Wm-SPLITOUR: a Min-Max Weighted Multi-TSP Algorithm for Robotic Patrolling

Alex Bassot[1⋆], Stefano Carpin[2], and Nicola Basilico[3]

[1] University of Edinburgh, Edinburgh, UK a.h.j.bassot@sms.ed.ac.uk
[2] University of California, Merced, USA scarpin@ucmerced.edu
[3] University of Milan, Milano, Italy nicola.basilico@unimi.it

Abstract. We address the multi-robot patrolling problem where assets have varying importance. Traditional patrolling strategies minimize weighted idleness by visiting high-value assets more frequently, but this approach has the potential of revealing sensitive information to adversaries. We introduce the min-max weighted multi-TSP (WmTSP), an NP-hard problem that enforces uniform visitation frequencies within cycles to achieve value-masking. To solve this problem, we propose Weighted m-SPLITOUR (Wm-SPLITOUR), a polynomial-time approximation algorithm that scales cycle lengths by their maximum vertex value. Simulations show our method improves weighted idleness by up to 50% over previous approaches, especially in complex, large-scale environments.

1 Introduction

With the increasing adoption of robots for security and patrolling applications, interest in algorithms for planning the actions of robots engaged in these tasks has grown accordingly [13, 29, 32]. With the term *patrolling algorithms* we refer to computational methods that determine how one or more robots should move within an environment with the goal of protecting a set of assigned assets against external threats [13]. In this context, the environment of interest is typically modeled as a graph, where vertices represent the assets to protect and edges model the ability of a robot to move between assets. Accordingly, if a malicious attacker tries to compromise an asset when a robot patroller is at the associated vertex, the attack is neutralized. When the number of assets exceeds the number of robots[1], these must move between different assets to ensure they are not left unguarded. Vertices might be given a value representing how important the asset is, and edges are associated with costs that model the time it takes for a robot to move from asset to asset.

Patrolling algorithms can be viewed as solutions to optimization problems, whose goal is to determine how to allocate available resources (i.e., robots) so as to provide the best possible protection to the assets, according to a given objective function that measures "how good" a particular allocation is. Within

⋆ This work was conducted while the author was affiliated with University of Milan.
[1] If this is not the case, then one places one robot per asset and the solution is trivial.

this general framework, numerous problem variations exist, some of which are discussed in Section 2. A common theme among most solutions is the requirement that assets be revisited periodically to ensure that they are not under attack by an adversary. This naturally leads to the notion of *idleness*, defined as the maximum time elapsed between two successive visits to an asset. Evidently, the higher the idleness for an asset, the weaker protection is being provided to it due to the presence of a wider temporal window to initiate and complete an attack. A related concept is the so-called *weighted idleness*, i.e., the product between the value of a vertex and its idleness. These quantities often lead to patrolling strategies where vertices with higher weight are visited at a higher frequency to reduce their weighted idleness, while it is admissible to visit vertices with lower weights less frequently. These concepts are illustrated in Figure 1.

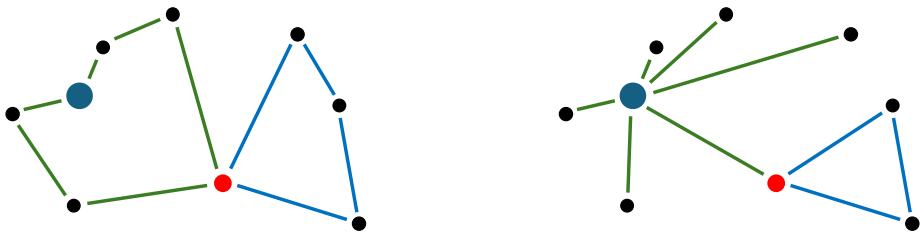

Fig. 1: Two possible strategies for patrolling a weighted graph with two agents (green and blue) where one vertex (blue vertex) has value higher than the others.

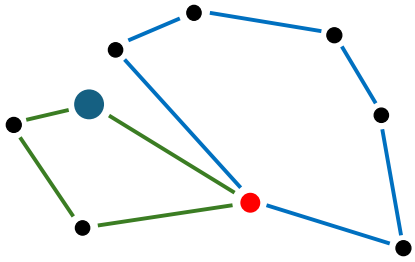

Fig. 2: Our strategy aims at providing effectiveness by minimizing the weighted idleness of the assets, while being robust to information leakage.

The two paths shown in the left panel of Figure 1 (green and blue path), split the graph in two subgraphs, with the red vertex being on both paths (the role of a shared vertex will be clarified later). In this case, the weighted idleness of the vertex with the highest value (blue vertex) is determined by product between its weight and the time it takes to cycle through the green path it is part of.

In this situation, longer paths lead to higher weighted idleness. To mitigate this problem, strategies like the one on the right panel of Figure 1 are often introduced, where the green path implements a *star shaped* patrolling strategy alternating visits between the high value vertex and the others. A large portion of the literature is devoted to algorithms centered on these two concepts, and many proposed solutions ultimately result in deterministic patrolling strategies, often inspired by game-theoretic models (e.g., Stackelberg equilibria [23]). However, deterministic routes may in principle be observed and learned by an adversary, who could then time attacks based on predicted robot movements. For example, an external agent observing a robot implementing the star-shaped strategy would easily figure out that the blue vertex has higher value, even without having any preliminary knowledge. In our previous work [6,7,12], we addressed this issue by introducing varying degrees of randomness into the patrolling strategies, with the goal of reducing an adversary's ability to learn and predict robot behavior and assets' values. We call this feature *value masking*. In this paper, we continue along this line of research, but approach the problem from a different perspective that still relies on deterministic patrolling strategies.

Observe that the two features required for solutions to be effective are conflicting: lowering the idleness of more important assets demands more frequent visits to ensure security, but doing so proportionally to the asset's value reduce value masking potential. To mitigate the information leakage, we consider patrolling strategies by multiple robots in which each vertex is visited as part of a cycle, thereby preventing any disclosure of relative asset importance because all vertices that are visited as part of a cycle experience all the same idleness. At the same time, we seek solutions designed to have the assets' worst weighted idleness minimized, providing at the same time patrolling tours that are aware of assets' importance.

This approach is shown in figure 2. As it can be seen, the vertex with the higher weight is included in a shorter path, so that its weighted idleness is smaller. Note that in this case all vertices along the green path experience the same idleness.

Therefore, given m robots to patrol a graph with n vertices, how should these vertices be partitioned and patrolled so that the overall maximum weighted idleness is minimized? This problem, which is related to the multi-TSP problem and which we call *min–max weighted m TSP*, is formally defined in Section 3 and shown to be $\mathcal{NP}$–hard. We therefore propose an approximation algorithm that is also applicable to a broader class of multi vehicle routing problems. The main contributions of this paper are as follows:

- We introduce a novel patrolling problem, called *min–max weighted m TSP* (or WmTSP in short), and prove its $\mathcal{NP}$–hardness.
- We propose a novel approximation algorithm called Wm-SPLITOUR, which augments the approximation logic of the min–max multi-TSP with vertex weights.
- We conduct experiments showing how our algorithm performs well in practice, significantly outperforming existing baselines.

The remainder of the paper is organized as follows. Related work is briefly discussed in Section 2. The WmTSP problem is formally defined in Section 3, where its computational complexity is also established. The proposed approximation algorithm and the associated theoretical analysis is presented in Section 4. Experimental results are reported in Section 5, while conclusions together with directions for future work are given in Section 6.

2 Related Works

The first studies on the patrolling problem for teams of robots within a shared environment identify the idleness of the surveilled locations as the key objective function to be minimized [4,19]. The usual abstraction of the patrolled environment is a graph where the vertices represent locations and edges represent the time needed to go from one location to another. The research in this field links with the field of vehicle routing, since minimizing the travel length is a key factor in minimizing the idleness of the vertices.

Typically, Vehicle Routing Problems (VRP) are $\mathcal{NP}$–hard problems [26], therefore the research focus on finding approximate solutions. The work done in [19] is one of the first studies that adopts approximate solutions of the *Traveling Salesman Problem* (TSP) to generate strategies that lower the idleness of the patrolled sites within approximation bounds. Variants of the multi–robot patrolling problem have been studied in literature [11,28,30,31], where the solutions proposed are generated through various heuristics and approximation algorithms for routing. One of the most important procedures that finds an approximate solution for the TSP in general metrics is the Christofides algorithm [20], which provides a $\frac{3}{2}$–approximation and it is still widely adopted in literature. This factor was recently slightly improved [25]. For multi–robot routing, many variants of approximation algorithms for minimization and min–max problems have been studied [9, 21]. One of the most popular and natural extensions of the TSP is the *min–max multi TSP* (mTSP), where, given a depot vertex and a team of m robots, the objective is to find a set of m disjoint cycles starting and ending at the depot vertex, such that the maximum length among the cycles is minimized. The best known approximation factor for this problem is $\frac{5}{2} - \frac{1}{m}$, provided by the m–SPLITOUR algorithm in [21].

Besides minimizing the lengths of the robots' paths, the patrolling problem presents other critical issues that must be taken into account in the solution. One of this criticalities is the fact that in real–life scenarios some locations have to be patrolled more often than others due to their higher importance. Work done toward this idea considers the objective of minimizing the maximum *weighted idleness* of the vertices [3,27]. Typically, approximate solutions for this problem increase as the ratio γ between the highest and the lowest values of the vertices. The single–robot version is tackled in [3], where the authors provide a $O(\log \gamma)$–approximation algorithm for the min–max weighted idleness exploiting the novel idea of latency walks. In [1] the results from [3] are adopted to solve the multi–robot version of the problem with m robots, with an approximation factor of

$O(m^2 \log \gamma)$, recently improved to $O(m \log \gamma)$ in [18]. Notice that the multi-robot solutions have approximation factors that grow at least linearly with the number of robots m. A similar concept to the min–max weighted idleness is explored in [11], where the vertices have latency constraints instead of values representing their importance. Another multi-robot patrolling problem variant which aims at minimizing the worst weighted idleness is studied in our recent works [8, 16], where locations are divided into a high-priority core patrolled by all robots and a lower-priority periphery partitioned into subsets, each assigned to one robot only. In [8] we defined the problem and we addressed it with heuristic methods, while in [16] we proposed a first approximation algorithm for a simplified version of the problem. Problem instances that involve values associated to vertices are not only related to patrolling. In [17], for example, the authors present the demand–weighted routing problem, a variant of the min–max multi–TSP where the cost of each cycle is its length multiplied by the sum of the vertex values that it contains. The key difference between our weighted variant and the formulations above is that weights are not additive but treated under a worst-case logic (we do not multiply the sum of weights of the visited vertices to the cycle length, but rather multiply the maximum value by the cycle length). Combining this with the requirement of avoiding repeated visits to the same vertex during each cycle iteration yields a variant that shares unique aspects with both the multi-TSP and the multi-robot patrolling domains. To the best of our knowledge, such a variant has not been previously studied.

Another pivotal issue that comes with patrolling is the capacity of being robust to attackers observations. This research field, known as *adversarial patrolling* [13], considers scenarios where possible adversaries have total [2] or partial [6, 12] knowledge about the environment or the patrolling strategy. Usual approaches model the problem as a stochastic game [14] and focus on constructing non–deterministic paths for the robots, often obtained with Markov chains, with the objective of minimizing the probability of a successful attack [10,12,15]. Other studies focus their work on deterrence, i.e., how the strategy can influence the attacker decision of attacking or not attacking [5, 24]. With this work, we move toward this concept by building strategies that do not reveal information about the assets' importance, providing deterrence in the following sense: by observing the robot's paths, the attacker cannot distinguish the most important asset from the others in the same cycle. This implies that a rational attacker must take into account the fact that they might also end up attacking the vertex with the lowest value.

3 Problem Formulation and Objectives

In our problem, the environment to be patrolled is modeled by a complete metric graph $G = (V, E, w, \ell)$ where:

- $V = \{0, 1, \ldots, n\}$ is the set of *vertices* and a special vertex called *depot* is by convention assumed to be the vertex 0;
- $E = \{(i, j) : i, j \in V\}$ is the set of undirected *edges*;

- $w(i) = w_i \in (0, 1]$ denotes the *value* of vertex $i \in V \setminus \{0\}$;
- $\ell_{i,j} \in \mathbb{R}_+$ is the *length* (or travel time) of edge (i, j); lengths satisfy the triangle inequality: $\ell_{i,j} \leq \ell_{i,k} + \ell_{k,j}$ for all $i, j, k \in V$.

A *path* π is an ordered sequence of vertices such that no vertex appears in the sequence more than once. π naturally defines an ordered sequence of edges, formed by the ones that link each vertex in π with the one immediately following, except for the last one. A *cycle* C is obtained by the closure of a path, that is, by adding the edge that connects its first and last vertices. In the following, we identify a path or a cycle with the set of its constituent vertices or edges, depending on the context.

Given a subset of edges $S \subseteq E$ (such as a path π or a cycle C), we extend the length and value notations such that the length of S is the sum of the lengths of its edges:

$$\ell_S \coloneqq \sum_{(i,j)\in S} \ell_{i,j}$$

The value of S is defined as the maximum value among the vertices it traverses:

$$w_S \coloneqq \max_{(i,j)\in S} \max\{w_i, w_j\}$$

We define $w_{\max} \coloneqq \max_{i\in V} w_i$ and $w_{\min} \coloneqq \min_{i\in V} w_i$ as the highest and lowest weights among the vertices of the graph, and we call their ratio $\gamma \coloneqq \frac{w_{\max}}{w_{\min}}$.

For a given environment G, let $m < n$ be the number of patrolling robots. We define a m-tour $\Theta = \{C_1, \ldots, C_m\}$ as a set of at most m disjoint cycles rooted at the depot (i.e., starting and ending at vertex 0) that collectively visit every vertex in V. The left panel in Figure 1 and Figure 2 show two different 2-tours for the same graph. The red vertex in the graph identifies the depot vertex. This is known to be a well-defined candidate solution for the mTSP problem [21]. Let $\mathcal{T}_m$ be the set of all m–tours on G. We introduce the following problem.

Min-Max Weighted mTSP (WmTSP) Let $G = (V, E)$ be an environment to patrol defined as above, and let $m < n$ be the number of robots tasked with patrolling G. Find the m-tour Θ^* on G that satisfies the following objective:

$$\Theta^* = \arg\min_{\Theta\in\mathcal{T}_m} c(\Theta) \tag{1}$$

$$\text{where } c(\Theta) \coloneqq \max_{C\in\Theta}\{w_C \ell_C\}.$$

WmTSP naturally aligns with our motivating patrolling setting, requiring a balance between weighted idleness minimization and the enforcement of vertex value masking as we discussed in Section 1. The problem remains grounded in multi-robot patrolling since the objective function of Eq. (1) encodes a worst-case weighted idleness across G. Specifically, the model assumes that each robot repeatedly traverses a uniquely assigned sub-cycle of the m-tour Θ, thereby

bounding the maximum time an asset remains unguarded relative to its importance. Simultaneously, the *value-masking* feature emerges as a direct structural consequence of this formulation. By constraining robots to traverse fixed cycles, every vertex within a given cycle is subject to the exact same revisit frequency. Consequently, an adversarial agent tracking a single robot observes a uniform visitation pattern, preventing them from distinguishing high-value assets from lower-value ones based on visit statistics. Note that the problem formulation enforces solutions to be disjoint tours sharing only a depot vertex. This improves the method's deployability, as shared vertices among different robots would introduce non-trivial challenges related to coordination.

This problem generalizes the *min–max multi–TSP* (mTSP), which corresponds to the special case where $w_i = 1$ for all $i \in V \setminus \{0\}$. Since the mTSP is $\mathcal{NP}$-hard [21], WmTSP is $\mathcal{NP}$-hard as well. Our objective is to devise an algorithm that provides theoretical approximation guarantees while achieving satisfactory empirical performance. The natural starting point is analyzing the approximability of the mTSP, given that the two problems are closely related. To the best of our knowledge, the tightest approximation factor currently known for the mTSP is provided by Frederickson et al. [21]: $\rho_{mTSP} = 1 + \rho_{TSP} - \frac{1}{m}$, where ρ_{TSP} is the approximation factor for the single-robot TSP. Their proposed heuristic, *m–SPLITOUR*, operates in two phases: (i) it computes an approximate single-robot TSP tour, and (ii) it removes the depot vertex (and its incident edges) to create a path, which is then split into sub-paths. This splitting step utilizes a heuristic designed to balance the lengths of the resulting sub-paths. In metric environments (where lengths satisfy the triangle inequality, as in our case), step (i) can be solved using Christofides' algorithm [20] ($\rho_{TSP} = \frac{3}{2}$), yielding therefore an overall approximation factor of $\rho_{mTSP} = \frac{5}{2} - \frac{1}{m}$.

These premises lead to a natural question: does the approximation logic for the standard mTSP transfer to our generalized setting considering weights? Specifically, can it handle the additional complexity given by the fact that the cost of a cycle is *weighted* by the maximum value among its visited vertices? In the following, we show that the answer is affirmative. We capture this result in this observation (we omit the proof since it follows trivially from the problem definitions).

Observation 1 *A ρ_{mTSP}-approximation for m TSP also yields an approximation for Wm TSP with factor $\rho_{WmTSP} = \gamma\rho_{mTSP}$.*

The *m-SPLITOUR* algorithm yields a valid baseline approximation, but it remains oblivious to the weighting mechanism inherent to our formulation. Effectively minimizing our objective requires balancing the *weighted* costs, scaling each cycle's length by the maximum value of its visited vertices, rather than merely balancing geometric lengths. Neglecting this coupling often leads to solutions that, while still satisfying worst-case approximation guarantees, perform poorly in practice. This discrepancy arises because accommodating cycle-dependent weights frequently requires tours of unbalanced lengths. This divergence is exemplified in Figure 3, where we show an example of how a weight-

Fig. 3: Impact of vertex weights. The graph has unitary edges and contains one high-value vertex (blue, $w = 1$) and three low-value ones (black, $w = 0.1$). Left: *m–SPLITOUR* balances geometric lengths, but yields a high cost of 3. Right: the optimal WmTSP solution has unbalanced lengths but a lower cost of 2.

aware optimal solution could be significantly different from the one computed with an unweighted approach.

Our goal is to bridge this gap by designing a heuristic that rigorously preserves the approximation guarantees induced by *m-SPLITOUR*, while exploiting vertex weights to minimize the objective in practice. To this end, we introduce Weighted m-SPLITOUR (Wm-SPLITOUR). This novel method informs the original splitting step with a weight-aware optimization criterion. By doing so, our approach retains the theoretical robustness of the baseline while performing significantly better on empirical evaluations. In the next section, we detail our method also proving how it maintains the same approximation factor of Observation 1, while in Section 5 we empirically show how it outperforms the baseline.

4 The W*m*-SPLITOUR Algorithm

In this section, we present Weighted m-SPLITOUR (Wm-SPLITOUR), a constructive approximation algorithm for the WmTSP. Our method augments the tour-splitting step proposed by Frederickson et al. [21]: instead of merely balancing the geometric lengths of the resulting sub-paths, we search for a split that minimizes the maximum *weighted* cost, effectively scaling the length of each sub-path by the maximum weight among the vertices it visits. This last problem of optimally splitting a path is well-known in the field of multi-robot patrolling; notably, Pasqualetti et al. [28] proposed a *linear graph partitioning* framework to efficiently address the unweighted variant. We build upon the same framework by mapping the vertex sequence from an initial TSP approximation onto a linear metric space. Subsequently, we extend their *left-induced m-partition* algorithm to minimize our specific, weight-dependent objective function.

We start by assuming to have computed a path $\pi = (p_{(1)}, \dots, p_{(n)})$ that visits all the n vertices in $V \setminus \{0\}$, Here $p_{(i)}$ denotes the i-th vertex in the sequence followed by the path. We map the vertices of π onto the real line $\mathbb{R}_+$ and we assign a coordinate $x \in \mathbb{R}_+$ to each vertex based on its cumulative distance from the start of the path. To be able to refer to a vertex both directly by its id and indirectly by its index of visit along the path π, we shall use the subscript

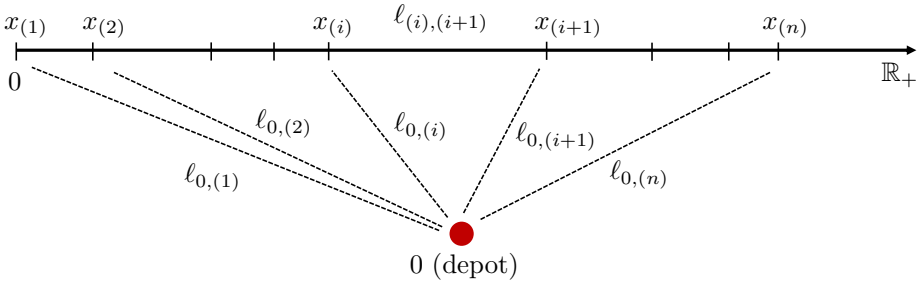

Fig. 4: A construction where the vertices visited by a path π are placed on the real line and connected to the depot.

$\cdot_i$ to identify the vertex by id (vertex i) and the parenthesized subscript $\cdot_{(i)}$ to identify the vertex by its index on the path (the i-th visited vertex). So, x_i denotes the coordinate of vertex $i \in V$, while $x_{(i)}$ refers to the coordinate of the i-th vertex in path π's sequence (vertex $p_{(i)}$). Coordinates are assigned according to the following rules: $x_{(1)} = 0$ and $x_{(i+1)} - x_{(i)} = \ell_{p_{(i)},p_{(i+1)}}$. Figure 4 (top part) depicts an example of this construction.

We now introduce two key definitions on which our method will rely on.

Definition 1. *An* m-partition *of a path* π *is an ordered set* $\Pi \coloneqq \{\pi_1, \ldots, \pi_m\}$ *of at most* m *disjoint sub-paths obtained by removing at most* $m-1$ *edges from* π*. Let* $1 \le i_1 \le i_2 \le \cdots \le i_{m-1} < n$ *be the splitting breakpoints. For instance, when we have* m *different breakpoints, the sub-paths are defined as:*

$$\begin{aligned}
\pi_1 &= \big(p_{(1)}, \ldots, p_{(i_1)}\big) \\
\pi_2 &= \big(p_{(i_1+1)}, \ldots, p_{(i_2)}\big) \\
&\vdots \\
\pi_m &= \big(p_{(i_{m-1}+1)}, \ldots, p_{(n)}\big)
\end{aligned}$$

We also define the set of all m*–partitions of* π*:*

$$Part_m(\pi) = \{\Pi \mid \Pi \text{ is an } m\text{-partition of } \pi\}.$$

Definition 2. *The* weighted rooted cost *of a path* $\pi = \langle (i,\cdot), \ldots (\cdot, j) \rangle$ *is the real positive number*

$$\ell_w(\pi) = w_\pi(x_j - x_i + \ell_{0,i} + \ell_{j,0})$$

The weighted rooted cost *of an* m*-partition* $\Pi \in Part_m(\pi)$ *is the value*

$$c_w(\Pi) = \max_{\pi_r \in \Pi} \ell_w(\pi_r)$$

Algorithm 1: WRL-Partition

Input : G, π, ρ
Output: Π^ρ

$\Pi^\rho \leftarrow \varnothing, i, j \leftarrow 1$;
while $i \leq n$ **do**
  $i \leftarrow j$;
  $x_{(i)} \leftarrow x_{(j)}$;
  $\sigma \leftarrow \varnothing$;
  while $j \leq n$ **do**
    $cost \leftarrow (x_{(j)} - x_{(i)} + \ell_{0,(i)} + \ell_{(j),0}) \max\{w_{(i)}, \ldots, w_{(j)}\}$;
    if $cost \leq \rho$ **then**
      $\sigma \leftarrow \sigma \cup \{p_{(j)}\}$;
    else
      break
    $j \leftarrow j + 1$;
  if $\pi = \varnothing$ **then**
    return $\varnothing$
  $\Pi^\rho \leftarrow \Pi^\rho \cup \{\sigma\}$;
return Π^ρ

Denote by $\overline{\pi}$ the cycle formed by connecting the endpoints of a subpath π to the depot. The term $x_{(j)} - x_{(i)} + \ell_{0,(i)} + \ell_{0,(j)}$ corresponds to the length of such a cycle (see Figure 4). When a robot repeatedly traverses $\overline{\pi}$, this total length determines the constant revisit interval, and thus the idleness, for every constituent vertex. Consequently, the weighted cost in the above definition captures the worst-case weighted idleness by scaling this cycle length by the maximum vertex value found in π. Additionally, given $\Pi = \{\pi_1, \ldots, \pi_m\} \in Part_m(\pi)$, observe that:

- $\Theta_\Pi \coloneqq \{\overline{\pi_1}, \ldots, \overline{\pi_m}\}$ is an m–tour
- $c(\Theta_\Pi) \coloneqq c_w(\Pi)$

The polynomial-time algorithm called *left induced m-partition* and proposed in [28] finds the optimal m–partition $\Pi_{opt} \in Part_m(\pi)$ as the one minimizing the maximum length among its sub-paths, hence addressing a more specific variant of our problem where vertex weights are uniform. In Algorithm 1 we extend such a method by plugging into it our rooted and weighted setting encoded by the cost function of Definition 2 and hence allowing to find a m–partition $\Pi^* \in Part_m(\pi)$ such that $\Pi^* = \arg\min_{\Pi \in Part_m(\pi)} c_w(\Pi)$. We call this procedure *Weighted-Rooted Left-Induced m-partition* (WRL–Partition). Given a path π and a cost threshold ρ, the algorithm finds, if it exists, a partition Π^ρ such that every sub-path has a weighted cost at most ρ. The partition is essentially built greedily, by traversing π from left to right and closing a sub-path σ only when adding the next vertex would cause its weighted cost to exceed ρ.

If we abstract the size of the returned partition $|\Pi^\rho|$ as a function of the threshold ρ, denoted as $f(\rho) = |\Pi^\rho|$, we observe that $f(\rho)$ is monotone non-increasing and right-continuous. Intuitively, increasing the cost upper bound ρ

allows sub-paths to grow longer, thereby requiring fewer sub-paths to cover the entire sequence of vertices. The function has at most $|\pi| = n$ discontinuity points $\{\rho_1, \ldots, \rho_n\}$, satisfying:

$$\begin{cases} |\Pi^\rho| \leq k & \text{if } \rho \geq \rho_k \\ |\Pi^\rho| > k & \text{if } \rho < \rho_k \end{cases} \tag{2}$$

Also, notice that the following statement arises naturally: $\rho_m = \min\{\rho \mid |\Pi^\rho| \leq m\}$. In the following, we show that the optimal cost of a weighted m-partition ρ^* corresponds to one discontinuity point of the above function and specifically to ρ_m. This property is given by the following theorem, which is an extension of *Theorem 3.4* from [28].

Algorithm 2: WR-Partition

Input : G, π, ε
Output: Π^*

$\ell^0_{max} \leftarrow \max\{\ell_{0,(i)}\}$;
$a \leftarrow 0, b \leftarrow 2w_{max}(\frac{x_{(n)}}{m} + 2\ell^0_{max})$;
$\rho \leftarrow \frac{a+b}{2}$;
while $b - a > 2\varepsilon$ **do**
  $\Pi^\rho \leftarrow$ WRL-Partition(G, π, ρ);
  if $|\Pi^\rho| > m$ *or* $\Pi^\rho = \varnothing$ **then**
    $a \leftarrow \rho$;
    $\rho \leftarrow \frac{a+b}{2}$;
  else
    $\Pi^* \leftarrow \Pi^\rho$;
    $b \leftarrow \rho$;
    $\rho \leftarrow \frac{a+b}{2}$;
return Π^*

Theorem 1 (Optimal weighted rooted m-partition). *Consider a path $\pi = (p_{(1)}, \ldots, p_{(n)})$ with corresponding vertex weights $w_{(i)}$. Let $\Pi^* = \{\pi^*_1, \ldots, \pi^*_m\}$ an optimal weighted m–partition of π of cost $c_w(\Pi^*) = \rho^*$. Let $\Pi^\rho = \{\pi^\rho_1, \pi^\rho_2, \ldots\}$ be the weighted left–induced partition computed by Algorithm 1 for a given ρ. Then,*

$$\rho^* = \min\{\rho \in \mathbb{R}_+ \mid |\Pi^\rho| \leq m\}.$$

Proof. We prove that ρ^* verifies Condition 2, i.e, that $\rho^* = \rho_m$.

Consider $\rho < \rho^*$ and suppose by contradiction that $|\Pi^\rho| \leq m$. This implies that Π^ρ is an m–partition such that $c_w(\Pi^\rho) < \rho^*$, but this cannot be the case because Π^* is optimal.

Suppose now that $\rho \geq \rho^*$ and call $\pi^\rho_i = x^\rho_{i,F} \ldots x^\rho_{i,L}$, and $\pi^*_i = x^*_{i,F} \ldots x^*_{i,L}$ (with L and F we refer to the first and last vertices of their belonging path, respectively). We prove that $x^*_{i,L} \leq x^\rho_{i,L}$, $\forall i \in \{1, \ldots, m\}$, which implies that

$|\Pi^\rho| \leq m$ since $x^\rho_{1,F} = x^*_{1,F} = x_{(1)}$ and $x^*_{m,L} = x_{(n)}$. We prove it by induction on i. Now, notice that $\rho \geq \rho^*$ implies $\pi^*_1 \subseteq \pi^\rho_1$ by construction, which means that $x^*_{1,L} \leq x^\rho_{1,L}$. By consequence, $x^*_{2,F} \leq x^\rho_{2,F}$. Hence, suppose that $x^*_{i,L} \leq x^\rho_{i,L}$, which implies that $x^*_{i+1,F} \leq x^\rho_{i+1,F}$. If, by contradiction, $x^\rho_{i+1,L} < x^*_{i+1,L}$, then the path $\pi = \{x^\rho_{i+1,F}, \dots, x^*_{i+1,L}\}$ would be such that $l_w(\pi) > \rho$ since $\pi^\rho_{i+1} \subsetneq \pi$. But since $x^*_{i+1,F} \leq x^\rho_{i+1,F}$, this implies the following (L and F are implicitly referred to path π^*_{i+1}):

$$l_w(\pi^*_{i+1}) = (x^*_L - x^*_F + \ell^*_{0,L} + \ell^*_{0,F}) \max\{w^*_F, \dots, w^*_L\} \geq l_w(\pi) > \rho \geq \rho^*$$

which is impossible, since $\rho^* = \max_i\{l_w(\pi^*_i)\}$. Thus, the thesis follows. □

Algorithm 3: Wm–SPLITOUR

Input : G, ε
Output: Θ_{Π^*}
$C \leftarrow \text{TSP}(G)$; `// Christofides algorithm [20]`
$\pi \leftarrow C \setminus \{(0,\cdot),(\cdot,0)\}$;
$\Pi^* \leftarrow$ WR-Partition(G, π, ε);
$\Theta_{\Pi^*} \leftarrow \varnothing$;
for $\pi^*_i \in \Pi^*$ **do**
  $C_i \leftarrow \overline{\pi^*_i}$;
  $\Theta_{\Pi^*} \leftarrow \Theta_{\Pi^*} \cup \{C_i\}$;
return Θ_{Π^*}

Now, notice that for each discontinuity point ρ_i we have that $\rho_i \in \{l_w(\pi_r) : \pi_r \subseteq \pi\}$[2]. Also the function $\{l_w(\pi_r) : \pi_r \subseteq \pi\} \to \{l(\pi_r) : \pi_r \subseteq \pi\}$ is bijective and $|\{l(\pi_r) : \pi_r \subseteq \pi\}| \leq |\{\pi_r : \pi_r \subseteq \pi\}|$. As a consequence, to find the optimal solution it suffices to test only $|\{\pi_r : \pi_r \subseteq \pi\}| = \frac{n(n-1)}{2}$ different values of ρ with Algorithm 1. Theorem 1 extends the result of [28] providing an algorithm that finds an optimal weighted rooted m–partition in polynomial time. We report in Algorithm 2 the version of the algorithm that computes an $(1+\varepsilon)$–approximation of the optimal solution with time complexity $O(n^2 \log(\varepsilon^{-1}))$, using a similar bisection method as the one described in [28]. We call this algorithm *Weighted rooted m–partition* (WR–Partition). The convergence is straightforward after noticing that $\rho^* \in \big(0, 2w_{max}(\frac{x_{(n)}}{m} + 2\ell^0_{max})\big)$, where $\ell^0_{max} = \max_{1 \leq i \leq n} \ell_{0,i}$. Notice that the exact version of the algorithm, which replaces the bisection search of ρ^* with an iteration over all the $\frac{n(n-1)}{2}$ possible values of ρ, remains polynomial, with time complexity $O(n^4)$.

Algorithm 3 integrates Algorithm 2 in the m–SPLITOUR algorithm [21] to construct an approximate solution for the WmTSP. The *Weighted m–SPLITOUR* (Wm-SPLITOUR) works along these steps:

[2] With $\pi_r \subseteq \pi$ we mean a subpath of path π, not a generic subset of vertices or edges contained in π.

- finds an approximate TSP cycle C on the graph G.
- removes the cycle's edges that connect the depot, obtaining a path π that does not visit the depot, and apply the weighted rooted m-partition (Algorithm 2) on it obtaining $\mathit{\Pi}^* = \{\pi_1^*, \ldots, \pi_m^*\}$.
- forms the WmTSP solution by closing each sub-path π_i^* back to the depot: $\Theta_{\mathit{\Pi}^*} = \{\overline{\pi_1^*}, \ldots, \overline{\pi_m^*}\}$.

We now prove that Wm-SPLITOUR achieves the approximation factor stated in Observation 1, thus confirming that the theoretical guarantees of Frederickson's framework extend naturally to our method for the weighted generalization of the problem.

Theorem 2. *The* weighted m–SPLITOUR *is a polynomial–time algorithm that solves the WmTSP with depot with approximation factor*

$$\rho_{WmTSP} = \gamma \left(\rho_{TSP} + 1 - \frac{1}{m} \right)$$

Proof. Let Θ^* be the optimal solution of the WmTSP, i.e., $c(\Theta^*) = OPT_{WmTSP}$ and let us call C the approximate solution of a TSP instance of length L on the set V, i.e., $L \leq \rho_{TSP} OPT_{TSP}$. By the triangular inequality,

$$\frac{w_{min} L}{m} \leq \rho_{TSP} OPT_{WmTSP}.$$

Now, apply on C the *weighted m–SPLITOUR*, from which we have that $c(\Theta_{\mathit{\Pi}^*}) = c_w(\mathit{\Pi}^*) = \max_{1 \leq i \leq m} \ell_w(\pi_i^*)$, and $\mathit{\Pi}^*$ is optimal. It follows that

$$c(\Theta_{\mathit{\Pi}^*}) \leq w_{max} \left(\frac{L - 2\ell_{max}^0}{m} + 2\ell_{max}^0 \right),$$

where recall that $\ell_{max}^0 = \max_{1 \leq i \leq n} \ell_{0,i}$. Hence, the cost of the approximate solution $\Theta_{\mathit{\Pi}^*}$ is such that

$$\begin{aligned} c(\Theta_{\mathit{\Pi}^*}) &\leq \frac{w_{max}}{w_{min}} \left(\frac{w_{min} L}{m} + 2 w_{min} \ell_{max}^0 - \frac{2 w_{min} \ell_{max}^0}{m} \right) \\ &\leq \frac{w_{max}}{w_{min}} \left(\rho_{TSP} + 1 - \frac{1}{m} \right) OPT_{WmTSP}. \end{aligned}$$

The thesis follows. □

The time complexity of Algorithm 3 is bounded by the maximum time complexity between the Christofides algorithm [20] and Algorithm 2: $O(n^3)$ when using Algorithm 2 to construct the near-optimal weighted m-partition, or $O(n^4)$ when employing the exact version of the same algorithm.

Improving the current approximation factor presents a significant theoretical challenge, requiring either an advance on the (decades-old) mTSP bounds or a specialized analysis that reduces the factor when $\gamma > 1$ and aligns with the

approximation factor of the mTSP when $\gamma = 1$. A third way consists in moving away from the case $\gamma = 1$ and develop algorithms with better guarantees for the weighted case ($\gamma > 1$), yielding poor guarantees for the unweighted case ($\gamma = 1$). As this work intends to establish a primary constructive baseline, we defer these complex theoretical refinements to future research.

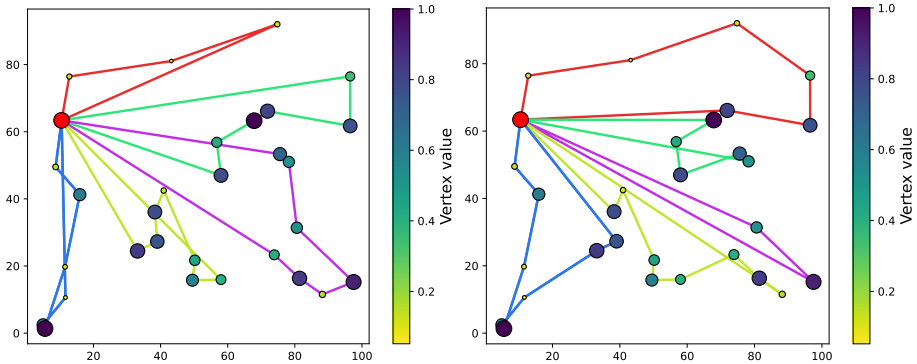

Fig. 5: Example of tours calculated by the m–SPLITOUR and the Weighted m–SPLITOUR, respectively, on the same 30–vertices graph instance with 5 robots. The depot vertex is evidenced in red. The worst weighted idleness of each tour in the 5–tour solution calculated by the m–SPLITOUR (left figure) are, approximatively, 18, 205, 198, 145, 131. The solution proposed by the Weighted m–SPLITOUR (right figure) is: 166, 171, 183, 178, 156.

5 Empirical Evaluation

We evaluate the *weighted m-SPLITOUR* presented in Section 4 as a better alternative for vertex-valued graphs to the state-of-the-art *m–SPLITOUR* method [21]. To the best of our knowledge, our method is the first approximation algorithm for solving the *multi–TSP with depot* on graphs that include vertex values. We therefore evaluate its performance against the *m–SPLITOUR*. The code and the experiments are publicly available at `https://github.com/alexbassot97/Weighted_mTSP`. For the experiment campaign, we generated 50 instances of vertex–valued graphs with $|V| = n \in \{30, 50, 100, 200, 400\}$ and randomly designated a vertex of each graph instance as the depot. Each vertex x_i was then randomly assigned a value $y_i \in (0, 1] \subset \mathbb{R}$, except for the depot vertex. Values were then normalized to guarantee at least one vertex with value 1. For each graph instance, both the Wm–SPLITOUR and the m–SPLITOUR were applied to construct the m–tours for teams of robots $\mathcal{R}$ with cardinality $|\mathcal{R}| = m \in \{3, 5, 10, 15, 20\}$. We call Θ_W and Θ_S an m–tour generated by the Wm–SPLITOUR and by the m–SPLITOUR, respectively, on the same graph instance.

$\lvert\mathcal{R}\rvert$ \ $\lvert V\rvert$	30		50		100		200		400	
3	1.00	1.34	1.00	1.28	1.01	1.24	1.01	1.14	1.00	1.12
5	1.00	1.50	1.03	1.41	1.03	1.33	1.04	1.25	1.00	1.34
10	1.00	1.35	1.05	1.35	1.03	1.43	1.06	1.36	1.09	1.29
15	1.00	1.18	1.01	1.29	1.07	1.39	1.09	1.44	1.08	1.37
20	1.00	1.14	1.00	1.23	1.01	1.36	1.05	1.44	1.13	1.37

Table 1: This table shows the highest (green) and lowest (blue) value of WI_{ratio} (Eq 3) calculated over the 50 graph instances, for each pair of number of vertices and number of robots.

See Figure 5 for a graphical example of the two solutions. The self-intersecting tours are due to the partitioning method adopted on the approximated TSP (which is calculated using the OR-Tools Routing Library [22]). Clearly, the tours could be improved without compromising the approximation factor by solving a TSP for each vertex partition together with the depot. We chose not to do this to ensure an unbiased comparison between the two methods.

As Figure 5 shows, one of the main features of the solutions calculated by the Wm–SPLITOUR is a better distribution of patrolling workload among the robots, proportional to the vertex with the highest value within the tour. The tour lengths are more unbalanced respect to the one provided by the m–SPLITOUR, allowing robots assigned to higher-value vertices to follow shorter tours. The consequence of the unbalanced tour lengths provided by the Wm–SPLITOUR is that, when evaluated under the worst weighted idleness metric (which scales tour lengths by the maximum vertex value within each tour) they outperform those computed by m–SPLITOUR.

To see this, we calculate the ratio between the costs of Θ_S and Θ_W:

$$WI_{ratio} = \frac{c(\Theta_S)}{c(\Theta_W)}. \tag{3}$$

Table 1 shows the lowest and highest values of WI_{ratio} for each combination of n and m, over all the 50 graph instances. The table empirically proves that on our large benchmark dataset Algorithm 3 is never worse than the m–SPLITOUR because the ratio is always equal or greater than 1 and may reach improvements of up to 50% (see for example the case of 30 vertices patrolled with 5 robots).

To provide a deeper analysis, we also evaluate the *average weighted idleness* over the 50 graphs for each combination of n and m:

$$AWI_S = \frac{1}{50}\sum c(\Theta_S), \quad AWI_W = \frac{1}{50}\sum c(\Theta_W).$$

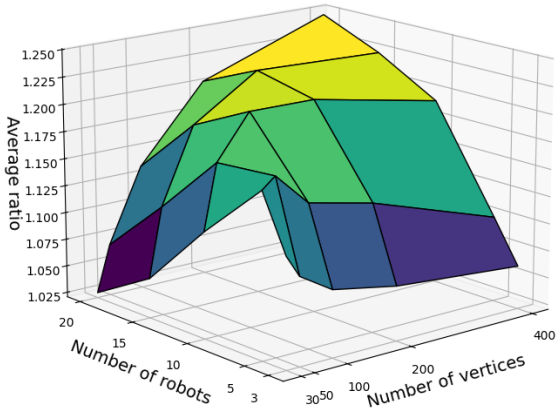

Fig. 6: Surface graph that shows how the average ratio as defined in Equation 4 changes when varying the number of robots and the number of vertices.

These values are then used to calculate the *average ratio*, that measures the advantage of applying the *weighted m–SPLITOUR* instead of the *m–SPLITOUR*, given different configurations of number of robots employed and cardinality of the set of vertices:

$$AWI_{ratio} = \frac{AWI_S}{AWI_W}. \tag{4}$$

As shown in Figure 6, the differences grow when both the number of robots and vertices increase. This means that our method becomes more and more advantageous as the problem becomes harder. Clearly, when the number of robots is similar to the number of vertices, the tour splits into balanced partitions in both methods because the clusters are made of few vertices. On the other hand, when the set of vertex is big and we employ a small set of robots, the average ratio degrades because there is more possibility of having high-valued vertices in all the length-balanced partitions provided by the m–SPLITOUR. Therefore Algorithm 3 outputs solutions that are similar to the ones provided by the m–SPLITOUR.

6 Conclusions and Future Work

In this work we introduced the *min-max weighted multi-TSP* problem for multi-robot patrolling and provided a first heuristic that solves it within an approximation bound, extending a well-known existing method. Future work is essentially twofold: one direction goes toward studying new heuristics for the WmTSP that would improve the approximation factor of the Wm-SPLITOUR algorithm; a second direction is to exploit the WR-Partition algorithm for other min-max problems that involve the construction of tours on graphs with vertex values.

References

1. Afshani, P., De Berg, M., Buchin, K., Gao, J., Löffler, M., Nayyeri, A., Raichel, B., Sarkar, R., Wang, H., Yang, H.T.: Approximation algorithms for multi-robot patrol-scheduling with min-max latency. In: Algorithmic Foundations of Robotics XIV (WAFR 2020). pp. 107–123 (2021)
2. Agmon, N., Kaminka, G.A., Kraus, S.: Multi-robot adversarial patrolling: facing a full-knowledge opponent. Journal of Artificial Intelligence Research **42**, 887–916 (2011)
3. Alamdari, S., Fata, E., Smith, S.L.: Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. The International Journal of Robotics Research **33**(1), 138–154 (2014)
4. Almeida, A., Ramalho, G., Santana, H., Tedesco, P., Menezes, T., Corruble, V., Chevaleyre, Y.: Recent advances on multi-agent patrolling. In: Brazilian Symposium on Artificial Intelligence. pp. 474–483 (2004)
5. Alpern, S., Chleboun, P., Katsikas, S., Lin, K.Y.: Adversarial patrolling in a uniform. Operations Research **70**(1), 129–140 (2022)
6. Alvarenga, C.D., Basilico, N., Carpin, S.: Multirobot patrolling against adaptive opponents with limited information. In: Proceedings of the IEEE Conference on Robotics and Automation. pp. 2486–2492 (2020)
7. Alvarenga, C.D., Basilico, N., Carpin, S.: Learning generalizable patrolling strategies through domain randomization of attacker behaviors. In: Proceedeings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 4406–4412 (2024)
8. Alvarenga, C.D., Basilico, N., Carpin, S.: Combining coordination and independent coverage in multirobot graph patrolling. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). pp. 4413–4419 (2024)
9. Arkin, E.M., Hassin, R., Levin, A.: Approximations for minimum and min-max vehicle routing problems. Journal of Algorithms **59**(1), 1–18 (2006)
10. Asghar, A.B., Smith, S.L.: A patrolling game for adversaries with limited observation time. In: 2018 IEEE Conference on Decision and Control (CDC). pp. 3305–3310 (2018)
11. Asghar, A.B., Sundaram, S., Smith, S.L.: Multirobot persistent monitoring: Minimizing latency and number of robots with recharging constraints. IEEE Transactions on Robotics **41**, 236–252 (2025)
12. Basilico, N., Carpin, S.: Balancing unpredictability and coverage in adversarial patrolling settings. In: Algorithmic Foundations of Robotics XIII (WAFR 2018). pp. 762–777 (2020)
13. Basilico, N.: Recent trends in robotic patrolling. Current Robotics Reports **3**(2), 65–76 (2022)
14. Basilico, N., Gatti, N., Amigoni, F.: Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In: Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS). pp. 57–64 (2009)
15. Basilico, N., Gatti, N., Amigoni, F.: Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. Artificial intelligence **184**, 78–123 (2012)
16. Bassot, A., Carpin, S., Basilico, N.: Approximation algorithms for cooperative multi-robot patrolling in core-periphery graph settings. In: 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE). pp. 2267–2274 (2025)

17. Camm, J.D., Magazine, M.J., Kuppusamy, S., Martin, K.: The demand weighted vehicle routing problem. European Journal of Operational Research **262**(1), 151–162 (2017)
18. Chen, L.H., Hung, L.J., Klasing, R.: Improved approximation algorithms for patrol-scheduling with min-max latency using multiclass minimum spanning forests. In: International Conference on Algorithmic Aspects in Information and Management. pp. 99–110 (2024)
19. Chevaleyre, Y.: Theoretical analysis of the multi-agent patrolling problem. In: Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology(IAT). pp. 302–308 (2004)
20. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. In: Operations Research Forum. vol. 3, p. 20 (2022)
21. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: 17th annual symposium on foundations of computer science (SFCS). pp. 216–227 (1976)
22. Furnon, V., Perron, L.: Or-tools routing library, `https://developers.google.com/optimization/routing/`
23. Guo, Q., Gan, J., Fang, F., Tran-Thanh, L., Tambe, M., An, B.: On the inducibility of stackelberg equilibrium for security games. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 2020–2028 (2019)
24. Heitzenrater, C., Taylor, G., Simpson, A.: When the winning move is not to play: Games of deterrence in cyber security. In: International Conference on Decision and Game Theory for Security GameSec. pp. 250–269 (2015)
25. Karlin, A.R., Klein, N., Gharan, S.O.: A (slightly) improved approximation algorithm for metric tsp. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. pp. 32–45 (2021)
26. Mor, A., Speranza, M.G.: Vehicle routing problems over time: a survey. Annals of Operations Research **314**(1), 255–275 (2022)
27. Pasqualetti, F., Durham, J.W., Bullo, F.: Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. IEEE Transactions on Robotics **28**(5), 1181–1188 (2012)
28. Pasqualetti, F., Franchi, A., Bullo, F.: On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. IEEE Transactions on Robotics **28**(3), 592–606 (2012)
29. Portugal, D., Rocha, R.: A survey on multi-robot patrolling algorithms. In: Doctoral conference on computing, electrical and industrial systems. pp. 139–146 (2011)
30. Scherer, J., Schoellig, A.P., Rinner, B.: Min-max vertex cycle covers with connectivity constraints for multi-robot patrolling. IEEE Robotics and Automation Letters **7**(4), 10152–10159 (2022)
31. Scherer, J., Rinner, B.: Multi-uav surveillance with minimum information idleness and latency constraints. IEEE Robotics and Automation Letters **5**(3), 4812–4819 (2020)
32. Tambe, M.: Security and game theory: algorithms, deployed systems, lessons learned. Cambridge University Press (2011)