec}$ corresponds to a bichromatic tour in the final patrol graph.

For the last part, consider the step j of the walk and let $I_j = [A_j, B_j]$ and $I_{j+1} = [A_{j+1}, B_{j+1}]$ be the intervals that correspond to the edges $e_j = (v_j, u_j)$ and $e_{j+1} = (v_{j+1}, u_{j+1})$ of $\mathcal{P}_i^{Ec}$ where we assume A_j, B_j, A_{j+1}, and B_{j+1} correspond to v_j, u_j, v_{j+1}, and u_{j+1} respectively. We make the convention that index $x + 1$ refers to index 1 in the above definitions. Note that with this notation, e_j and e_{j+1} could be directed in either way in the

Euler tour, i.e., each edge can be directed from left to right, or right to left. If e_j is directed from v_j to u_j (i.e., from A_j to B_j), then define $\textsc{Start}(e_j) = A_j$ and $\textsc{End}(e_j) = B_j$ and if it is from u_j to v_j (B_j to A_j), then define $\textsc{Start}(e_j) = B_j$ and $\textsc{End}(e_j) = A_j$. Consider the same definitions for e_{j+1}.

We now turn the Euler tour into a cyclic tour by considering an imaginary robot that travels along it. The robot starts at $\textsc{Start}(e_1)$ and takes j steps. At step j, it travels the entire length of e_j, meaning, it traverse the route that corresponds to e_j, then it takes the shortcut from $\textsc{End}(e_j)$ to $\textsc{Start}(e_{j+1})$ (to be more precise, it takes the shortcut from the sites that correspond to those endpoints) and then continues with step $j+1$. Note that at the end of step x, it takes the shortcut from $\textsc{End}(e_x)$ to $\textsc{Start}(e_1)$. Observe that for each edge e_j, we take one shortcut from e_{j-1} to e_j and one shortcut from e_j to e_{j+1}. Next, observe that by Lemma 4, we can charge the cost of the first shortcut to $\ell(\textsc{End}(e_{j-1})) + \ell(\textsc{Start}(e_j))$ and the cost of the second shortcut to $\ell(\textsc{End}(e_j)) + \ell(\textsc{Start}(e_{j+1}))$. The crucial point here is that each of $\ell(\textsc{Start}(e_j))$ and $\ell(\textsc{End}(e_j))$ is used exactly once for each edge. Finally, observe that $\ell(\textsc{Start}(e_j))$ plus $\ell(\textsc{End}(e_j))$ plus the length of the interval corresponding to e_j is exactly L. This shows that the imaginary robot travels a tour of length at most xL. ◀

We are now almost done and we can finally describe our cyclic solution. Bear in mind that in our scheme, we have k' non-empty intervals that corresponds to tours that visit all the sites. We say a robot is *useful* if its interval is non-empty and *useless* otherwise. Consider a connected component P_i of the patrol graph (**not** the final patrol graph) with y black edges. If $y > k'/2$, then we call P_i a *big* connected component, otherwise, it is *small*. Observe that by this definition, there can be at most one big connected component. Consider the graphs $\mathcal{P}_i^c$, $\mathcal{P}_i^{Ec}$, and $\mathcal{P}_i^f$ associated with $\mathcal{P}_i$. We now consider a few cases.

- (case 1) $\mathcal{P}_i^c$ has at least two vertices v_1 and v_2 of degree 1. Clearly, this means that v_1 and v_2 were not adjacent to any blue edges before contraction of blue edges, i.e., they have still degree 1 in $\mathcal{P}_i$. In this case, by Lemma 12, we can Eulerize $\mathcal{P}_i^c$ by duplicating at most all the y edges. As a result, $\mathcal{P}_i^f$ has at most $2y$ edges and therefore by Lemma 20, we can turn it into a tour of length at most $2yL$. In this case, we can distribute all the y useful robots corresponding to the y non-empty intervals of $\mathcal{P}_i$ and one useless robot. We have a useless robot available because by Lemma 5, a vertex that is not adjacent to a blue edge can be charged to a bag of an empty interval and thus two such vertices can pay for a useless robot. Thus, using $y+1$ robots, we obtain a tour with latency

$$\frac{2yL}{y+1} \leqslant \frac{2(k-1)L}{k}$$

  where the inequality uses the fact that $y \leqslant k-1$ (as there is at least one useless robot).
- (case 2) $\mathcal{P}_i^c$ has exactly one vertex v_1 of degree 1. In this case, by Lemma 12, we can Eulerize $\mathcal{P}_i^c$ by duplicating at most $y-1$ edges. As a result, $\mathcal{P}_i^f$ has at most $2y-1$ edges and therefore by Lemma 20, we can turn it into a tour of length at most $(2y-1)L$. But here we have two additional sub-cases. If P_i is not a big connected component, then we simply assign all the y robots to the resulted Euler tour. Here, P_i and $\mathcal{P}_i^c$ have at most $k'/2$ edges, meaning, $y \leqslant k'/2$. In addition, since v_1's degree is 1, there must be at least one empty bag—and, hence, at least one useless robot—by Lemma 5. Thus $k' \leqslant k-1$. (Note that we are not using this useless robot here; we are still assigning only y robots to the tour. We are only using the existence of the useless robot to conclude that $k' \leqslant k-1$.) Hence, the latency can be bounded as follows.

$$\frac{(2y-1)L}{y} \leqslant \frac{(2k'/2-1)L}{k'/2} = \frac{2(k'-1)L}{k'} < \frac{2(k-1)L}{k}.$$

However, if P_i is big, then we allocate one useless robot to the resulting tour. Note that here as $\mathcal{P}_i^c$ only has one vertex of degree one, it can only be charged to one bag of an empty interval, i.e., "half a useless robot". But since we have at most one big component, we can afford to allocate a full useless robot to the resulting tour. The latency here would be

$$\frac{(2y-1)L}{y+1} \leqslant \frac{(2k'-1)L}{k'+1} \leqslant \frac{(2k-3)L}{k} < \frac{2(k-1)L}{k}.$$

- (case 3) $\mathcal{P}_i^c$ has no vertex of degree one. In this case, by Lemma 12, we can Eulerize $\mathcal{P}_i^c$ by duplicating at most $y-2$ edges. As a result, $\mathcal{P}_i^f$ has at most $2y-2$ edges and therefore by Lemma 20, we can turn it into a tour of length at most $(2y-2)L$. Thus, using y robots, we obtain a tour with latency

$$\frac{(2y-2)L}{y} \leqslant \frac{2(k-1)L}{k}.$$

In all of the above cases, we obtain a cyclic tour of latency at most $2(k-1)L/k$ proving our main result.

B On the decidability of the decision problem

We note that the problem whether the latency is at most ℓ is decidable for the case that ℓ and the distances are all integers.[2] To see this, consider the graph, in which we add dummy sites along the edges in $P \times P$ at unit distance.

We already know that we can assume that robots at time $t = 0$ start at sites, and that at any time they either wait at a site or move at unit speed. Assume we have an optimal schedule $\sigma(R)$ with these properties and latency L, where $L \leqslant \ell$. We modify this schedule by rounding the times t at which a robot leaves a site down the next integer $\lfloor t \rfloor$ (and then let it move with unit speed to the next site).

First observe that in such a schedule robots always reach sites at integer times. Thus, inductively we can conclude that the robot that would have left a site at time t already was at the site at time $\lfloor t \rfloor$ (and therefore can actually leave then).

Secondly the latency cannot increase to a value larger than ℓ: The latency might increase by some value $x < 1$ and is an integer after adapting the schedule. If it would be larger than ℓ, it would be at least $\ell + 1$. But then the latency L of the original schedule would have been $L \geqslant \ell + 1 - x > \ell$, a contradiction.

With these observations, we can now conclude that the problem is decidable. To describe a schedule, we only need to know for any time $t = 0, 1, \ldots$ at which site (original or dummy) it is. Consider the schedule $\sigma(R)(t) = (f_1(t), \ldots, f_k(t))$ at such a time. If we have N sites, there are only N^k options for $\sigma(R)(t)$. This means by the pigeonhole principle that after $N^k + 1$ time steps, we must have had a repeated schedule $\sigma(R)(t)$. At that point in time, we can make the schedule periodic without increasing the latency. Thus, by checking at most $\left(N^k\right)^{N^k+1}$ schedules we can decide the problem.

This however, leaves the case for points in $\mathbb{R}^2$ open, even if the points have integer coordinates.

[2] We thank a reviewer for pointing out this case.