Approximation Algorithms for Cooperative Multi-Robot Patrolling in Core-Periphery Graph Settings

Alex Bassot[1], Stefano Carpin[2], Nicola Basilico[1]

***Abstract*— The adoption of autonomous robots for patrolling introduces efficiency and resilience into automated surveillance systems. A fundamental challenge in this domain lies in optimal planning of patrol routes, a problem typically addressed through graph-based models of the environment. In this work, we consider core-periphery graph settings, where locations are divided into a high-priority core and a lower-priority periphery, a structure that naturally aligns with many real-world scenarios, such as urban surveillance and infrastructure security. We extend the Overlapping Partition Problem (OPP), a recently proposed formalization for core-periphery settings, providing novel theoretical insights by deriving approximation bounds under the assumption that the core is known, a realistic assumption in many surveillance contexts. Our theoretical contributions are complemented by empirical comparisons of our method with two state-of-the-art baselines, demonstrating our method's superior performance in computing effective patrolling strategies.**

I. Introduction

Patrolling and asset surveillance are repetitive and hazardous tasks where intelligent automation has gained prominence in recent years. The deployment of autonomous robots, including unmanned aerial and ground units, has increased significantly [1], [2], [3]. Robots offer clear advantages in terms of performance, resilience to failures, and the ability to improve efficiency through cooperation and shared workloads. Among the foundational problems of automated robotic surveillance systems, there is optimal planning of robot routes. This problem often relies on graph-based representations of the environment [4] where vertices represent the locations to monitor, and edges are associated with paths, enabling robots to move between locations while incurring a travel cost. Sometimes, vertices are assigned *values* to express the criticality of these locations in terms of the required protection. Optimally solving the patrolling problem involves finding a scheduling policy that minimizes a cost, typically associated with the maximum idleness on the graph. This means minimizing the maximum time that any vertex remains unguarded between successive visits by any robot.

In [5], we introduced a novel variant for this problem where instead of considering arbitrary graphs, we focused on what we called *core–periphery* graph settings. In such graphs, the vertices and thus their associated locations are divided into two groups: the *core* and the *periphery*. The core consists of vertices that are highly valuable and demand intensive surveillance, whereas the periphery includes vertices that are less critical and require milder surveillance needs than the core. We consider this feature to be relevant to many real–world surveillance applications where some areas demand significantly more attention than others. For example, in urban surveillance (see Figure 1) busy areas and main roads require continuous monitoring due to the increased risk of crime or congestion, whereas residential streets and parking lots demand less frequent patrols. Other examples include airports, where terminal areas, security checkpoints, and boarding gates constitute the core, while maintenance areas and parking lots are the periphery. Even in agricultural field monitoring, certain regions, such as high-yield crops and irrigation points, require frequent inspections to ensure productivity, whereas peripheral field edges and pathways do not. In these problem settings, the patrolling routes of the robots overlap in the core area and are separate in the periphery. In the core, robots coordinate to minimize idleness, whereas the periphery is partitioned in areas where each robot follows an independent route.

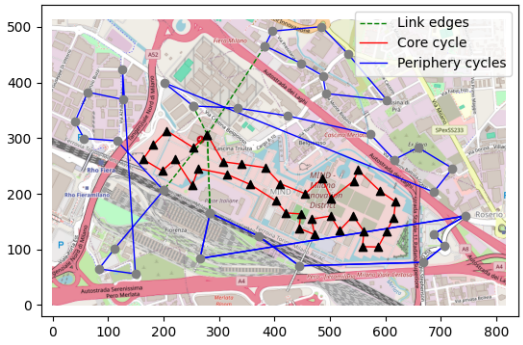


Fig. 1: Four robots patrolling a core-periphery setting. Triangles are vertices of the core, while circles are vertices of the periphery. Each robot alternates between a shared tour in the core (red) and a non-shared tour in a portion of the periphery (blue) connecting the two via a link edge (green).

The original problem formulation is called Overlapping Partition Problem (OPP) [5]. Solving this problem involves tackling an exact formulation, which is generally intractable because both the core and each robot's peripheral region need to be computed as part of the optimal solution. In this work, we propose several novel results for this representative class of patrolling problems. Specifically, we derive approximation bounds showing how solutions with bounded suboptimality

[1]A. Bassot and N. Basilico are with the Department of Computer Science, University of Milan, Milan, Italy alex.bassot@unimi.it, nicola.basilico@unimi.it

[2]S. Carpin is with the Department of Computer Science and Engineering, University of California, Merced, CA, USA. scarpin@ucmerced.edu

can be efficiently computed, in the case in which the core is given, an assumption realistic to many surveillance scenarios. Our theoretical analysis is supported by an approximation algorithm that, in addition to constructing bounded-suboptimal solutions, performs well in core-periphery settings when empirically compared to the two mainstream approaches from the literature. The first being based on a "divide and conquer" strategy to partition the environment into disjoint subregions, assigning each subregion to a robot [6], [7], [8], while the second approach allows all robots to visit any site of interest, avoiding partitioning and instead addressing a Traveling Salesman Problem (TSP) over the entire environment [9], [10]. Our findings offer key insights into the theoretical properties of the multirobot patrolling problem, and provide an effective method for scheduling surveillance units in various scenarios highly pertinent to real-world patrolling setups.

II. Related works

A common objective in optimal patrolling is minimization of the worst idleness in weighted graphs without node values. A key question addressed in this scope is whether cyclic strategies based on a TSP solution might be effective. The first answer was given by the seminal work of [11], where the author showed that the best single-agent cyclic strategy is a TSP solution. The work also extends to the multi-agent case, showing that placing m agents well spaced on the single agent TSP solution S will decrease idleness by a factor of $\frac{1}{m}$. The work also shows that the strategy of extending the single-agent case to the multi-agent one leads to a constant approximation factor for the worst idleness. To conclude, the strategy of adopting the TSP for the multi-robot patrolling problem is the best strategy if the maximum edge cost is "not too big". Otherwise, partition-based strategies should be taken into account.

Such a work shows the difficulties in achieving a clear distinction about which strategy is the best to adopt based on the topology of the environment. Building on this statement, it seems worthwhile to explore hybrid solutions that feature both cooperative and partitioned components, like the one we address in this work. The approximate solution of the TSP cycle used in [11] is based on the well-known work of Christofides [12], which constructed a $\frac{3}{2}$-approximation algorithm (known as the *Christofides Algorithm*) to find a TSP cycle on weighted metric graphs (the approximation has been slightly improved in [13]). On the other hand, approximation algorithms for partition strategies are widely studied in the robotic community, where these strategies assume different names. In [14] authors considered a range of different problems that involve the search of minimum and min-max m-covers of a graph: for example, a 4-approximation algorithm called Min-Max Path Cover is presented for the problem of finding m disjointed paths with min-max weight (a path here is intended as a sequence of vertices and edges where the edges are crossed only once), and a 3-approximation algorithm for the version with walks (intended as paths with possibly repeated edges), called Min-Max Postmen Cover, is derived. In [15] an approximation algorithm for Min-Max Rootless k-Cycle Cover Problem is presented: it finds k disjointed cycles that minimize the max-weighted cycle, with an approximation ratio of $\frac{16}{3}$ (the best known so far, according to [16]). Furthermore, in [17], a study on min-max latency of vertices is approached by the initial search for an optimal m-partition of a given path cover of the graph (created by doubling the edges of an MST). They provide an algorithm called *Optimal Left-induced m-partition* that outputs the optimal partition of a path into m subpaths, and results in a $(\frac{n-2}{n}8\frac{e_{max}}{e_{min}})$-approximation algorithm, where n is the number of vertices and e_{max} and e_{min} are the maximum and minimum edge lengths, respectively. A more general study involving the problem of finding a minimum latency *patrol schedule* is explored in [18]. Here, the focus is on *cyclic solutions*, which are partitions of the set of vertices into $l \leq m$ subsets, and to each of them are assigned m_i agents equally spaced in the TSP tour, with $\sum m_i = m$. The authors show that an optimal generic solution can be transformed into a cyclic one with an approximation ratio of $2(1 - \frac{1}{m})$. The generic problem is even more complicated, since extending the single-agent strategy to the multi-agent case will not give the result found in [11] of reducing the latency of the vertices, even if we equally space the agents on the tour; an example can be found in [19].

Another type of strategy that can be used to deal with the multi-robot patrolling problem is finding a "good" partition of the vertices and then applying some sort of approximation algorithm on the subgraphs. For example, in [20] is explored the case in which the graph is already partitioned into K clusters, and the goal is to find m tours with a common starting vertex, covering all vertices with min-max latency. It requires an approximation algorithm for the Rural Postman Problem (ρ_R ratio) and an approximation algorithm for the TSP Problem (ρ_P ratio), for a total ratio of $\rho_R+2\rho_P+1-\frac{1}{k}$.

In essence, TSP and multi-TSP solutions are central to our problem because the worst idleness of the vertices covered by the tour is at least the sum of the weights of the edges composing the tour, but cannot be directly applied to output hybrid solutions, as in our case. Also, our problem is more complicated since we involve valued vertices, and hence the function to minimize becomes dependent on the vertices' importance, as well as the length of the tour.

For the general min-max weighted latency problem, in [21] authors show cases where adopting a TSP cycle is far from the best solution, even for the single-agent case, underlining the fact that dealing with vertex values can really modify the min-max weighted latency problem. In the end, they also provide an $O(\log \frac{v_{max}}{v_{min}})$-approximation algorithm and an $O(\log n)$-approximation algorithm for the problem in single-agent settings. Another interesting work on patrolling over a graph with valued vertices can be found in [22], where they consider the multi-agent case of the patrol-scheduling problem, and create an approximation algorithm for the min-max weighted latency problem with k robots of factor $O(k^2 log \frac{v_{max}}{v_{min}})$, exploiting the results of [15]. This was then improved in [23], where the authors provided

an approximation algorithm of factor $O(klog\frac{v_{max}}{v_{min}})$ for the same problem. However, these types of solution have a prohibitive approximation ratio and require online procedures for scheduling the next vertex to be patrolled.

III. Problem formulation

We assume to represent the environment as a weighted complete graph $G := (V, E, c, v)$, with a set of vertices $V := \{1, 2, \ldots, n\}$, an edge set $E := V^2$, a traveling cost function $c : E \to \mathbb{R}^+$, and a node value function $v : V \to \mathbb{R}^+$. Given a team of robots $R := \{1, 2, \ldots, m\}$ moving over the graph, the traveling cost $c(e)$, with $e := (i, j)$, can be interpreted as the time needed by any robot for traveling from vertex i to vertex j. The costs returned by c satisfy the triangle inequality and, given the undirected nature of the graph, $\forall i, j \in V, c(i, j) = c(j, i)$.

The problem we aim at solving is to compute a joint cyclic patrolling route for the team of robots. Formally, we indicate a solution as $\pi := (\pi_1, \pi_2, \ldots, \pi_m)$, where each π_r is, in general, a cyclic walk over a subset of V. The concurrent execution of the paths in π by the team of robots induces over each vertex i a *maximum idleness* $I^\pi(i)$, defined as the maximum temporal delay between two successive visits (not necessarily by the same robot) to i. To be admissible, a solution π must induce a finite $I^\pi(i)$ on each vertex i, which amounts to guarantee that each vertex is patrolled by at least one robot. To be optimal, a solution π^* must achieve the minimum maximum weighted idleness, that is

$$w(\pi^*) := \min_{\pi} \max_{i \in V} v(i) I^\pi(i) \tag{1}$$

Solving the above problem is generally NP-hard and a typical approach to deal with it is to seek for heuristic or approximated solutions by restricting the search space with additional constraints. A popular approach, drawing from a *divide et impera* rationale, is that of requiring the solution to be structured as m non-intersecting cycles over a partition of V. This simplification allows to avoid embedding the robots' coordination dynamics into the problem's model. Indeed, coordination poses additional difficulties when patrolling routes overlap in specific areas of the environment, since the resulting maximum idleness on vertices shared by two or more robots can exhibit complex patterns that are difficult to express and optimize. However, sharing and coordination of areas can lead to more effective patrolling strategies.

The model we proposed in [5], referred to as Overlapping Partition Problem (OPP), adopts an approach that is more general with respect to distributing robots over a partition of the environment. Specifically, we try to maintain the *divide et impera* rationale while allowing a level of constrained coordinated patrolling building upon the idea of having two types of vertices: the *core vertices* and the *periphery vertices*. The guiding principle is that, while each robot r is still assigned a subset of vertices $\bar{V}_r$ and the union of all the assigned regions equals V, their intersection $V_0 := \cap_{r \in R} \bar{V}_r$ is non-empty. The set V_0 is called core, and represents a region of the environment where all robots share and coordinate their efforts. For each robot r the set $V_r := \bar{V}_r \setminus V_0$ is called the periphery of r, corresponding to those vertices that fall under robot r's exclusive responsibility (notice that $\{V_0, V_1, \ldots, V_m\}$ is still a partition of V, but without a one-to-one correspondence with robots). This vertex clustering scheme allows to increase efforts (lowering the idleness) via cooperation in regions of the environment (the core) that, in an optimized solution, would ideally tend to be the most important. The proposed model obtains such a clustering of vertices by imposing specific constraints for π to be admissible.

Specifically, each π_r is required to be a cycle over $V_r \cup V_0$, defined as $\pi_r := (\pi_{V_r}, \pi_{V_0})$. We denote π_Q, with $Q \subseteq V$, the part of the solution that covers only vertices in Q. Thus, π_{V_r} is a path over V_r, starting and ending at vertices $s_r, t_r \in V_r$, respectively. Similarly, π_{V_0} is a path over V_0, starting and ending at $s_0, t_0 \in V_0$. These points are also referred to as the core's entrance and exit, respectively. The cycle is then implicitly completed by traveling on the edge (t_0, s_r). Notice that, according to the above requirements, V_0 and π_{V_0} should be the same for each robot. See Figure 2 for a visual intuition in a setting with two robots.

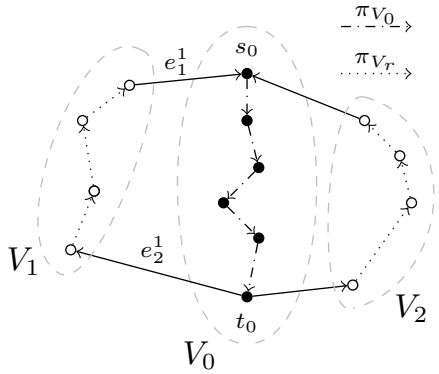


Fig. 2: Abstract visualization of a 2-robot OPP solution.

Similarly to the canonical case, the optimal joint patrolling route π^*, and the corresponding partition $\{V_0^*, V_1^*, \ldots, V_m^*\}$ induced by it, must satisfy the above constraints and its cost must achieve the minimum maximum weighted idleness. However, under the scheme we described above, (1) is rewritten to account for the synergies among the robots over the shared area represented by V_0:

$$w(\pi^*) = \min_{\pi} \max_{r \in R} \left\{ Ac(\pi_r), A^0 \frac{c(\pi_r)}{m} \right\} \tag{2}$$

where, $A := \max_{i \in V \setminus V_0} v(i)$, $A^0 := \max_{i \in V_0} v(i)$, and, with a slight overload of notation, $c(\pi) := \sum_{e \in \pi} c(e)$.

A. Model's assumptions

Equation 2 computes the maximum idleness of a solution π that complies with the constraints adopted in the model, which also account for the cooperation between the robots in the core V_0. To do so, it makes two key assumptions.

The first is a simplification of the structure of the patrolling route. In fact, to avoid complicated strategies, the work of [5] excludes all solutions in which a robot follows a tour that

alternates visits to the core and periphery. The only admitted solutions are those where each tour has the structure depicted in Fig. 2, namely (i) visit all vertices in the core, (ii) visit all the vertices in the periphery, repeat from (i).

The second assumption is about the *execution* of the patrolling routes. Despite in π^* different robots can be assigned cycles with different costs, the objective function assumes that *all robots will incur the same cost at execution time*, equivalent to the cost of the longest single-robot route. In the first term of the max operator, which computes the maximum weighted idleness outside the core, this is expressed by defining A as the maximum value outside V_0, regardless of the periphery V_r in which it happens to be. This implies that the maximum idleness on the graph (induced by the highest-cost route) will be multiplied by the highest value anywhere outside the core. The second term computes the maximum weighted idleness inside the core V_0. Similarly to the previous case, the maximum idleness will now be multiplied by the highest value in V_0. A key element is that, in such a term, the idleness is scaled by a factor m, representing a benefit derived from overlapping patrolling routes. According to the well-established results in [11], the idleness scaling is attainable only when the robots' visits to the vertices in V_0 are equally spaced in time. The underlying assumption in (2) ensures that this uniform temporal displacement can be achieved at execution time, by coordinating the robots to enter the core via the same vertex every $\max_{r\in R}\left\{\frac{c(\pi_r)}{m}\right\}$ time. This assumption of having a single entry/exit vertex might deteriorate the solution's quality by over-constraining it. However, it plays a key role in allowing robots to properly coordinate.

IV. Approximation analysis and algorithms

Consider an optimal solution to the problem, $\pi^* = \{\pi_1^*, \ldots, \pi_m^*\}$ and call $opt = \max_r c(\pi_r^*)$. Given the constraints on the structure of each π_r^* we introduced, we can express $c(\pi_r^*)$ with a sum of different terms[1]:

$$c(\pi_r^*) = \pi^*_{V_r^*} + \pi^*_{V_0^*} + e_1^r + e_2^r$$

The terms e_1^r and e_2^r are the costs of the edges that connect, in the robot r's cycle, the optimal solution's periphery V_r^* and core V_0^*. Without loss of generality, we assume that $e_1^r \leq e_2^r$. If the robots can coordinate themselves in order to maintain the scaling factor in the core, we can rewrite the objective function in the following way:

$$\begin{aligned} w(\pi^*) = \min_{\pi}\max_{r\in R} w(\pi) = \\ \min_{\pi}\max_{r\in R}\left\{Ac(\pi_r), A^0\frac{c(\pi_r)}{m}\right\} = \max\left\{A, \frac{A^0}{m}\right\}opt, \end{aligned} \tag{3}$$

where, in the right–hand term, A and A^0 are induced by the optimal partition, that is $A = \max_{i\in V\setminus V_0^*} v(i)$ and $A^0 = \max_{i\in V_0^*} v(i)$. We will focus on approximating opt.

[1]We overload the notation to indicate with the same symbol an element and its cost; depending on the context, p will be used to represent both a path p and its cost $c(p)$.

A. Given partition

We consider a setting where the partition $\{V_0^*, V_1^*, \ldots, V_m^*\}$ is given. We will relax this assumption later. Let us define opt_Q as the cost of the optimal Hamiltonian cycle computed over a subset of vertices $Q \subseteq V$ (that is, the cost of a TSP solution over Q, for which exact and, more importantly, well–known approximation methods are available).

Algorithm 1: ConnectCycles $O(mn^2)$

Input : A set of $m+1$ disjoint cycles $\{C_{V_0}, \ldots, C_{V_m}\}$ over a partition of G, where V_0 is the core.
Output: A set of paths $\pi = \{\pi_1, \ldots, \pi_m\}$.
for $r \in R$ **do**
  $e_r \leftarrow \arg\min_{i\in V_r, j\in V_0} c(i,j)$
end
$\bar{r} \leftarrow \arg\max_{r\in R}\{C_{V_r} + e_r\}$;
$(i_{\bar{r}}, u) \leftarrow e_{\bar{r}}$
for $r \in R$ **do**
  $v_r \leftarrow \arg\min_{i\in V_r} c(i,u)$;
  $s_r \leftarrow (v_r, u)$;
  $\pi_r \leftarrow \texttt{concat}(C_{V_0}, s_r, C_{V_r}, s_r)$
end
return $\{\pi_1, \ldots, \pi_m\}$;

Consider the solution obtained by applying Algorithm 1 to the $m+1$ optimal cycles (TSPs) computed for partition $(V_0^*, \ldots, V_m^*)$. Each robot r follows this strategy:

(i) follow the cycle of length $opt_{V_r^*}$, starting and ending at vertex v_r;
(ii) travel to the core via edge $s_r = (v_r, u)$;
(iii) follow the cycle of length $opt_{V_0^*}$, starting and ending at vertex u;
(iv) return to V_r^* via the same edge s_r and repeat from (i).

We first show two approximation lemmas related to a single robot's tour.

Lemma 1: $\forall r \in R$, $opt_{V_0^*} + opt_{V_r^*} + 2e_r \leq 2c(\pi_r^*)$

Proof: $opt_{V_0^*} \leq \pi^*_{V_0^*} + a_0$ and $opt_{V_r^*} \leq \pi^*_{V_r^*} + a_r$ where a_0 and a_r are the costs of the edges that connect the first and last vertices of $\pi^*_{V_0^*}$ and $\pi^*_{V_r^*}$ respectively. Recall that e_r is the minimum–cost edge between V_0^* and V_r^*. Then,

$$\begin{aligned} opt_{V_0^*} + opt_{V_r^*} + 2e_r &\leq opt_{V_0^*} + opt_{V_r^*} + 2e_1^r \\ &\leq \pi^*_{V_0^*} + a_0 + \pi^*_{V_r^*} + a_r + e_1^r + e_2^r \\ &\leq c(\pi_r^*) + a_0 + a_r \end{aligned}$$

The Lemma follows from the triangular inequality that guarantees that $a_0 \leq \pi^*_{V_0^*}$ and $a_r \leq \pi^*_{V_r^*}$. ∎

Lemma 2: $\forall r \in R$, $opt_{V_0^*} + opt_{V_r^*} + 2s_r \leq 2c(\pi_r^*) + opt_{V_0^*}$

Proof: From the triangular inequality we have $s_r \leq e_r + \omega_r$ where ω_r is the shortest sub-path of $opt_{V_0^*}$ connecting u_r and u_1. Notice that since $s_1 = e_1$, $\omega_1 = 0$. From the fact that $\omega_r \leq \frac{opt_{V_0^*}}{2}$ and Lemma 1 we have that

$$\begin{aligned} opt_{V_0^*} + opt_{V_r^*} + 2s_r &\leq opt_{V_0^*} + opt_{V_r^*} + 2(e_r + \frac{opt_{V_0^*}}{2}) \\ &\leq 2c(\pi_r^*) + opt_{V_0}, \end{aligned}$$

which prove the Lemma. ∎

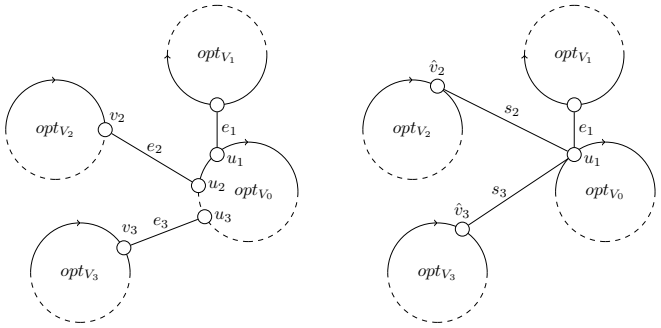
Lemma 1 states that, for every robot, adopting the TSP independently on the core and the periphery implies an optimality loss bounded by a constant factor (see Figure 3a). Lemma 2 derives a bound for the construction where the edges that link the core to the peripheries have in common the same core–vertex u_1 (see Figure 3b). We can leverage this result to derive an approximation bound for Algorithm 1.

Theorem 1: Given the Overlapping Partition Problem with m robots on a complete metric graph $G = (V, E, c, v)$, suppose that the robots can coordinate themselves by slowing down their speed until they match the speed of the slowest one, in order to be equally distanced in the core. Suppose that the optimal partition $\{V_0^*, V_1^*, \ldots, V_m^*\}$ of V is given. Then, there exists a 4.5–approximation algorithm that computes a solution in $O(n^3)$ time.

Proof: Suppose that $\pi^* = (\pi_1^*, \pi_2^*, \ldots, \pi_m^*)$ is the optimal solution for OPP, with partition $\{V_0^*, V_1^*, \ldots V_m^*\}$. By Equation 3, $w(\pi^*) = \max\left\{A, \frac{A^0}{m}\right\} opt$, where $A = \max_{i \in V \setminus V_0^*} v(i)$, $A^0 = \max_{i \in V_0^*} v(i)$ and $opt := \max_r c(\pi_r^*)$. From Lemma 2 we have $opt_{V_0^*} + opt_{V_r^*} + 2s_r$, for all robots. Since $opt_{V_0} \leq \pi^*_{V_0} + a_0 \leq \pi^*_{V_0} + \pi^*_{V_r} + e_1^{(r)} + e_2^{(r)} = c(\pi_r^*)$, it follows that $opt_{V_0^*} + opt_{V_r^*} + 2s_r \leq 3opt$. Using Christofides algorithm, we create cycles $C_{V_0^*}, C_{V_1^*}, \ldots, C_{V_m^*}$, such that $C_{V_i^*} \leq \frac{3}{2} opt_{V_i^*}$. Hence,

$$\max_{r \in R} \{C_{V_0^*} + C_{V_r^*} + 2s_r\} \leq 4.5opt. \tag{4}$$

The time complexity of this approximation algorithm is dominated by Christofides algorithm, that is, $O(n^3)$ [12]. ■



(a) Connecting via the minimum cost periphery–core edge for all robots.

(b) Connecting via the minimum cost periphery–core edge incident to u_1.

Fig. 3: Comparison of connection strategies

B. Given core

We extend now the results provided by Theorem 1 by removing the key assumption of knowing the best partition of V. Thus, we set our problem in the case where the only known subset of vertices is the core $V_0^* \subseteq V$.

Theorem 2: Given the Overlapping Partition Problem with m agents on a complete metric graph $G = (V, E, c, v)$, suppose that the robots can coordinate themselves by slowing down their speed until they match the speed of the slowest one, in order to be equally distanced in the core. Suppose that the core subset $V_0^* \subseteq V$ is given. Then, there exists an approximation algorithm that creates a solution $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ for OPP for which

$$w(\pi) \leq 2\rho \cdot w(\pi^*),$$

where ρ is the approximation factor of an algorithm for the min-max Rootless m-Cycle Cover Problem (RCCP).

Proof: Suppose that $\pi^* = (\pi_1^*, \pi_2^*, \ldots, \pi_m^*)$ is the optimal solution for OPP, with vertices' partition $\{V_0^*, V_1^*, \ldots V_m^*\}$. By Equation 3, $w(\pi^*) = \max\left\{A, \frac{A^0}{m}\right\} opt$, where $A = \max_{i \in V \setminus V_0^*} v(i)$, $A^0 = \max_{i \in V_0^*} v(i)$ and $opt := \max_r c(\pi_r^*)$.

Given that we do not know the best partition of the periphery $V \setminus V_0^*$, we construct the m periphery–cycles by applying a ρ–approximation algorithm for the m–rootless cycle cover problem [15]. The algorithm outputs a set of m disjoint cycles $C_{V_1}, \ldots, C_{V_m}$ over the periphery $V \setminus V_0^* = V_1 \cup \ldots \cup V_m$. The TSP cycle over V_0^* is approximated by Christofides, i.e., $C_{V_0^*} \leq \frac{3}{2} opt_{V_0^*}$. Now, consider the optimal solution opt_{cover} for the m–cycle cover problem of $V \setminus V_0^*$ and the cycle cover composed by the set $\{opt_{V_1^*}, \ldots, opt_{V_m^*}\}$. By definition, $opt_{cover} \leq \max_r \{opt_{V_r^*}\}$, and hence

$$\max_{r \in R} \left\{C_{V_r}\right\} \leq \rho \max_{r \in R} \left\{opt_{V_r^*}\right\}.$$

It follows that

$$C_{V_0^*} + \max_{r \in R} \left\{C_{V_r}\right\} + 2s_r \leq \tag{5}$$

$$\frac{3}{2} opt_{V_0^*} + \rho \max_{r \in R} \left\{opt_{V_r^*}\right\} + 2s_r \leq \tag{6}$$

$$\frac{3}{2} opt_{V_0^*} + (\rho - 1) \max_{r \in R} \left\{opt_{V_r^*}\right\} + 2opt \leq \tag{7}$$

$$(\rho - \frac{5}{2}) \max_{r \in R} \left\{opt_{V_r^*}\right\} + 5opt \leq \tag{8}$$

$$(2\rho - 5)opt + 5opt = 2\rho \cdot opt, \tag{9}$$

where (6)-(7) follows from Lemma 2, (7)-(8) follows from Lemma 1 and (8)-(9) again from Lemma 2.

The time complexity depends on the approximation algorithm used to construct the cycle cover of $V \setminus V_0^*$. ■

Algorithm 2: ComputeSolution

Input : Graph $G = (V, E, c, v)$, where V_0^* is the core.
Output: A set of paths $\pi = \{\pi_1, \ldots, \pi_m\}$.
$C_{V_0^*} \leftarrow$ TSP(V_0^*); `// Christofides [12]`
if $V \setminus V_0^*$ *is partitioned into* $\{V_1, \ldots, V_m\}$ **then**
  for $r \in \{1, \ldots, m\}$ **do**
    $C_{V_r} \leftarrow$ TSP(V_r)
  end
else
  $\{C_{V_1}, \ldots, C_{V_m}\} \leftarrow$ RCCP($V \setminus V_0^*, m$); `// Xu [15]`
end
return `ConnectCycles`$(C_{V_0^*}, C_{V_1}, \ldots, C_{V_m})$

The last theorem is proved using the ratio ρ of an hypothetical approximation algorithm. Thus, we can apply the results in [15] to find an approximated m-cycle cover of the graph with approximation factor $\frac{16}{3}$ to achieve a $\frac{32}{3}$–approximation and a time complexity of $O((n^2m^2 + m^5) \log n)$, but the

choice of using this specific algorithm is not restrictive; in other words, one can apply any algorithm that finds an optimal cycle cover with an approximation ratio ρ and the theorem still holds. Algorithm 2 reports the pseudocode of the complete method.

V. Experimental Evaluation

Algorithm 2 is significant as it helps us to determine an approximation factor for the problem. In this section, we aim to evaluate how tight its approximation bound is on average. This means assessing whether the algorithm not only has theoretical value but also practical applicability, providing good solutions to real-world problem instances. Notice that approximation algorithms have theoretical importance but might lack practical utility, as they might be outperformed by heuristics that, despite lacking worst-case guarantees, result in a better performance on the average case.

To this end, we compare the solutions obtained using our method (which we label OPP) with two baselines, obtained by applying two widely used approaches. The first is based on the TSP, where robots follow the same minimum-cost Hamiltonian cycle (π_{TSP}) while maintaining equal spacing along it. The second approach employs a "divide-and-conquer" strategy that partitions the environment, assigning each robot its own patrol route (π_r) over a partition element. This second baseline is computed by solving an RCCP instance on the whole graph. As evaluation metric, we report the worst (max) weighted idleness as defined in Equation 1. As we discussed above, for our method, its definition reduces to that of Equation 2. Clearly, for the TSP baseline, it reduces to $w_{TSP} = \frac{A}{m} c(\pi_{TSP})$ while for the partition-based one, it becomes $w_P = \max_{r \in R}\{A_r c(\pi_r)\}$.

In the first batch of experiments we compared the three methods on an extensive set of randomly generated instances. We considered teams of m robots with $m \in \{3, 4, \ldots, 10\}$ and graphs with $n = |V| \in \{20, 40, 60, 80\}$. For each pair (m, n) we average over 50 random graphs where vertices are uniformly and independently sampled from the plane $[0, 100]^2 \subset \mathbb{R}^2$. Edge sets are complete, with cost equal to the Euclidean distance between the corresponding vertices.

	20 vertices	40 vertices	60 vertices	80 vertices
$T = 0.02$	16.5	26.68	37.38	38.54
$T = 0.1$	8.94	9.72	13.62	9.96
$T = 0.4$	3.26	3.4	5.76	2.86
$T = 0.8$	2.12	2.88	4.46	2.1

TABLE I: Average number of vertices in the core V_0 for each threshold T

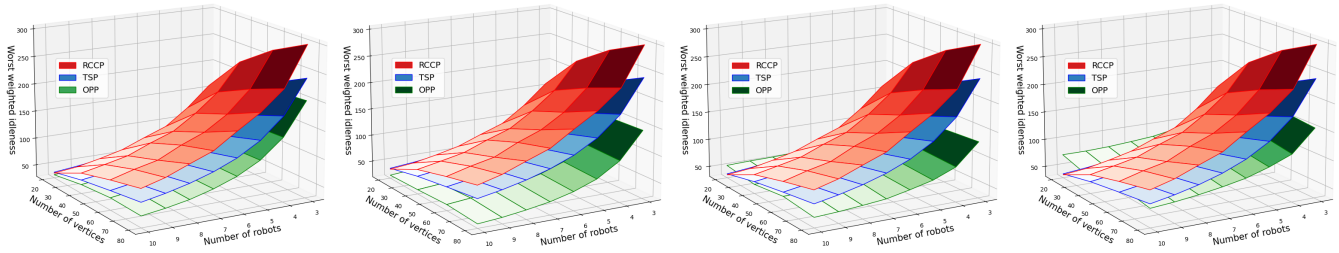
To obtain a core-periphery profile of the vertices, we generate a set of values and then we apply a thresholding. Specifically, the value function $v : V \subset [0, 100]^2 \to \mathbb{R}$ that we adopt is the *inverse sphere distance* (ISD), centered in the mean point of the vertices. Formally

$$v(x, y) := \begin{cases} 100 & \text{if } (x, y) = (x_M, y_M) \\ \frac{1}{(x - x_M)^2 + (y - y_M)^2} & \text{otherwise} \end{cases}$$

with $(x_M, y_M) := \frac{1}{|V|} \sum_{(x,y) \in V} (x, y)$. The values are normalized to a range of $[0, 1]$ by dividing them by the maximum vertex value obtained. The sphere distance, known also as the first De Jong's function, is a well-known benchmark for evaluating optimization algorithms and effectively allows one to model scenarios where key patrol sites are near the environment's center. As a final step, the vertices with values exceeding a certain threshold T are marked as part of the core: $(x, y) \in V_0$ if and only if $v(x, y) > T$. In our experiments, we considered $T \in \{0.02, 0.1, 0.4, 0.8\}$ to have cores of different cardinalities. Table I reports the average core sizes obtained in our runs while Figure 5 provides a visual example of the OPP strategy on the same instance when applying different threshold values. Our method and the baselines are implemented in Python and exploit the VRP solver provided by OR-tools [24] (notice that, especially in the multi-robot case, the solver returns a suboptimal solution which, in principle, could be improved with additional heuristic methods).

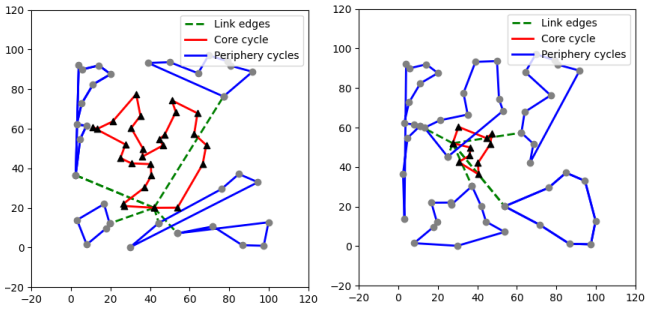
Figure 4 shows the results obtained. The first trend that can be observed is that the differences between the strategies fade out as the number of vertices becomes small and the number of agents becomes large. This seems reasonable since in instances with such a profile the abundance of patrolling units compared with the number of locations to be patrolled enables multiple and easy ways to obtain low idleness. The RCCP method turned out to be worse on average than the one based on the TSP. Ideally, both strategies aim to achieve a scaling factor of m in the worst idleness. The TSP method accomplishes this by definition, since robots are equally spaced while covering the cycle. The RCCP method, on the other hand, should ideally create m cycles of equal length, approximately $1/m$ of the TSP cycle. However, this is rarely the case because of the environment's topology and value distribution. Consequently, the TSP-based solution results in lower idleness. This finding aligns with the intuition provided in [11], suggesting that in graphs where the variance in edge costs is not very large (as in our case), TSP-based strategies tend to outperform partition-based ones. This result also suggests that, in these settings, robot cooperation generally outperforms "divide and conquer". The observed performance of the OPP method, which mostly achieves lower idleness compared to the two baselines, supports this insight. OPP might be seen as a "divide and cooperate" strategy that is beneficial in core-periphery scenarios. However, when the core size decreases, OPP is outperformed in instances with few vertices and many robots, as these instances are less challenging and only barely align with the core-periphery profile.

We replicate the comparison described above on the same instances but now considering a heuristic method to include vertices in the core that is defined with the goal of seeking an advantageous separation between core and periphery vertices considering the vertex values and the number of available robots. This method, proposed in [5] and called *Balanced Weights Heuristic* (BWH) composes the core as $V_0 = \{i \in V | v(i) > \frac{v_{max}}{m}\}$. As can be seen in Figure 6a, the results

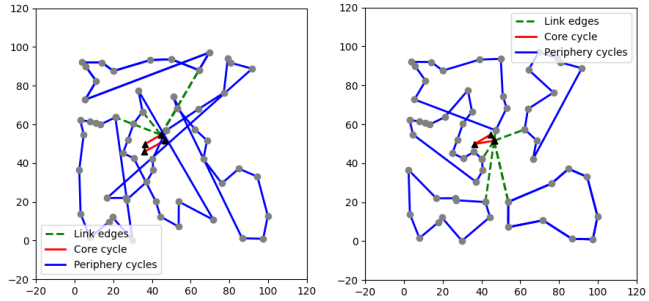


(a) $T = 0.02$ (b) $T = 0.1$ (c) $T = 0.4$ (d) $T = 0.8$

Fig. 4: Worst weighted idleness comparisons on random graph instances.

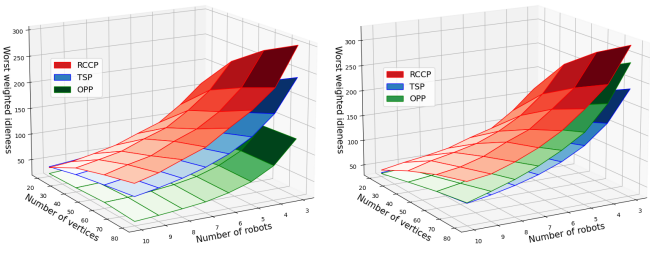


(a) $T = 0.02$, $\omega = 104.39$ (b) $T = 0.1$, $\omega = 61.24$



(c) $T = 0.4$, $\omega = 106.67$ (d) $T = 0.8$, $\omega = 120.05$

Fig. 5: OPP strategy with cores of different size obtained with different values of the threshold T. We also report the worst weighted idleness ω.



(a) ISD values (b) Random values

Fig. 6: Worst weighted idleness comparisons with core selection based on balanced weights heuristic (BWH).

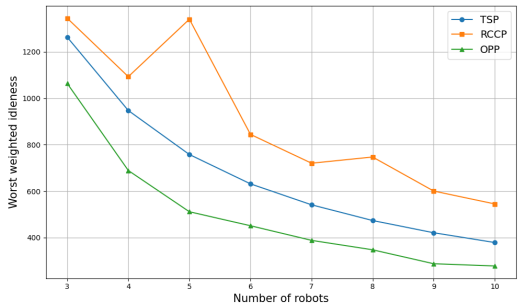


Fig. 7: Worst weighted idleness for different number of robots on a real use case.

obtained mirror those with fixed thresholding (Figure 4) confirming how OPP is robust with respect to the core selection method. Figure 6b shows the results obtained from the random graphs using BWH core selection, where the values have been drawn uniformly from $[0, 1]$. These instances are expected not to align well with the core-periphery scheme, as the core will likely not exhibit any centrality in the graph layout. As expected, OPP does not achieve the best performance here. Although its approximation guarantees remain valid, the TSP-based strategy performs better on average. However, the performance gap with OPP is not significant, and our method still outperforms RCCP, making it a viable option even in these challenging scenarios.

To conclude our evaluation, we tested our method on a problem instance derived from a real-world use case. We envision an urban surveillance scenario conducted with UAVs in an area of the city of Milan that hosted the 2015 Expo and which today is the site of the Milan Innovation District (MIND), an area that will feature a new campus for the University of Milan. We manually composed the core by placing vertices on the streets and open areas in the central regions of the district, which are likely to be the most crowded during events or peak hours. The outer regions, including highways, closeby stations, and parking lots, have been included in the periphery. The value distribution is still the ISD, centered at the mean point of coordinates $(x_M, y_m) = (402.29, 240.08)$. Figure 7 shows a qualitative comparison of the three strategies with 10 robots for which OPP confirms as the method that achieves the lowest maximum idleness. The performance trend in this case

(a) RCCP, $\omega = 544.96$ (b) TSP, $\omega = 378.86$ (c) OPP, $\omega = 277.91$

Fig. 8: Patrolling strategies for a team of 10 robots on a real use case.

with a different number of robots can be seen in Figure 7.

VI. Conclusions and Future Directions

This work presents the first approximation algorithm for the *Overlapping Partition Problem*, a novel formulation of multi-robot patrolling on core-periphery graphs. We provide theoretical analysis, experimental evaluation, and a bounded-suboptimal algorithm with strong empirical performance.

The future directions identified during this study will focus on reducing the prior assumptions taken. Future work should be done on improving the approximation bounds and on the research of a more general approximation that does not require the core to be given, while trying at the same time to keep a reasonable performance during the execution. The second assumption that aims to be removed is about the coordination of the robots. The current mechanism prescribes that robots must, at times, wait while covering their path. One interesting direction for future work is to devise online algorithms that might exploit such temporal budget to further improve the solution quality.

References

[1] N. Basilico, "Recent trends in robotic patrolling," *Current Robotics Reports*, vol. 3, no. 2, pp. 65–76, 2022.

[2] M. Tambe, *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[3] D. Portugal and R. Rocha, "A survey on multi-robot patrolling algorithms," in *Technological Innovation for Sustainability*, 2011, pp. 139–146.

[4] A. Machado, G. Ramalho, J.-D. Zucker, and A. Drogoul, "Multi-agent patrolling: An empirical analysis of alternative architectures," in *International workshop on multi-agent systems and agent-based simulation*. Springer, 2002, pp. 155–170.

[5] C. D. Alvarenga, N. Basilico, and S. Carpin, "Combining coordination and independent coverage in multirobot graph patrolling," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4413–4419.

[6] Y. Elmaliach, N. Agmon, and G. Kaminka, "Multi-robot area patrol under frequency constraints," *Annals of Mathematics and Artificial Intelligence*, vol. 57, no. 3-4, pp. 293–320, 2009.

[7] J. Palacios-Gasós, D. Tardioli, E. Montijano, and C. Sagüés, "Equitable persistent coverage of non-convex environments with graph-based planning," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1674–1694, 2019.

[8] D. Portugal, C. Pippin, R. Rocha, and H. Christensen, "Finding optimal routes for multi-robot patrolling in generic graphs," in *Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS)*, 2014, pp. 363–369.

[9] C. Diaz Alvarenga, N. Basilico, and S. Carpin, "Multirobot patrolling against adaptive opponents with limited information," in *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, 2020, pp. 2486–2492.

[10] Y. Oshart, N. Agmon, and S. Kraus, "Non-uniform policies for multi-robot asymmetric perimeter patrol in adversarial domains," in *Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems*, 2019, pp. 136–138.

[11] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, 2004, pp. 302–308.

[12] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," in *Operations Research Forum*, vol. 3, no. 1. Springer, 2022, p. 20.

[13] A. R. Karlin, N. Klein, and S. O. Gharan, "A (slightly) improved approximation algorithm for metric tsp," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 32–45.

[14] E. M. Arkin, R. Hassin, and A. Levin, "Approximations for minimum and min-max vehicle routing problems," *Journal of Algorithms*, vol. 59, no. 1, pp. 1–18, 2006.

[15] W. Xu, W. Liang, and X. Lin, "Approximation algorithms for min-max cycle cover problems," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 600–613, 2013.

[16] S. Schwartz, "An overview of graph covering and partitioning," *Discrete Mathematics*, vol. 345, no. 8, p. 112884, 2022.

[17] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 592–606, 2012.

[18] P. Afshani, M. de Berg, K. Buchin, J. Gao, M. Löffler, A. Nayyeri, B. Raichel, R. Sarkar, H. Wang, and H. T. Yang, "On cyclic solutions to the min-max latency multi-robot patrolling problem," in *38th International Symposium on Computational Geometry, SoCG 2022*, 2022.

[19] A. B. Asghar, S. Sundaram, and S. L. Smith, "Multi-robot persistent monitoring: minimizing latency and number of robots with recharging constraints," *IEEE Transactions on Robotics*, vol. 41, pp. 236–252, 2025.

[20] X. Bao, L. Xu, W. Yu, and W. Song, "Approximation algorithms for the min-max clustered k-traveling salesmen problems," *Theoretical Computer Science*, vol. 933, pp. 60–66, 2022.

[21] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014.

[22] P. Afshani, M. De Berg, K. Buchin, J. Gao, M. Löffler, A. Nayyeri, B. Raichel, R. Sarkar, H. Wang, and H.-T. Yang, "Approximation algorithms for multi-robot patrol-scheduling with min-max latency," in *Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2021, pp. 107–123.

[23] L.-H. Chen, L.-J. Hung, and R. Klasing, "Improved approximation algorithms for patrol-scheduling with min-max latency using multi-class minimum spanning forests," in *Proceedings of the International Conference on Algorithmic Aspects in Information and Management*. Springer, 2024, pp. 99–110.

[24] V. Furnon and L. Perron, "Or-tools routing library," Google. [Online]. Available: https://developers.google.com/optimization/routing/