On Optimal Cooperative Patrolling

Fabio Pasqualetti, Antonio Franchi, and Francesco Bullo

Abstract—This work considers the problem of designing optimal multi-agent trajectories to patrol an environment. As performance criterion for optimal patrolling we consider the worst-case time gap between any two visits of the same region. We represent the area to be patrolled with a graph, and we characterize the computational complexity of the trajectory design (patrolling) problem with respect to the environment topology and to the number of robots employed in the patrolling task. Even though the patrolling problem is generally NP-hard, we identify particular cases that are solvable efficiently, and we describe optimal patrolling trajectories. Finally, we present a heuristic with performance guarantees, and an $\varepsilon$-approximation algorithm to solve the NP-hard patrolling problem.

I. INTRODUCTION

The recent development in the autonomy and the capabilities of mobile robots has greatly increased the number of application suitable for a team of autonomous agents. Particular interest has been received by the tasks requiring continual execution, such as the detection of forest fires and the patrol (surveillance) of an environment [1], [2]. The surveillance of an area of interest requires the robots to continuously and repeatedly travel the environment, and the challenging problem consists in scheduling the robots trajectories so as to optimize certain performance criteria. The reader familiar with network location, multiple traveling salesman, or graph exploration problems may observe a close connection with the patrolling problem we address, e.g., see [3], [4]. It is worth noting, however, that these classical optimization problems do not capture the repetitive, and hence dynamic, aspect of the patrolling problem, nor the synchronization issues that arise when a timing among the visits of certain zones is required.

A precise formulation of the patrolling problem requires the characterization of the environment to be patrolled, of the robots capabilities, and of the performance criteria. In this work, we represent the environment as a graph, in which the vertices correspond to physical, and strategically important, locations, and in which the edges denote the possibility of moving between two connected locations. We assume that the robots are able to independently build such graph by exploring the environment. For what concern the robots, we assume them to be identical and capable of sensing and communicating within a certain spatial range, and of moving according to a first order integrator dynamic. Additionally, we assume that, when a robot is placed at each of the graph vertices, the union of the sensors footprint provide complete sensor coverage of the environment. Regarding the performance criterion, we consider the longest time gap (refresh time) between any two visits of the same location in the environment.

The problem of designing team trajectory to satisfy certain temporal constraints has recently gained attention, e.g., see [5]. The situation in which the motion of the robots needs to guarantee a uniform frequency coverage of the environment has been considered, among others, in [6]. However, because we are interested in the worst case scenario, the trajectories proposed in these works are not optimal in our sense. In [7], an empirical evaluation of existing patrolling heuristics is performed. In [8], two classes of strategies, based respectively on space decomposition and traveling salesperson tour computation, are presented and qualitatively compared. In [2] and [9], an efficient and distributed solution to the perimeter patrolling problem is proposed.

In this work, we will study the computational complexity of the minimum refresh time patrolling problem. The computational complexity theory, a comprehensive discussion of which is in [10], is a branch of the theory of computation in computer science and mathematics that focuses on classifying problems according to their inherent difficulty. Of particular interest is the NP-hard class of problems, which contains all the problems that are, informally, as hard as the hardest problems in the class NP, for which no polynomial time algorithm is known to compute an optimal solution.

The main contributions of this work are as follows. We give a procedure to build a graph (roadmap) to represent the topological structure of the area to be patrolled, and we study the computational complexity of designing minimum refresh time team trajectories as a function of the shape of the environment, and of the number of robots employed in the patrolling task. We identify the following three cases: (i) chain environment, (ii) tree environment, and (iii) cyclic environment. For the case of a chain environment, we characterize a family of minimum refresh time team trajectories. We show that such trajectories can be computed with an algorithm which is polynomial in the number of vertices of the graph representing the environment. We also develop an algorithm to compute a partition (of given cardinality) of a chain graph, so as to minimize the maximum length of the clusters in the partition. For the case of a tree environment,
we prove that, if the number of robots is allowed to be
a function of the cardinality of the graph representing
the environment, then the problem of finding minimum refresh
time team trajectories belongs to the class of NP-hard
problems. However, a polynomial algorithm exists if the
number of robots if fixed a priory and constant. For the latter
case, we characterize a family of minimum refresh time team
trajectories. Finally, we show that the general case of cyclic
environment is computationally hard to solve. We propose
two approximation algorithms, and we characterize their
performance. The first approximate solution is extremely
easy to compute, but its performance depend upon the ratio
of the longest to the shortest edge in the graph representing
the environment. The second approximation algorithm is
based on a path-covering procedure, which is polynomial
in the size of the graph, and which allows us to compute a
team trajectory whose refresh time is, independently of the
environment topology, within a factor 8 of the refresh time of
an optimal team trajectory. To the best of our knowledge, this
algorithm is the first constant factor approximation algorithm
for the NP-hard minimum refresh time patrolling problem.

The rest of the paper is organized as follows. In Section II
we define the notation and the problem under consideration.
In Sections III, IV, and V we study the patrolling problem for
the case of chain, tree, and cyclic environment, respectively.
Finally, Section VI contains our conclusion.

II. DEFINITIONS AND PRELIMINARY CONCEPTS

We are given a team of \( m \) identical robots capable of
sensing and moving in a connected environment.\(^2\)

Regarding sensing, we assume that the environment can be
completely covered by simultaneously placing a robot at
each of a set of \( n \) viewpoints in the configuration space.\(^3\)
In other words, if and only if \( m = n \) robots were placed
at the \( n \) viewpoints, then the union of the sensors footprint
of each robot would provide complete sensor coverage of the
environment. However, we assume \( n > m \) so that at
least one robot needs to visit more viewpoints for the entire
environment to be monitored over time.

Regarding motion, we assume that the robots are holo-
nomic, i.e., modeled as first order integrators, and move at
most at unitary speed. Additionally, we associate a robotic
roadmap \( G \) with the environment [12]. Precisely, the \( n \)
viewpoints form the vertex set of \( G \), and the unordered pair
\((i, j)\) belongs to the edge set of \( G \) if a robot can travel from \( i \)
to \( j \). We constrain the motion of the robots on the roadmap,
and we assume that the paths corresponding to the edges of
\( G \) verify the triangle inequality. Finally, we let the length
of a shortest path from \( i \) to \( j \) equal the weight of the edge
\((i, j)\). An example is in Fig. 1.

Let \( G = (V, E) \) denote a robotic roadmap, where \( V \) and \( E \)
de note the vertex set and the edge set, respectively. A team

\[ \text{RT}(X) = \max_{v \in V} \max_{(t_1, t_2) \in \Omega(v, X)} t_2 - t_1 \]

Fig. 1. A polygonal environment and an associated roadmap. The 12
vertices coincide with a set of viewpoints from which the robots provide
sensor coverage of the entire environment. Each edge corresponds to the
shortest path between its endpoints. The 3 robots, which are holonomic and
disk-shaped (grey-filled circles), are constrained to move on the roadmap.
Because of the dimension of the robots, some paths are not straight lines.

\(^2\)Even though our results hold for any \( m \in \mathbb{N} \), we focus on the case
\( m \geq 2 \), and we remark that, if \( m = 1 \), then the methods developed for the
shortest tour computation can be used to design patrolling trajectories.

\(^3\)The set of viewpoints can be computed, for instance, by solving an art

III. MINIMUM REFRESH TIME TEAM TRAJECTORY ON A
CHAIN ROADMAP

We characterize in this section an optimal refresh time
team trajectory when the roadmap associated with the envi-
ronment has a chain structure.
A. Open loop team trajectory characterization

Let $N_i$ denote the neighbor set of the vertex $i$, and let $|N_i|$ denote the degree of $i$. A chain roadmap is an undirected, connected, and acyclic graph, in which every vertex has degree two, except for two vertices which have degree one. Without losing generality, we assume that the $n$ vertices are ordered in a way that $|N_1| = |N_n| = 1$, and $N_i = \{i-1, i+1\}$ for each $i \in \{2, \ldots, n-1\}$. We define a relative order of the robots according to their position on the roadmap. A team trajectory is order invariant if the order of the robots does not change with time, i.e., if $x_i(t) \leq x_{i+1}(t)$ for each $i \in \{1, \ldots, m-1\}$ and for every instant $t$, where $x_i$ denotes the distance on the roadmap from the first vertex of the chain to the position of the $i$-th robot.

Proposition 3.1 (Order invariant team trajectory): Let $X$ be a team trajectory. There exists an order invariant team trajectory $\bar{X}$ such that $RT(X) = RT(\bar{X})$.

Proof: Let $X$ be a team trajectory, and consider the permutation matrix $P(t)$, that keeps track of the order of the robots at time $t$. Notice that the $(i,j)$-th entry of $P(t)$ is 1 if, at time $t$, the $i$-th robot occupies the $j$-th position in the chain, and it is 0 otherwise. Since $X$ is continuous, when the function $P(t)$ is discontinuous, the positions of the robots directly involved in the permutation overlap. Therefore, the order invariant team trajectory $\bar{X} = P^{-1}(t)X(t)$ is a feasible team trajectory such that $RT(X) = RT(\bar{X})$. \hfill \Box

For a team trajectory $X$, let $V_i$ be the set of viewpoints visited over time by the agent $i$, and let $\{V_1, \ldots, V_m\}$ be the image of $X$. Let $l_i = \min_{v \in V_i} v$ and $r_i = \max_{v \in V_i} v$, and let $d_i = r_i - l_i$. Finally, let $RT^* = \min_{X} RT(X)$. A team trajectory is non-overlapping if $V_i \cap V_j = \emptyset$ for all $i, j \in \{1, \ldots, m\}$ with $i \neq j$.

Proposition 3.2 (Non-overlapping team trajectory): Given a chain roadmap, there exists an order invariant and non-overlapping team trajectory with refresh time $RT^*$. \hfill \Box

As a consequence of Propositions 3.1 and 3.2, there exists a minimum refresh time team trajectory whose image $\{V_1, \ldots, V_m\}$ coincides with an optimal $m$-partition of the chain roadmap. To conclude this section, we prove the following.

Theorem 3.3: (Minimum refresh time on a chain roadmap): Let $G$ be a chain roadmap, and let $m$ be the number of robots. Then $RT^* = 2 \min_{\Pi_m} \dim(\Pi_m)$.

Proof: As a consequence of Propositions 3.1 and 3.2, there exists a minimum refresh time team trajectory in which the set $\{V_1, \ldots, V_m\}$ coincides with an optimal $m$-partition $\Pi_m$ of the chain roadmap. Since each robot is assigned to a different cluster, we have $RT^* \geq 2 \dim(\Pi_m)$. Let $X$ be a trajectory where each robot sweeps at maximum speed its cluster. Clearly, $RT(X) = RT^* = 2 \dim(\Pi_m)$.

As a consequence of Proposition 3.2 and Theorem 3.3, there exists a minimum refresh time team trajectory in which the set $\{V_1, \ldots, V_m\}$ coincides with an optimal $m$-partition of the chain graph.

B. Optimal $m$-partition centralized computation

In the remaining part of the section we describe an algorithm to compute an optimal $m$-partition. \hfill \Box

Given a set of viewpoints $V$, we call left-induced partition of length $l$ the partition $\Pi^l = \{\pi_1, \ldots, \pi_k\}$ defined recursively as

$$\pi_i = \{v \in V : a_i \leq v \leq a_i + l\}, \quad i = 1, \ldots, k,$$

where (cf. Fig. 2(a))

$$a_1 = v_1,$$
$$a_i = \min\{v \in V : v > a_{i-1} + l\}, \quad i = 1, \ldots, k.$$  

Note that, by definition, $k$ is such that $\{v \in V : v > a_k + l\} = \emptyset$, and $\dim(\Pi^l) \leq l$. Observe that the function $|\Pi^l|$ is monotone, non-increasing, and right-continuous (cf. Fig. 2(b)). Let $\{l_1, \ldots, l_{n-1}\}$ be the discontinuity points of the function $|\Pi^l|$, then, for $k \in \{1, \ldots, n-1\}$,

$$|\Pi^l| \leq k, \quad \text{if } l \geq l_k, \quad \text{and } |\Pi^l| > k, \quad \text{if } l < l_k. \quad (2)$$

Note that two or more discontinuity points of $|\Pi^l|$ may coincide, so that the function $|\Pi^l|$ may not assume all the values of the set $\{1, \ldots, n\}$, as in Fig. 2(b) for $|\Pi^l| = 9$.

By removing the longest edges in the chain the average length of the clusters is minimized. Note that, in general, such partition does not minimize the dimension of the $m$-partition, and hence it is not optimal in our sense.
Algorithm 2: Optimal left-induced $m$-partition

Input : $\{v_1, \ldots, v_n\}$, $0 < m < n$, $\varepsilon > 0$;
Set $a := 0$, $\delta > 0$, $b := \frac{\varepsilon n + \delta}{m}$, $l := \frac{(a+b)}{2}$;
while $(b-a) > 2\varepsilon$ do
   $\Pi^l := \text{left-induced}(\{v_1, \ldots, v_n\}, l)$;
   if $|\Pi^l| > m$ then
      $a := a$, $l := \frac{a+b}{2}$;
   else
      $\Pi^l := \Pi^l$, $b := l$, $l := \frac{a+b}{2}$;
   return $\Pi^*$

Theorem 3.4 (Optimal $m$-partition): Let $G$ be a chain roadmap $G$. Let $\Pi_m$ be an $m$-partition of $G$, and let $\Pi^l$ be the left-induced partition of length $l$ of $G$. Then

$$\min_{\Pi_m} \dim(\Pi_m) = \min \{l : |\Pi^l| \leq m\}.$$

Proof: Let $\Pi_m$ be an $m$-partition, and let $\Pi^l = \{\pi^1, \ldots, \pi^m\}$ be the left induced partition of length $l$ of a chain roadmap $G$. Let $l^* = \min_{\Pi_m} \dim(\Pi_m)$. We want to show that $l^*$ is one of the discontinuity points of the function $|\Pi^l|$, i.e., that $l^*$ verifies the conditions (2). By contradiction, if $l < l^*$ and $|\Pi^l| \leq m$, then an $m$-partition with dimension smaller than the optimal would exist. Therefore, if $l < l^*$, then $|\Pi^l| > m$. Suppose now that $l \geq l^*$, and let $\Pi^*_m = \{\pi^1, \ldots, \pi^*_m\}$ be an $m$-partition with minimum dimension. Notice that $|\pi^1_l| \geq |\pi^*_1|$, because the cluster $\pi^1_l$ contains all the viewpoints within distance $l$ from $v_1$, and hence also within distance $l^*$. It follows that $\max \pi^1_l \geq \max \pi^*_1$, and also that $\min \pi^l_2 \geq \min \pi^*_2$. By repeating the same procedure to the remaining clusters, we obtain that $\max \pi^l_m \geq \max \pi^*_m$, so that, if $|\Pi^l| = m$ and $l \geq l^*$, then $|\Pi^l| \leq m$. ■

Following Theorem 3.4, an optimal left-induced partition of cardinality at most $m$ is also an optimal $m$-partition. A procedure to compute an optimal left-induced partition is in Algorithm 2, where the function left-induced($\{v_1, \ldots, v_n\}, l$) returns the left-induced partition defined in (1). We next characterize its convergence properties.

Lemma 3.5 (Convergence of Algorithm 2): Let $G$ be a chain roadmap, and let $\Pi_m$ be an $m$-partition of $G$. Let $l^* = \min_{\Pi_m} \dim(\Pi_m)$. Then,

(i) Algorithm 2 returns a left-induced partition of dimension at most $l^* + \varepsilon$, and cardinality at most $m$,

(ii) the time complexity of Algorithm 2 is $O(n \log(\varepsilon))$.

Proof: Algorithm 2 search for the minimum length $l^*$ that generates a left-induced partition of cardinality at most $m$. Because of Theorem 3.4, the length $l^*$ coincides with one of the discontinuity points of the function $|\Pi^l|$, and it holds $l^* \in (0, v_n/m + \delta)$, where $\delta > 0$. Indeed, $l^* > 0$ because $m < n$, and $l^* < v_n/m + \delta$, because $(v_n/m + \delta)m > v_n$. Recall from (2) that $|\Pi^l| > m$ for every $l < l^*$, and that the function $|\Pi^l|$ is monotone. Note that the interval $[a, b]$ as updated in Algorithm 2, contains the value $l^*$ at every iteration. The length of the interval $[a, b]$ is divided by 2 at each iteration, so that, after $\log_2((v_n/m + \delta)/\varepsilon)$, the value $l^*$ is computed with precision $\varepsilon$. Since the computation of $|\Pi^l|$ can be performed in $O(n)$ operations, the time complexity of Algorithm 2 is $O(n \log(\varepsilon))$. ■

For ease of notation, in the following sections, we use the set $\{V_1, \ldots, V_m\}$ to denote both the image of a team trajectory and an $m$-partition of the chain graph.

IV. MINIMUM REFRESH TIME TEAM TRAJECTORY ON AN ACYCLIC ROADMAP

The problem of designing team trajectories for a tree roadmap is now considered. Let $T = (V, E)$ denote an undirected, connected, and acyclic roadmap (tre). Recall that a vertex path is a sequence of vertices such that any pair of consecutive vertices in the sequence are adjacent. A tour is a vertex path in which the start and end vertices coincide, and in which every vertex of $T$ appears at least once in the sequence. A depth-first tour of $T$ is a tour that visits the vertices $V$ in a depth-first order [13]. Let $DFT(T)$ denote the length of a depth first tour of $T$. Notice that the length of a depth-first tour of a connected tree equals twice the sum of the length of the edges of the tree, and that any depth-first tour is a shortest tour visiting all the vertices. We now show that, for the case of tree roadmap, the set of cyclic and partition strategies described in [8] does not contain, in general, a minimum refresh time trajectory. Recall that in a cyclic based strategy the robots travel at maximum speed and equally spaced along a minimum length tour visiting all the viewpoints. Consider the tree roadmap of Fig. 3(a), and

5 A distributed implementation of Algorithm 2 requires the computation of the left-induced partition in a distributed way. Such computation can be performed by simple programming operations and it is not described here.
suppose that two robots are assigned to the patrolling task. Clearly, the minimum refresh time is $2\varepsilon$, while the refresh time of a cyclic strategy equal $1 + \varepsilon$. Consider now the tree roadmap in Fig. 3(b), where the edges have unit length, and assume that two robots are in charge of the patrolling task. Observe that any partition of cardinality 2 contains a path of length 2, so that, since only one robot is assigned to each cluster, the minimum refresh time that can be obtained is 4. Suppose, instead, that the robots visit the vertices of the roadmap as specified in Table I, where $x(t)$ denotes the position of a robot at time $t$. Since the refresh time of the proposed trajectory is 3, we conclude that neither the cyclic based nor the partition based strategy may lead to a minimum refresh time team trajectory on a tree roadmap.

![Fig. 3. Two examples of tree roadmap.](image)

**TABLE I**

<table>
<thead>
<tr>
<th>Robot</th>
<th>$x(0)$</th>
<th>$x(1)$</th>
<th>$x(2)$</th>
<th>$x(3)$</th>
<th>$x(4)$</th>
<th>$x(5)$</th>
<th>$x(6)$</th>
<th>$x(7)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 v_1</td>
<td>v_2</td>
<td>v_3</td>
<td>v_4</td>
<td>v_2</td>
<td>v_3</td>
<td>v_4</td>
<td>v_2</td>
<td></td>
</tr>
<tr>
<td>2 v_2</td>
<td>v_3</td>
<td>v_1</td>
<td>v_4</td>
<td>v_2</td>
<td>v_3</td>
<td>v_4</td>
<td>v_2</td>
<td></td>
</tr>
</tbody>
</table>

We now introduce some definitions. Let $X$ be a team trajectory on the tree roadmap $T$. We say that the edge $(v_j,v_k) \in E$ is used by $X$ if there exists $i \in \{1, \ldots, m\}$ and $t_1,t_2 \in [0,RT(X)]$ such that $x_i(t_1) = v_j$ and $x_i(t_2) = v_k$, and it is used otherwise. Note that, because in a tree there exists only one path connecting two vertices, the above condition ensures that the edge $(j,k)$ is traveled by the robot $i$. Let $\bar{E}$ denote the set of unused edges, and let $F_T$ be the forest obtained from $T$ by removing the edges $\bar{E}$ from $E$, i.e., the collection of subtrees $\{T_1, \ldots, T_k\}$, with $T_i = (V_i, E_i)$, such that $V = \cup_{i=1}^k V_i$ and $E_i \subseteq E$, for each $i \in \{1, \ldots, k\}$. Let $m_i$ be the number of robots that visit at least one vertex of $T_i$ in the interval $[0,RT(X)]$, and note that $m_j > 0$ because we consider finite refresh time trajectories. Let $M = \{m_1, \ldots, m_k\}$. We refer to the pair $(F_T, M)$ as the subtree collection associated with the team trajectory $X$. Notice that the same subtree collection can be associated with different team trajectories. We say that a team trajectory is efficient if its refresh time is the smallest among all the team trajectories associated with the same subtree collection.

**Theorem 4.1 (Efficient team trajectory):** Let $(F_T, M)$ be the subtree collection associated with the team trajectory $X$ on the tree roadmap $T$, where $F_T = \{T_1, \ldots, T_k\}$, and $M = \{m_1, \ldots, m_k\}$. Then, $X$ is efficient if

$$RT(X) = \max_{j \in \{1, \ldots, k\}} \frac{DFT(T_j)}{m_j}.$$ 

**Proof:** Let $i \in \{1, \ldots, k\}$, and let $m_i$ be the number of robots assigned to $T_i$. Notice that the robots in $T_i$ travel, in total, at least $DFT(T_i)$ to visit all the vertices. Since the speed of the robots is bounded by 1, the smallest refresh time for the vertices of $T_i$ is $DFT(T_i)/m_i$. Given a subtree collection, an efficient team trajectory, an example of which is in Table I, can be computed with the following procedure.

**Lemma 4.2 (Efficient team trajectory computation):** Let $(F_T, M)$ be a subtree collection of a tree roadmap, where $F_T = \{T_1, \ldots, T_k\}$, and $M = \{m_1, \ldots, m_k\}$. An efficient team trajectory is as follows: for each $i \in \{1, \ldots, k\},$

(i) compute a depth-first tour $\tau_i$ of $T_i$,
(ii) equally space $m_i$ robots along $\tau_i$, and
(iii) move the robots clockwise at maximum speed on $\tau_i$.

**Proof:** Let $X$ be the team trajectory defined above, and observe that, for each $T_j \in F_T$, the leaves of $T_j$ appear only once in a depth first tour of $T_j$. It follows that the leaves are visited every $DFT(T_j)$ instants of time. Let $v$ be a vertex of $T_j$, and let $v$ have degree $d$. Note that $v$ is visited at least every $DFT(T_j)/m_j$, so that $RT(X) = \max_{j \in \{1, \ldots, k\}} DFT(T_j)/m_j$.

Let $P(m)$ be the partition set of $m$, i.e., the set of all the sequences of integers whose sum is $m$. The following problem is useful to characterize the complexity of designing minimum refresh time trajectories on a tree roadmap.

**Problem 2 (Optimal subtree collection):** Let $T$ be a tree roadmap and $m$ the number of robots. Find a subtree collection $(F_T, M)$ that minimizes $\max_{j \in \{1, \ldots, |F_T|\}} DFT(T_j)/m_j$, subject to $M \in P(m)$ and $|F_T| = |M|$. 

**Lemma 4.3 (Equivalent problem):** For the case of a tree roadmap, Problems 1 and 2 are equivalent.

**Proof:** As a consequence of Theorem 4.1, the minimum refresh time on a tree roadmap $T$ can be written as $\min_{(F_T, M)} \max_{j \in \{1, \ldots, k\}} DFT(T_j)/m_j$, where $(F, M)$ is a subtree collection of $T$, and $|M| = |F_T| = k \leq m$. It follows that a solution to Problem 1 can be derived in polynomial time from a solution to Problem 2 by using the procedure described in Lemma 4.2. Suppose now we have a solution to Problem 1, then an optimal subtree collection follows from the identification of the unused edges. We conclude that the two optimization problems are equivalent.

We are now able to prove the computational complexity of the problem of designing minimum refresh time team trajectories on a tree roadmap.

**Theorem 4.4 (Computational complexity of designing minimum refresh time trajectories on a tree):** Let $T$ be a tree roadmap, and let $m$ be the number of robots. The optimization Problem 1 with input $(T, m)$ is NP-hard.

**Proof:** By Theorem 4.3, Problem 1 is equivalent to Problem 2. Let $m = 2$, and $P(m) = \{\{2\}, \{1, 1\}\}$. Then Problem 2 contains the NP-hard problem presented in [14], and it is therefore also computationally hard.
Remark 1 (Fixed number of robots): If both the tree roadmap $T$ and the number of robots $m$ are arbitrary, then, because of Theorem 4.4, the problem of designing minimum refresh time team trajectories is computationally hard. However, if the number of robots is not part of the input of Problem 2, then this optimization problem is solvable in polynomial time, and precisely with time complexity $O((m-1)!n)$. We refer the interested reader to [15] for a detailed proof. In this situation, Lemma 4.2 can be used to efficiently compute a minimum refresh time team trajectory.

V. Minimum Refresh Time Team Trajectory on a Cyclic Roadmap

In this section we propose an approximate solution to Problem 1 in the case of a cyclic roadmap. Let $G = (V, E)$, with $|V| = n$, be an undirected and connected roadmap. Note that there exists an open tour $\tau$ with $2(n-1)$ edges that visits all the vertices.\(^6\) We associate a chain roadmap $\Gamma$ with $\tau$, such that $\Gamma$ has $2n-1$ vertices and $2(n-1)$ edges, and such that the length of the $i$-th edge of $\Gamma$ equals the length of the $i$-th edge of $\tau$. Our first approximation method consists of computing Trajectory 1 for an optimal $m$-partition of $\Gamma$.

Theorem 5.1: Performance of the chain approximation strategy: Let $G$ be a connected roadmap, let $n$ be the number of vertices of $G$, and let $\delta$ be the ratio of the longest to the shortest length of the edges of $G$. Let $RT^*$ be the minimum refresh time on $G$. Let $\tau$ be an open tour with $2(n-1)$ edges that visits all the $n$ vertices, and let $\Gamma$ be the chain roadmap associated with $\tau$. Let $RT^*_\Gamma$ be the minimum refresh time on $\Gamma$. Then $RT^*_\Gamma \leq 8\delta RT^*$.

Proof: Let $w$ be the shortest length of the edges of $G$, and note that the length of $\Gamma$ is upper bounded by $2n(\delta w)$. It follows that $RT \leq \frac{4n|w|}{m}$. Since $m < n$ by assumption, some robots need to move along $G$ for all the viewpoints to be visited. Because each robot can visit only a vertex at a time, at least $\lceil \frac{n}{m} \rceil$ steps are needed to visit all the vertices of $G$, and therefore $RT^* \geq \frac{2n-1}{m} w \geq \frac{1}{\delta} \frac{n}{2} \frac{1}{m} w$. By taking the ratio of the two quantities we get $RT^*_\Gamma \leq 8\delta RT^*$.

In what follows, we describe a polynomial time constant factor approximation algorithm for the Team refresh time problem. Given a roadmap $G = (V, E)$ and a positive integer $k < |V|$, we define a path cover of cardinality $k$ as the collection of paths $\{p_1, \ldots, p_k\}$ such that $V \subseteq \bigcup_{i=1}^{k} p_i$. Let the cost of a path equal the sum of the length of its edges. The min-max path cover problem asks for a minimum cost path cover for the input graph, where the cost of a cover equals the maximum cost of a path in the cover.

Theorem 5.2 (Min-max path cover [4]): There exists a 4-approximation polynomial algorithm for the NP-hard min-max path cover problem.

A constant factor approximation algorithm for the NP-hard Problem 1 is described in the proof of the next theorem.

Lemma 5.3 (Constant factor approximation): There exists an 8-approximation polynomial algorithm for the NP-hard Problem 1.

Proof: Let $\{p_1, \ldots, p_m\}$ be a 4-approximation path cover of the graph $G$. Note that the length of each path is within $4RT^*$. Indeed, in a minimum refresh time team trajectory, every vertex is visited after at most $RT^*$ instants of time. Let $X$ the team trajectory obtained by letting each robot sweep at maximum speed a different path. Clearly, $RT(X) \leq 8RT^*$. Notice that, because of Theorem 5.2, the team trajectory $X$ can be computed in polynomial time.

VI. Conclusion

The problem of designing the trajectory of a team of robots to patrol an environment has been considered. With respect to the refresh time criterion, optimal team trajectories have been described. The computational complexity of the design problem has been characterized as a function of the environment topology and of the cardinality of the team to be employed. It is shown that the patrolling problem is generally NP-hard, and a polynomial time 8-approximation algorithm has been proposed. Particular instances have been identified for which an exact optimal solution can be computed efficiently.

REFERENCES


