2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Combining Coordination and Independent Coverage in MultiRobot Graph Patrolling

Carlos Diaz Alvarenga Nicola Basilico Stefano Carpin

***Abstract*— Graph patrolling algorithms provide effective strategies for coordinating mobile robots in the context of autonomously surveilling valuable assets. Optimizing patrolling strategies often aims to minimize the time between subsequent visits to a vertex, a measure known in the literature as *idleness*. In the domain of multi-robot patrolling, two approaches have received the most attention so far. The first involves coordinating all robots to follow a shared patrolling strategy covering the entire graph, while the second approach partitions the environment into disjoint areas that are then assigned to individual robots. Starting from these existing solutions, this paper introduces a new method that bridges these two complementary approaches. Our technique splits the vertices of the graph into a partition that includes a shared portion of the environment patrolled collectively by all robots, along with disjoint areas allocated exclusively to individual robots. This problem is formulated in terms of minimizing the maximum weighted idleness of the graph and is shown to be NP-hard. We then describe an exact solution for the problem and propose various heuristics to efficiently compute solutions for large problem instances. We evaluate and compare the proposed techniques in simulation and demonstrate that, in most cases, our methods produce better patrolling strategies when compared to classic solutions. Moreover, for small problem instances where the exact solution can be found, we show that our proposed heuristic has a competitive performance ratio.**

I. Introduction and Related Work

In recent years, algorithms for graph patrolling, especially in autonomous surveillance using mobile robots, have received great attention [4]. This involves modeling environments as graphs and devising strategies to guide patrollers across vertices, mirroring real-world scenarios [20] where vertices represent locations of interest, patrollers are robots with surveillance capabilities, and visits entail checking locations for threats and taking action if needed.

Computing a patrolling strategy is usually addressed as an optimization problem defined by constraints and costs. Typical goals include following the graph's topology, ensuring a minimum frequency of visits on vertices [10], or complying with latency constraints [3]. Cumulative costs are normally associated with traveling along edges and, in some cases, with vertex visits as well. Different approaches have been applied to solve these problems. These range from the minimization of selected optimization criteria [1], [12] to using game-theoretical frameworks where the behavior of a rational adversary is considered [6], [9].

N. Basilico is with the Department of Computer Science, University of Milan, Milan, Italy. C. Diaz Alvarenga and S. Carpin are with the Department of Computer Science and Engineering, University of California, Merced, CA, USA.

Our work belongs to the first group of methods as we adopt an optimization criterion that has been extensively studied in the literature: the *idleness* between visits [7], [19]. This metric measures the temporal difference between subsequent visits (attack-clearing actions) to a vertex. By measuring the idleness in each vertex of the graph and computing an aggregate function of the obtained values (maximum, average, or variants) it is possible to assess how well a strategy protects an environment. The rationale is that the lower the idleness of a vertex, the less likely that vertex is to be subject to an attack. The idleness of a vertex is often scaled by the vertex's importance obtaining what is often referred to as the *weighted* idleness [2].

Solving these optimization problems is usually hard, often leading to high computational costs when seeking exact solutions. Prior research has therefore proposed heuristics and approximations to efficiently find sub-optimal strategies that offer practical performance. This is especially crucial in MultiRobot Patrolling (MRP) scenarios where the problem's complexity grows significantly with the number of robots. Nonetheless, employing multiple patrollers remains an effective means to increase performance and robustness. In our work, we propose a novel offline approach to computing MRP strategies by combining two commonly used techniques, which are typically considered mutually exclusive.

Assuming that m is the number of robots, the first technique involves assigning each robot the patrolling task over the whole environment and using a coordinated strategy [15]. This strategy typically relies on a single Hamiltonian cycle (or traveling salesman problem – TSP cycle) that all robots follow. Coordination entails synchronizing the traversal of this path by the m robots, resulting in a collective reduction of maximum idleness on any vertex by a factor of m when robots are uniformly spaced along the tour [7]. While this approach enhances patrolling compared to a single robot, it necessitates computing a complete graph tour and does not leverage the advantages that a potential division of effort among the robots might induce.

The second technique is instead an implementation of the *divide et impera* principle [10], [16], [18]. The idea is to partition the environment into m subsets of the sites to protect, one for each robot, i.e., if V is the set containing all the graph vertices, the partition is defined as $P = \{V_1, \ldots, V_m\}$, with $\cup_i V_i = V$ and $V_i \cap V_j = \varnothing$ for $i \neq j$. The set P can prescribe a division of effort among the robots thus allowing to reduce the MRP problem to m simpler single-robot instances. In this setting, each robot independently patrols its assigned sub-graph. Since sub-graphs are disjoint, coordination among

979-8-3503-8457-4/24/$31.00 ©2024 IEEE

robots cannot introduce improvements and is therefore not necessary. Typically, each robot covers the TSP cycle (exact or approximate) on the vertices it is in charge of.

The two techniques can be characterized in terms of overlap between partition elements. The first case corresponds to full overlap (and hence a single element identical to V) while the second case would be associated with an empty overlap, namely a partition of V into m disjoint subsets. In this work, we introduce an intermediate approach to take advantage of *both* the coordination of patrollers (enabled by some overlap over portions of the environment) and the division of effort (requiring no overlap). Starting from the methods described above, we compute a partition of V into up to $m+1$ subsets, $P^+ = \{V_0, V_1, \ldots, V_m\}$ with $|V_0| \geq 2$, i.e., subset V_0 will always have at least two vertices (we motivate this constraint in the next sections.) In such partition, subset V_0 represents a portion of the environment that is patrolled by all m robots in a coordinated fashion, while the remaining subsets V_k define subsets patrolled by exactly one robot, i.e., V_k is patrolled exclusively by the k-th robot. The resulting patrolling strategy will be obtained by combining a set of Hamiltonian paths, computed on each partition's elements, to enable a scaling factor of m for the maximum weighted idleness over the shared vertices (V_0) while allowing independent patrolling over the non-shared parts (see also [7] for more details). By assuming such a structure in the joint patrolling strategy, we leverage situations where it is convenient to coordinate shared efforts on a selection of critical vertices that we dub "the core" while applying a disjoint division of effort over less important ones. Our formulation however allows for some or all of the V_k ($k > 0$) to be empty. When that happens, the corresponding k-th robot just patrols V_0. Note that in the limit, if all $V_k = \varnothing$ for $k > 0$, the patrolling strategy coincides with the classic coordinated strategy where each robot patrols the entire graph.

Below, we provide the following contributions.

- We define our patrolling setting (Section II), propose a novel formulation of the MRP problem (Section III) that can accommodate coordination and independent patrolling, and characterize its computational complexity.
- We define an exact formulation for computing optimal strategies that minimize a notion of weighted idleness (Section IV).
- We propose a set of principled heuristics (Section V) that enable efficient computation of strategies with competitive performance.
- We extensively compare and evaluate our methods (Section VI).

II. Problem Setting

We consider the classical graph patrolling setting refining a model we formerly considered in prior works [8], [5]. The environment is modeled by a graph $G = (V, E)$, where vertices $V = \{1, 2, \ldots, n\}$ represent locations and edges $(i, j) \in E$ represent their connections. A value c_{ij} represents the traveling cost (time or distance) to move from i to j. We assume that the graph is complete and that c_{ij} is the shortest cost. (Such a representation can always be computed from an arbitrary connected graph). Each vertex $i \in V$ is assigned an importance value $v_i > 0$, indicating the level of criticality for its protection. A set $R = \{1, 2, \ldots, m\}$ of m *patrollers* must protect the environment by moving from vertex to vertex in the graph. When a patroller visits a vertex, the vertex is protected, i.e., it cannot be compromised by an attacker, or, if the vertex is under attack, the attack is neutralized. Our optimization criterion uses the *idleness* of a vertex i, indicated as I_i and defined as the time between two successive visits to i by any of the patrollers. A common function to optimize is the weighted idleness:

$$\min \max_{i \in V} v_i I_i \tag{1}$$

This favors patrolling strategies where valuable vertices are visited more frequently, thus resulting in lower idleness. The m patrollers can be organized according to the following. (i) *Coordinated patrolling*: each patroller can visit any of the vertices in G. (ii) *Disjoint partitions*: the set of vertices V is partitioned into m non-overlapping subsets, and each patroller is assigned a subset of vertices (to prevent degenerate cases, we assume each subset has at least two vertices). (iii) *Overlapping partitions*: V is subdivided into m subsets that may share some vertices, each subset is then assigned to one of the patrollers.

III. Overlapping Partitions

We propose a new way to split the workload between m patrolling agents by introducing the concept of *core* and *periphery* on the vertex set V. The core V_0 is a subset of V patrolled by all m robots. The periphery $P = V \setminus V_0$ is the set of remaining vertices that is instead split into up to m non-overlapping subsets. Let these subsets be indicated as $V_1, V_2, \ldots, V_m$. The idea is that each robot k first patrols the core V_0 by traversing a Hamiltonian path between some start and end vertices, then patrols its assigned subset V_k, also traversing it with a Hamiltonian path, and then goes back to the core and repeats. This strategy is sketched in Figure 1 for a case where $m = 2$.

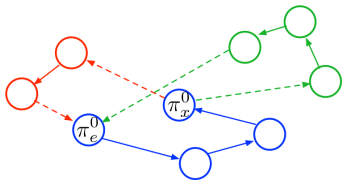

Fig. 1: Overlapping partition for $m = 2$. Both robots patrol the core's blue vertices following the same path between π_e^0 and π_x^0. Subsequently, robot 1 patrols the red vertices, while robot 2 patrols the green vertices.

We make these assumptions about robots' coordination.

- All robots traverse V_0 following the same Hamiltonian path π^0 from π_e^0 (entry) and π_x^0 (exit).
- When exiting π_x^0, if $V_k \neq \varnothing$ robot k traverses the assigned subset V_k by following an Hamiltonian path

π^k starting at π_e^k and ending at π_x^k. Then it travels back to π_e^0 and repeats. If instead $V_k = \varnothing$, robot k moves from π_x^0 to π_e^0 and resumes patrolling the core following π^0.

- Robots (i) adapt their speed so that each robot spends the same time to complete its tour (the time of the longest tour followed by any robot) (ii) uniformly distancing in time their arrivals at the core's entrance.

For a given core V_0 with entry vertex π_e^0 and exit vertex π_x^0, let t_c be the time it takes to follow the Hamiltonian path π^0. Similarly, for each of the subsets V_k, let t_k be the time to complete π^k also including the movement from π_x^0 to π_e^k (V_k's entry) as well from π_x^k (V_k's exit) to π_e^0 – see Fig. 1. Given this construction, the idleness of each vertex in V_k for $k > 0$ will be $t_c + t_k$ because the robot in charge of that subset must patrol, in sequence, V_0 and V_k. However, vertices in V_0 will be subject to a lower idleness thanks to robots' coordination. If robots coordinate their starts from π_e^0, then the maximum idleness experienced by the vertices in V_0 will be scaled by a factor of m. More formally, when we impose the existence of a shared core V_0, the objective function defined in Eq. (1) can be rewritten as follows:

$$\min \max_{1 \le k \le m} \left\{ A^c \frac{t_c + t_k}{m}, A^p (t_c + t_k) \right\} \tag{2}$$

where

$$A^c = \max_{i \in V_0} \{v_i\}, A^p = \max_{i \in V \setminus V_0} \{v_i\}.$$

Note that, to be well-defined, this formulation requires that $|V_0| \ge 2$ to allow for at least two vertices in V_0 as π_e^0 and π_x^0 must be distinct. However, if useful towards the solution of the above minimization problem, we do allow $V_k = \varnothing$ for one or more periphery sets ($k > 0$). When that is the case, the corresponding agent just patrols the core. The following problem formulation formalizes the problem we described.

> **Overlapping Partitions Problem (OPP)** Given a weighted graph $G = (V, E)$ with n vertices, edge costs $c : E \to \mathbb{R}^+$, and vertex values $v : V \to \mathbb{R}^+$. Let m be a given number of robots. Determine a core subset $V_0 \subset V$ with at least two vertices and a partition of $V \setminus V_0$ into at most m elements that solve the minimization problem defined by Eq. (2).

The pivotal question therefore is how to determine the core V_0, the periphery $V_1, \ldots, V_m$, and the corresponding Hamiltonian paths, solving the minimization problem formulated in Eq. (2), also in light of the following theoretical result.

Theorem 1: The OPP problem is NP-hard.

Proof: for the special case $m = 1$ the OPP problem is equivalent to the traveling salesman problem (TSP) over the entire set of vertices V.

IV. Exact formulation

We here provide an exact mathematical formulation for OPP. This formulation can be useful to solve small instances (i.e., graphs with few vertices), and to better understand the structure of the problem. Note that the formulation we provide allows for the case where all $V_k = \varnothing$ for $k > 0$, but imposes that $V_0 \neq \varnothing$. As will be shown in Section VI, in some peculiar cases this could be a disadvantage, i.e., the disjoint partitions approach could provide better solutions to the problem defined in Eq. 3.

Let $V^+ = V \cup \{0\}$ and $R^+ = R \cup \{0\}$ and let us consider a directed version of the edge set E, where we replace each edge with the two corresponding symmetric arcs. Notably, as we will show in the end, the formulation is nonlinear. Both sets include an additional 0 element, which will be useful for the following construction. In the set V^+, the element 0 represents an extra vertex that does not correspond to any real location but that allows us to express the resolution of the problem as finding a set of $m+1$ tours starting and ending at it. In the set R^+, the 0 element represents an extra robot which we will associate with the shared patrolling task of the core. We introduce the following binary decision variables for each $i, j \in V^+$ and $k \in R^+$.

$$x_{ij}^k = \begin{cases} 1, & \text{if robot } k \text{ will travel on edge } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

An assignment of the x_{ij}^k variables defines a partition of the environment. Each element $V^k \subseteq V$ ($k \in R^+$) of the partition can be recovered as follows: $V^k = \left\{ j \in V \mid \sum_{i=1}^n x_{ij}^k = 1 \right\}$. Another set of similar binary variables y_{ij}^k is introduced with the same definition but a different meaning (see usage below).

Notice that the above set definition assumes that the assignment to the x_{ij}^k variables is consistent with the set of constraints that will be introduced in the formulation, which will enforce that each vertex will be visited by exactly one robot (including robot 0). The definition has the following rationale: if the robot k travels vertex (i, j) for any i then j is considered part of that robot's subset of vertices. In our problem formulation, V^0 will represent the vertices assigned to the core, i.e., the portion that is shared among the m robots. Conversely, V^k for $1 \le k \le m$ represents the vertices that are patrolled exclusively by robot k.

The variable t_i represents the time at which vertex i is visited within the sequence of vertices in the partition to which i belongs. In the solution, we do not require that variables t_i will be assigned values that are consistent with traveling costs: we only require that if a vertex i is visited later than a vertex j then $t_i \ge t_j$.

The formulation shown in Table I is inspired by the ILP formulation for the maxmin Multiple TSP (see [14]) from which we take a basic set of constraints and we extend it to enforce the core/periphery structure we propose in this work.

Constraints (4) and (5) require that each robot leaves and returns to the depot along a single arc. Constraints (6) and (7) require that exactly one robot shall arrive and leave each vertex (except for vertex $i = 0$, the dummy depot). Constraints (8) ensure that the robot that arrives is the same one that leaves. Constraints (9) are the Miller-Tucker-Zemlin (MTZ) subtour-elimination constraints. They eliminate solutions where one robot covers its vertices through two or more disjoint tours on the graph. Here it is assumed that $n \ge m$

$$\min U \tag{3}$$

s.t.

$$\sum_{j=1}^{n} x_{0j}^{k} = 1, \quad \forall k \in R^{+} \tag{4}$$

$$\sum_{i=1}^{n} x_{i0}^{k} = 1, \quad \forall k \in R^{+} \tag{5}$$

$$\sum_{k=0}^{m}\sum_{i=0}^{n} x_{ij}^{k} = 1, \quad \forall j \neq i \in V \tag{6}$$

$$\sum_{k=0}^{m}\sum_{j=0}^{n} x_{ij}^{k} = 1, \quad \forall i \neq j \in V \tag{7}$$

$$\sum_{i=0}^{n} x_{ij}^{k} = \sum_{i=0}^{n} x_{ji}^{k}, \quad \forall j \in V, k \in R^{+} \tag{8}$$

$$u_i - u_j + (n-m+1)\sum_{k=0}^{m} x_{ij}^{k} \leq n-m, \quad \forall i \neq j \in V \tag{9}$$

$$x_{i0}^{0} + x_{0j}^{k} - 1 \leq y_{ij}^{k}, \quad \forall i \neq j \in V, k \in R \tag{10}$$

$$x_{0j}^{0} + x_{i0}^{k} - 1 \leq y_{ij}^{k}, \quad \forall i \neq j \in V, k \in R \tag{11}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\left(x_{ij}^{0} + x_{ij}^{k} + y_{ij}^{k}\right) \leq I^{k}, \quad \forall k \in R \tag{12}$$

$$\frac{1}{m}\sum_{i=0}^{n} x_{ij}^{0} v_j \leq A^{c}, \quad \forall j \in V \tag{13}$$

$$\sum_{i=0}^{n} x_{ij}^{k} v_j \leq A^{p}, \quad \forall j \in V, k \in R \tag{14}$$

$$A^{w} I^{k} \leq U, \quad w \in \{c, p\}, k \in R \tag{15}$$

TABLE I: An exact formulation for the OPP problem.

and that, in the solution, each robot will cover at least one vertex.

The above basic constraints will enforce solutions whose structure is the same as that required for classical multi-TSP problems: $m+1$ tours, each starting and ending at the depot $i = 0$. The tours are such that each vertex $i > 0$ is covered once by just one robot. The tour for robot k is composed in this way: leaving the depot, traveling an *inner path* π^k, and returning to the depot. We need to combine these $m+1$ tours in order to obtain m tours, one for each real robot $k > 0$, each defined as start at the first vertex of π^0, follow π^0, travel to the first vertex of π^k, follow π^k, travel to the first vertex of π^0, repeat. We can show that the optimal solution to our problem can be expressed in the multi-TSP form described above provided that the required objective function is defined. Hence we embed our required structure in the costs minimized by the objective function.

Constraints (10) and (11) select, through the y variables, those arcs that will connect, for each real robot, the inner path π^0 with the inner path for just that robot π^k. Constraints (12) define I^k as the upper bound on the tour cost (and hence, the idleness) for a real robot $k > 0$. Constraints (13) and (14) similarly define, an upper bound for the maximum vertex value in the core (A^c) and outside of it (A^p). Notice that these upper bounds will be "pushed" to be tight since the problem is a minimization one. Finally, Constraints (15) provide an upper bound to the objective function we seek to minimize. This cost is the maximum weighted idleness that accounts for a discount of $1/m$ for the maximum value among the vertices in the core (those visited by π^0). Notice how this final constraint, crucial to the whole problem, is also the one that introduces a non-linearity.

As mentioned earlier, the exact formulation illustrated in this section could be used to solve small problem instances, i.e., instances with few vertices and edges, but obviously do not scale to larger ones. For this reason, in the next section, we propose various heuristics that scale with the problem's size.

V. Heuristics for OPP

In light of the computational complexity of OPP, in this section we introduce various heuristic methods for finding solutions to large problem instances where exact methods cannot be applied. As mentioned in the previous section our problem is related to the min-max multiple traveling salesman problem (mTSP) formulation [14] and this insight informs our solving strategy. A valid solution to OPP can be determined as follows:

- Select some vertices to form the core.
- Compute a TSP tour on the core with arbitrary starting and ending vertices (this can be achieved for example by adding a *dummy vertex* to the graph that is connected to all other vertices with edges of zero cost) and restricting the TSP tour to start and end at the dummy vertex; these arbitrary starting and ending vertices become π_e^0 and π_x^0;
- Solve the min-max mTSP problem on the periphery where all m tours must start at π_x^0 and end at π_e^0. Furthermore, the solution to the mTSP will partition the periphery vertices into disjoint sets for the robots.
- Finally by combining the Hamiltonian path on the core with each of the m Hamiltonian paths on the periphery we form a valid path for each robot and thus a valid solution to the problem.

Thus given a core set of vertices, a TSP solver, and a min-max mTSP solver[1] one can form a solution by performing the aforementioned steps. However, the question of how to select the vertices that belong in the core so as to effectively minimize Eq. 2 remains open. In the following, we propose different methods aiming at obtaining low values for Eq. (2) that differ in the strategy to build the core.

A. K-means core

A first heuristic is based on the intuition that to minimize the objective function one should minimize the distance traveled by the robots, i.e., reducing the sum $t_c + t_k$. These two terms are conflicting since reducing the total distance traveled on the core means removing a vertex from there and adding it to the periphery hence increasing the distance

[1]Note that we are not requiring an exact solver for these two NP-hard problems.

traveled by some robot on its independent workload. Because we would like every robot to have some minimum workload we propose to use k-means clustering based on the Euclidean distance between vertices with $m+1$ clusters to identify the subsets $V_0, V_1, \ldots, V_m$. After running the algorithm, we assign the largest cluster to the core.

B. Weighted K-means core

Weighted k-means clustering [13] works similarly to the non-weighted version but exploits the fact that each of the vertices in the graph has an associated value. Since in Eq. (2) the values of vertices multiply the travel times, in the weighted k-means heuristic the values of the vertices are used to scale the weight of the clusters. Then just as we did with non-weighted k-means, we compute $m+1$ clusters and denote the largest cluster as the core set of vertices.

C. Balanced Weights Heuristic (BWH)

By examining Eq. (2) we note some principles governing the nature of the objective function. The variables t_c and t_k are found in every term and furthermore in general minimizing one of the variables means increasing the other since a smaller t_c can only be achieved by removing a vertex from the core and thus adding said vertex into the periphery. The same is true for decreasing the largest tour on the periphery - it can only be achieved, in general, by removing a vertex from the periphery and adding it to the core set or adding it to another partition whose t_k value will increase. This suggests that the optimal is found when the terms $\{A^0\frac{t_c+t_0}{m}, A^0\frac{(t_c+t_1)}{m}, ..., A^1(t_c+t_k), ..., A^k(t_c+t_k)\}$ are roughly similar in value. Note here that the maximum of the first k terms is simply $A^0\frac{t_c+max_k(t_k)}{m}$. Given that A^0 is the largest valued vertex in the core and each subsequent A^k is obtained from the largest valued vertex for each robot's independent workload, a simple heuristic for partitioning the set of vertices into core and periphery is obtained by adding a vertex into the core when its corresponding value satisfies the inequality:

$$v_j > \frac{max_{i\in V}(v_i)}{m}.$$

In this way, we can keep high-valued vertices in the core and, at the same time, push out into the periphery vertices with values similar to or less than the first coefficient in the objective function, $\frac{A^0}{m}$. This heuristic aiming at balancing the terms in the objective function is dubbed *balanced weights heuristic (BWH)*.

D. Local Search Heuristic (LSH)

Starting from the balanced weights heuristic, we can develop an improved version called *local search heuristic (LSH)*. Since the core is shared between k robots it follows that generally, we would like cores that include more vertices with respect to subsets in the periphery. Starting from the core proposed by BWH, the local search heuristic iteratively tries to improve the current selection of the core by evaluating random additions to it. In this way, our local search method tries to build solutions that are at least as good as those provided by the BWH approach. At each iteration for each vertex i in the periphery set, with probability p, it is removed from the periphery set and introduced into the core set ($p = 0.6$ in our experiements). We can then evaluate this new candidate core set by computing its objective value from Eq. 2. If the new core is better than the previous one it is saved otherwise, it is passed over. An outline of the local search heuristic is given in Algorithm 1.

Algorithm 1: Local Heuristic Search

```
Data: B search budget p vertex add probability
C ← Balanced Weight Heuristic;
o ← objective value for C (from equation 2);
for i ← 0 to B do
    D ← C;
    P ← V − C;
    for n in P do
        with probability p, D = D ∪ n;
    solve TSP and mTSP sub problems;
    s ← objective value for D (from equation 2);
    if s < o then
        o ← s    C ← D
return C, o
```

VI. Evaluation

In this section, we perform two types of evaluations. First, we assess the relative merit of the heuristics we introduced in section V. Second, we evaluate if the novel formulation we proposed in this paper is advantageous when compared with the coordinated patrolling and disjoint partition methods formerly proposed in the literature. To compare with the disjoint partition method, we use the k-Max Cut algorithm we formerly proposed in [8]. This approach partitions the graph into k subsets and subsequently assigns each robot a sub-graph to patrol. For each sub-graph, a TSP tour is calculated and serves as the robot's path through the area.

We performed evaluations on graph sizes ranging from 10 vertices to 60 vertices, with 10 instances for each graph size. Vertex locations are generated by randomly sampling points in a 50×50 square and edge weights are given by the Euclidean distance between the vertex coordinates. Finally, vertices values are sampled from the range 1 to 100 using a uniform distribution. To solve the TSP and mTSP problems we used the routing module provided by the Google OR-Tools library [11], while for k-means we use scikit-learn [17].

To get an idea of how far from the optimal solution our methods are, we first performed a set of experiments on graphs of small size comparing the solutions provided by the different heuristics and the exact solution. Figures 2a and 2b show the *competitive ratio* of the various heuristics for graph instances of size 10 and 15. The competitive ratio is defined as the ratio between the objective value returned by the heuristic method and the optimal value (with 1 therefore being the best one can aim for). Both the charts for 2 and 3 robots show similar trends, namely that BWH and LSH almost always perform better than the other methods, and have a competitive ratio close to 1. Furthermore, local search

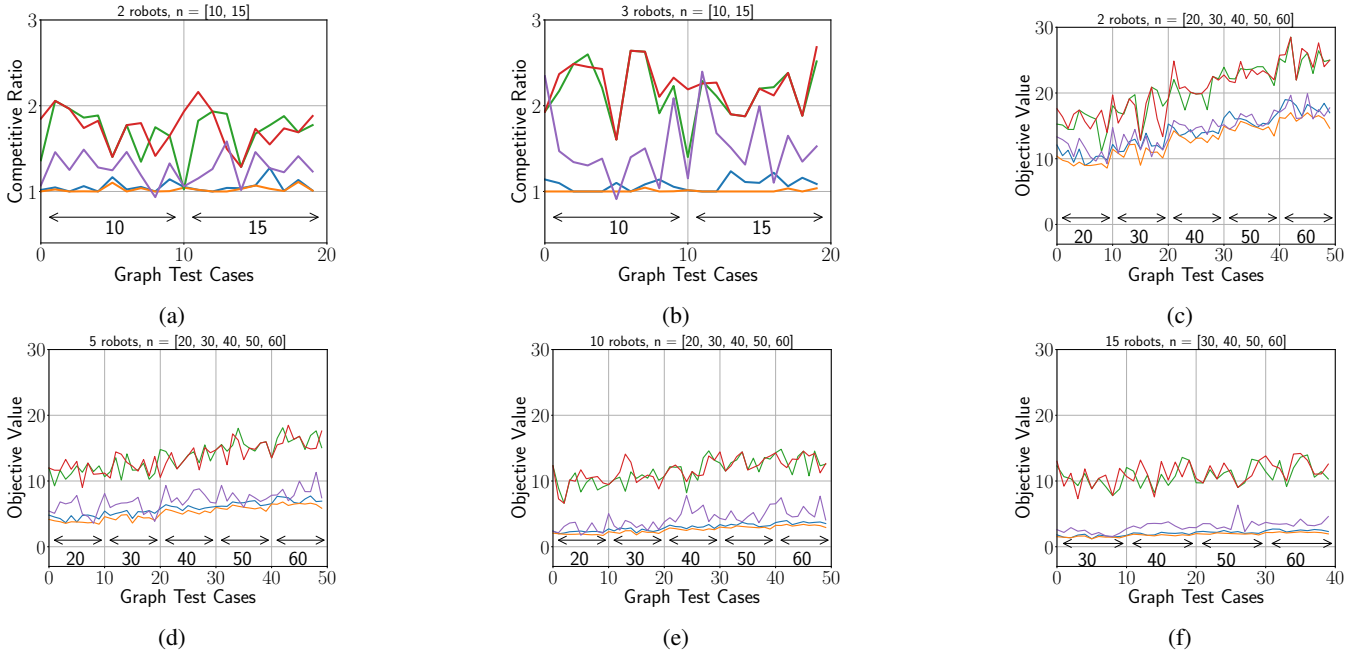

Fig. 2: Comparison between the different heuristics for different graph sizes and number of robots. In all figures, the following color coding is used: red for k-means; green for weighted k-means; purple for k-max cut; blue for the balanced weights heuristic; and orange for the local search heuristic.

can improve BWH as evidenced in Figure 2b. Important to note in both Figures 2a and 2b is that the k-Max Cut sometimes does even better and has a competitive ratio of less than 1. This apparent contradiction can be explained by the fact that some graph instances naturally cluster into distinct sub-graphs. This arrangement is advantageous to the k-Max Cut method since it will keep the independent robot tours small, while it is also detrimental to the heuristics and the exact method since selecting a core set of vertices will incur a large travel cost when moving between the two natural sub-graph clusters. The next set of experiments compares the objective values of the returned solutions for each of the various heuristics for larger graphs. Note that as the number of vertices exceeds 15 the exact solution becomes too costly to compute and we therefore compare the objective values between the different solutions rather than the competitive ratio. Figure 2c, 2d, 2e, and 2f show the values found for 2, 5, 10, and 15 robots and graph sizes varying from 20 to 60 vertices. As can be seen, the two heuristics proposed in this paper almost invariably are the most effective, and in particular they outperform the method based on disjoint partitions. The experiments described thus far show that the method we propose outperforms the disjoint partitions approach. However, one could wonder if it also outperforms the coordinated method with all robots patrolling all vertices. An analysis of the results produced shows that the answer is affirmative, especially in the case of constrained resources (e.g., a small number of robots and a large number of vertices). Figure 3 shows two examples, with the left graph showing the solution produced by the optimal method and the right one produced by LSH. In the figure, the size of the vertices is proportional to the v_i values and it outlines how important vertices are included in the core (plotted in blue). In both instances, better patrolling strategies are obtained by assigning each robot to a subset of vertices in the periphery (paths plotted in different colors) rather than including everything in the core as in the coordinated method. These two samples are representative of the entire dataset.

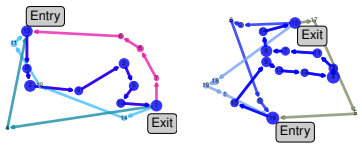

Fig. 3: Left: optimal solution for 3 robots and 15 vertices. Right: LSH solution for 3 robots and 20 vertices.

VII. Conclusions

We proposed a new strategy to solve the MRP problem bridging between the formerly proposed approaches based on uncoordinated patrolling and disjoint partitions. The key idea is that robots jointly patrol the "important" part of the graph (which we dub *core*), and independently patrol less important parts of it. Owing to the intrinsic computational complexity of optimal methods, we introduced four heuristics, one of which (LSH) emerges as the best. Our experiments show that for small problem sizes we obtain a competitive ratio of almost one, and for larger problem sizes it handily beats the disjoint partitions method. Our test cases also show that in most instances our method generates patrolling strategies better than those produced by the coordinated and disjoint partitions methods.

REFERENCES

[1] 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 *International Workshop on the Algorithmic Foundations of Robotics*, pages 107–123. Springer, 2020.

[2] 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*, 33(1):138–154, 2014.

[3] Ahmad Bilal Asghar, Stephen L Smith, and Shreyas Sundaram. Multi-robot routing for persistent monitoring with latency constraints. In *2019 American Control Conference (ACC)*, pages 2620–2625. IEEE, 2019.

[4] N. Basilico. Recent trends in robotic patrolling. *Current Robotics Reports*, pages 1–12, 2022.

[5] N. Basilico and S. Carpin. Balancing unpredictability and coverage in adversarial patrolling settings. In *Proceedings of the 2018 Workshop on Algorithmic Foundations or Robotics*, pages 762–777, 2020.

[6] N. Basilico, N. Gatti, and F. Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *ARTIF INTELL*, 184:78–123, 2012.

[7] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proc. IAT*, pages 302–308, 2004.

[8] C. Diaz Alvarenga, N. Basilico, and S. Carpin. Multirobot patrolling against adaptive opponents with limited information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2486–2492, 2020.

[9] X. Duan and F. Bullo. Markov chain–based stochastic strategies for robotic surveillance. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:243–264, 2021.

[10] Y. Elmaliach, N. Agmon, and G.A. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.

[11] Vincent Furnon and Laurent Perron. Or-tools routing library.

[12] S.K.K. Hari, S. Rathinam, S. Darbha, K. Kalyanam, S.G. Manyam, and D. Casbeer. The generalized persistent monitoring problem. In *Proceedings of the American Control Conference*, pages 2783–2788, 2019.

[13] K. Kerdprasop, N. Kerdprasop, and P. Sattayatham. Weighted k-means for density-biased clustering. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*, pages 488–497, 2005.

[14] R. Necula, M. Breaban, and M. Raschip. Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 873–880, 2015.

[15] 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*, pages 136–138, 2019.

[16] J.M. 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*, 38(14):1674–1694, 2019.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] D. Portugal, C. Pippin, R.P. 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*, pages 363–369, 2014.

[19] D. Portugal and R. Rocha. A survey on multi-robot patrolling algorithms. In Luis M. Camarinha-Matos, editor, *Technological Innovation for Sustainability*, pages 139–146, 2011.

[20] F. Rubio, F. Valero, and C. Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596, 2019.