Tracking Mobile Intruders in an Art Gallery: Guard Deployment Strategies, Fundamental Limitations, and Performance Guarantees

Guillermo J. Laguna · Sourabh Bhattacharya

1 Introduction

Security is an important concern in infrastructure systems. For decades, autonomous mobile robots have been utilized as surveillance [1][2], and crime-fighting agents for barrier assessments [3][4], intruder detection [5][6], building virtual terrains or maps [7][8], neutralizing explosives [9][10], and recognizing abnormal human behaviors [11][12]. Such robots have been designed with the ability to counter threats, limit risks to personnel, and reduce manpower requirements in hazardous environments [13][14][3]. Deployment of these surveillance platforms in teams has given rise to challenging problems in collaborative sensing and decision-making. Motivated from recent surge in interest in autonomous surveillance, we address an asset protection problem in which a team of mobile sensors collaborate to track suspicious mobile entities to secure an environment.

Although advanced electronic and biometric techniques can be used to secure facilities, vision-based monitoring is widely used for persistent surveillance. The idea is to visually cover the environment in order to obtain sufficient information so that appropriate measures can be taken to secure the area in case of any suspicious/malicious activity. The general formulation of a tracking problem consists of a team of autonomous sensing platforms, called observers, that visually track mobile entities, called targets. In this work, we consider the aforementioned formulation in the presence of features in the environment that can occlude the targets from the observers, for example, presence of reflex vertices on the boundary of the environment and obstacles. Precocious planning and coordination between observers can prevent the intruders from escaping the visual footprint of the observers around such occlusions.
Deploying a network of autonomous sensing platforms has been an active area of research in multi-robot systems (MRS). Multi-robot systems have emerged as an important area of research in robotics due to their potential applications in several areas, for example, autonomous sensor networks [15], building surveillance [16], transportation of large objects [17], air and underwater pollution monitoring [18], forest fire detection [19], transportation systems [20], or search and rescue after large-scale disasters [21]. Even problems that can be handled by a single multi-skilled robot may benefit from the alternative usage of a robot team, since robustness and reliability can often be increased by combining several robots which are individually less robust and reliable. In case of target tracking, multiple points of view from multiple robots add extra information on the target resulting in a better estimate of its position. However, the uncertainty in the future actions of the target, and the tight coupling between sensing, coordination and control within the team of mobile observers gives rise to challenging problems in multi-robot motion planning.

A simple solution to the tracking problem in bounded environments is to cover the environment with sufficient number of observers. This leads to the art gallery problem, a well-studied topic in computational geometry. In the classical art gallery problem, the goal is to determine the number of guards needed to visually cover a bounded polygon [22]. A guard can only see the portion of the polygon unobstructed by the boundary of the polygon or by internal obstacles. Over the years several variants of the problem have been studied based on the shape of the polygon (orthogonal [23,24,25,26], monotone [27], etc), the type of guards employed (static [28,29,30,31], mobile [32,33,34,35], and the notion of visibility (k-transmitters [36], multi-guarding [37,38] etc). Stationary guards can either be point guards (placed anywhere inside the polygon) or vertex guards (restrict them only to the vertices of the polygon). It has been shown that the problem of computing the minimum number of stationary guards required to cover a simply connected polygon is NP-hard [39]. In [28], it is shown that \( \lceil n/3 \rceil \) static guards with omni-directional field-of-view is sufficient, and sometimes necessary to cover the entire polygon, where \( n \) is the number of vertices of the polygon representing the environment. Efforts to obtain a bound on the minimum number of guards required to cover a polygon for special cases include approximation [40] and heuristic techniques [41].

The notion of mobile guards was first introduced by Toussaint [33] where each guard can travel back and forth along a segment inside the polygon, and every point in the polygon must be seen by at least one guard at some point of time along its path. Since the guards do not visually cover the entire polygon at all times, this notion of coverage is called mobile coverage. In [22] it is proved that \( \lceil n/4 \rceil \) diagonal guards are sufficient to provide mobile coverage. Therefore, if the guards have sufficient speed, at most \( \lceil n/4 \rceil \) of them are required to track a mobile intruder in a polygonal environment. This naturally leads to the following question: What is the minimum speed required for the diagonal guards to track a mobile intruder in the environment with known maximum speed? The answer depends on the coordination and control strategy used by the guards.

The need for coordination arises in multi-robot systems (MRS) when individual agents work together to complete a common task [42]. Depending on the communication scheme adopted by the MRS, coordination can be explicit or implicit. As the name suggests, explicit communication requires the presence of an on-board communication module on each robot of the MRS. On the other hand, implicit communication is generally contextual, based on the sensor data available to a robot regarding its teammates. Due to significant improvements in mobile communication hardware, the coordination scheme prevalent in current MRSs is predominantly explicit in nature. In this work, we propose an explicit coordination scheme for the team of observers. Our scheme explicitly relies on the location of the observers relative to each other thereby building a connection between the geometric aspects of the team formation, and communication topology required for persistent tracking.

In this work, we formulate the tracking problem as a multi-robot task allocation problem (MTAP) [43]. The allocation tries to balance the workload among the observers thereby minimizing the speed required to ensure tracking. The main contributions of this work are as follows:

1. We investigate a variant of the multi-robot target tracking problem in which observers are constrained to move along diagonals of the polygonal environment. We leverage guard deployment strategies proposed for mobile coverage [32] to design deployment and tracking strategies for multiple observers. To the best of our knowledge, our work is the first to build a connection between the well known art gallery problem which deals with coverage and multi-robot tracking.

2. We propose an algorithm that jointly partitions a polygon, and allocates observers to track a target inside the partition. The algorithm is a solution to a resource allocation problem in which the observers are resources for tracking that need to be assigned
to partitions of the polygon. The joint partitioning and allocation algorithm is based on the triangulation of the polygon, and therefore, can be extended to polygon with holes. Under the restriction that a single observer can be allocated to each partition, we show that the problem of finding the minimum observer speed required to track an intruder is NP-hard.

3. We present a taxonomy of the observers based on their task allocation, and the level of cooperation needed from other members of the team to perform the tracking task. We introduce the notion of critical curves to construct activation strategies for the mobile observers.

4. We derive the maximum number of targets that can be tracked using the deployment and partitioning algorithm proposed in this work.

The paper is organized as follows. Section 2 presents a brief description of the related work. Section 3 presents the problem formulation and a deployment strategy for the guards. Section 4 formulates the tracking problem as a multi-robot task allocation problem. Section 5 presents a partitioning technique for the polygon, and an allocation algorithm to assign the partitions to the guards to track a single intruder in the environment. Section 6 addresses the case of non-simple polygons. Section 7 extends the proposed deployment and allocation algorithm to address the case of multiple intruders in the environment. Section 8 presents the conclusions and future work.

2 Related Work

The multi-robot target-tracking problem was originally introduced by Parker and Emmons in [44]. The authors proposed the framework of Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) for a team of observers that simultaneously maximize the number of targets under observation, and the duration of observation of each target. It is shown that the CMOMMT problem is NP-hard. Consequently, several variants of CMOMMT have been studied to propose implementable solutions, for example, Approximate CMOMMT [45], personality-CMOMMT [46], behavioral-CMOMMT [47,48], formation-CMOMMT [49], to name a few. Alternate formulations of the CMOMMT based on particle filtering [50] and mixed non-linear integer programming formulation [51] have also been proposed in the past. These frameworks lead to a numerical approach for generating the trajectory of the observers.

In the past, several variants of the original tracking problem posed in [41] have been addressed. In [52], the authors analyze the focus of attention problem [53] for specific formations of the observers, and provide approximation algorithms for optimal target allocation. Dames et al. address the problem of detecting, localizing and tracking a team of targets with unknown and time-varying cardinality [54]. In [55], the authors address the cooperative control of a team of UAVs that try to solve the joint problem of self localization and tracking with on-board sensors. In [56], a region-based coarse approach is proposed to simultaneously observe several mobile targets. The observers use a local method to maximize the number of observed targets. The proposed technique does not require any communication among the robots. In [57], CMOMMT is addressed for cooperative targets. The targets are able to communicate which allows active cooperation through sharing data. Thus the average observation time is minimized through a force field approach. In [58], a team of winged UAVs tries to minimize the average time elapsed between two consecutive observations of each member of a group of targets. An optimal control scheme is used to obtain the motion strategy of the observers.

In this work, we formulate the problem of tracking as a multi-robot task allocation problem. Multi-robot task allocation (MRTA) can be considered as an instance of the well-known optimal assignment problem. In [59], algorithms to solve the matching problem for weighted bipartite multi-graphs are used to solve MTAP. In [60], a domain-independent taxonomy of multi-robot task allocation problems is presented. They also analyzed and compared some iterated assignment architectures: ALLIANCE [61], BLE [49], and M+ [62] and some on-line assignment architectures: MURDOCH (auction-based MRTA) [63], first-price auctions (market-based MRTA) [64] and dynamic role assignment [65], for MRTA, respectively. In [66], the authors present a mathematical modeling and analysis of the collective behavior of dynamic task allocation. In [67], a lightweight and robust multi-robot task allocation approach based on trade rules in market economy is presented. Other strategies are based on vacancy chains [68], auction-based mechanisms [69] and distributed market-based coordination [70].

In this work, we assume that the observers are omnidirectional cameras that slide along the diagonals of the polygon. In [24], the authors introduce the problem of placing sliding cameras on the boundary of the polygon for mobile coverage. Unlike the standard pinhole camera model, a sliding camera in [24] can see a point $p_1$ inside the environment if there exists a point $p_2$ along its trajectory such that the segment $p_1p_2$ is
perpendicular to the trajectory of the guard and \( p_1p_2 \) is fully contained in the environment. The objective in [24] is to find the minimum number of sliding cameras that can provide mobile coverage, referred to as the minimum sliding camera (MSC) problem. This problem is a variant of the art gallery problem which is NP-hard even for orthogonal polygons [72]. However, it has been proven to be APX-hard on simple polygons [73]. Polynomial time approximation algorithms have been developed for orthogonal polygons [24][25][26]. In [71], it is shown that MSC is NP-hard for non-simple polygons. For monotone orthogonal polygons, a linear time solution to MSC is presented in [27]. In [71], the minimum length sliding camera problem is introduced. The objective is to find a set of sliding cameras that minimizes the total length of the trajectory while providing mobile coverage [74]. A polynomial time solution for orthogonal polygons with holes is presented in [71].

### 3 Problem Statement

Let \( P \) be a simple \( n \)-vertex polygon representing a closed polygonal environment. First, we consider the case of a single unpredictable target \( I \), referred to as the intruder, inside \( P \). In Section 7, we extend our analysis to address multiple intruders inside the polygon. Let \( \bar{v} \) and \( x_t \) denote the maximum speed of the intruder and its location respectively at time \( t \). In order to track the mobile intruder, a team of mobile observers, referred to as guards, is deployed inside the polygon. Let \( G \) denote the set of all guards. Each guard is equipped with an omni-directional camera with infinite range. The objective of the guards is to ensure that \( I \) is visible to at least one guard at all times. The guards have knowledge of \( \bar{v}_c \). Each guard has a maximum speed \( \bar{v}_g \) and the speed ratio between the guards and the intruders is defined as \( r = \bar{v}_g/\bar{v}_c \).

Next, we define some graph-theoretic concepts associated with a polygon and its partitioning. A triangulation of \( P \) is defined as a partition of \( P \) into a set of disjoint triangles, such that the vertices of the triangles are vertices of the polygon. In general, the triangulation of a polygon is non-unique. The edges of the triangles of the triangulation are called diagonals [32], and they can be segments inside \( P \) (internal diagonals) or edges of \( P \). Thus, the triangulation of \( P \) is trivially represented as a planar graph \( G = G(P) \) called triangulation graph. We assume that a triangulation of \( P \) is given. Let \( V(G) \) denote the vertex set of \( G \) (corresponds to the vertices of \( P \)), \( E(G) \) denote the edge set of \( G \) (corresponds to the diagonals of the triangulation of \( P \)), and \( T(G) \) (triangle set of \( P \)) denote the faces of \( G \). Clearly, there is a bijection between the set of vertices of \( P \) and \( V(G) \). Also, there is a bijection between the set of diagonals of the triangulation of \( P \) and \( E(G) \). Hence, we do not make any distinction between the following pairs: (i) vertices of \( P \) and the vertices in \( V(G) \) (ii) diagonals of the triangulation of \( P \) and the edges in \( E(G) \) (iii) triangles of the triangulation of \( P \) and the faces in \( T(G) \).

Let \( G_D \) be the dual graph of \( G \). Each vertex in \( G_D \) corresponds to a face in \( T(G) \). An edge exists between a pair of vertices in \( G_D \) if the triangles in \( T(G) \) that correspond to such pair of vertices share an edge. Since \( P \) is simply connected, \( G_D \) is a tree.

The objective is to keep the intruder in the line-of-sight of at least one guard at all times, which is ensured if the triangle in which the intruder is located is covered by at least one guard. Each \( G_i \) in \( G \) is a diagonal guard, i.e., it is constrained to move along a unique diagonal \( h_i \in \mathbb{H} \), where \( \mathbb{H} \subseteq \mathbb{E}(G) \). Let \( l_i \) be the length of \( h_i \). The endpoints of \( h_i \) are denoted as \( v_1(i) \) and \( v_2(i) \). Guards \( g_i \) and \( g_k \) are neighboring guards if \( g_i \) and \( g_k \) are incident to the same triangle. We define \( G(T_k) \subseteq G \) as the set of guards incident to \( T_k \).

In this paper, the distance between two points \( x, y \in P \) is defined as the length of the shortest path between \( x \) and \( y \) on the visibility graph \( G \) constructed using the vertices of \( P \) (and the vertices of internal obstacles), \( x \) and \( y \). It is denoted as \( d(x, y) \). The distance between \( x \in P \) and a set \( R_i \subset P \) is defined as \( d(x, R_1) = \min\{d(x, p) : p \in R_1\} \). The distance between two sets of points \( R_1, R_2 \subset P \) is defined as \( d(R_1, R_2) = \min\{d(q, R_1) : q \in R_2\} \).

**Note:** Appendix C contains a list of important variables, and their definitions.

In the next section, we describe the deployment of the guards inside the polygon.

----

1. In this work, we define a triangle to be covered if and only if there is a guard located at the boundary of the triangle.
2. We say that a guard \( g_i \) is incident to triangle \( T \) if at least one of the endpoints of \( h_i \) is a vertex of \( T \).
3. A visibility graph of a polygon is a graph whose nodes correspond to the vertices of the polygon, and there is an edge between any two vertices if the segment joining them is contained inside the polygon.
3.1 Selection of Dominating Diagonals

In [28], it is shown that at most \([n/4]\) diagonals are sufficient to dominate \(G\). Hence, \([n/4]\) diagonal guards are sufficient to provide mobile coverage. In this section, we describe the strategy proposed in [28] to identify at most \([n/4]\) dominating diagonals \(d\) of a polygon’s triangulation. In [28], it is shown that there exists a set of dominating diagonals of size \([n/4]\) in every triangulation graph of a polygon with \(n \geq 5\). The correctness of the strategy is based on the following results [28]: (i) For any triangulation graph \(G\) of a simple polygon \(P\) with \(n \geq 10\) edges, it is always possible to find a diagonal \(d \in \mathcal{E}(G)\) that partitions \(G\) into subgraphs \(G_1\) and \(G_2\) such that \(G_1\) is the triangulation graph of a hexagon, heptagon, octagon or nonagon (we call these basic polygons). (ii) Any triangulation graph of a heptagon (or any polygon with fewer sides) has one dominating diagonal, and any triangulation graph of an octagon or nonagon has two dominating diagonals.

Algorithm 1 recursively partitions \(G\) into triangulation subgraphs of basic polygons. At each iteration, the algorithm searches for a diagonal \(d\) that separates a triangulation subgraph of a basic polygon (denoted as \(G_p\)) such that there is no other diagonal in \(\mathcal{E}(G_p)\) (edge set of \(G_p\)) that can separate a subgraph of a smaller basic polygon. Additionally, for each subgraph \(G_p\) obtained, a minimal set of diagonals \(\mathcal{E}(G_p)\) is found such that the diagonals in the set can dominate all the triangles \(f\) in \(T(G_p)\) (set of faces of \(G_p\)) that are still not dominated. The process is repeated until the remaining non-partitioned subgraph has 9 vertices or less. The first while cycle (Line 5) finds a diagonal \(d\) that partitions \(G\) into a pair of triangulation subgraphs, such that one of those subgraphs \(G_p\) corresponds to a basic polygon and such that there is no other diagonal in \(\mathcal{E}(G_p)\) (edge set of \(G_p\)) that can separate a subgraph of a smaller basic polygon. This can be completed in \(O(n)\) time by traversing \(G_D\) (dual graph of \(G\)).

\[G_{pol}\] (Line 10) is a tree such that each \(v \in \mathbb{V}(G_{pol})\) is associated with each subgraph \(G_p\) found in Line 6, and one vertex corresponds to the remaining subgraph \(G'\) after Line 9. An edge \(e \in \mathcal{E}(G_{pol})\) exists between vertices that correspond to a pair of triangulation subgraphs \(G_p\) that share a common diagonal \(d \in \mathcal{E}(G)\). In the second while loop the minimum set of diagonals that can cover the triangles in \(T(G_i)\) that are not dominated by other diagonals is found (set of appropriate diagonals, Line 14). The second While loop takes \(O(n)\) time by using \(G_{pol}\). Hence, Algorithm 1 takes \(O(n)\) time.

Algorithm 1 Guard Deployment

1. \textbf{Input:} \(G\)
2. \textbf{Output:} \(H\)
3. \(S_D \leftarrow \emptyset\) is the set of diagonals \(d\)
4. \(G' \leftarrow G\)
5. \textbf{while} \(G'\) has \(n \geq 10\) vertices \textbf{do}
6. \quad find \(d\) that separates a triangulation subgraph \(G_p\) from \(G'\) such that there is no triangulation subgraph of \(G_p\) that corresponds to a smaller basic polygon
7. \quad \(G'\) becomes the subgraph obtained by removing \(G_p\) excepting \(d\) and its vertices
8. \quad add \(d\) to \(S_D\)
9. \textbf{end while}
10. \(G_{pol}\) from \(G\) using the diagonals in \(S_D\) (all vertices in \(\mathbb{V}(G_{pol})\) are unmarked)
11. \textbf{while} there is an unmarked vertex in \(G_{pol}\) \textbf{do}
12. \quad \(v_i \leftarrow\) unmarked vertex in \(G_{pol}\) with at most one unmarked neighbor
13. \quad \(G_i \leftarrow\) subgraph of \(G\) that corresponds to \(v_i\)
14. \quad \(H_i \leftarrow\) appropriate diagonals of \(T(G_i)\)
15. \quad add diagonals of \(H_i\) to \(\mathbb{H}\)
16. \quad mark \(v_i\)
17. \textbf{end while}

We call \(H\) a deployment of the guards, since each \(g_i \in G\) is assigned to a diagonal \(h_i \in H\) (it is deployed along the diagonal). In the next section, we address the problem of finding the minimum speed of the guards that can ensure tracking.

4 A Multi-robot Task Allocation Problem

A sufficient condition to track the intruder is to cover at all times the triangle in which it lies. In the previous section, we described an algorithm to select at most \([n/4]\) diagonals of a polygon that can dominate the triangles of the triangulation of a polygon. Given sufficient speed, the guards assigned to the diagonals in \(H\) can move back and forth on their diagonals to cover the triangle in which the mobile intruder lies. This gives rise to a MRTA problem in which the task for each robot is to cover the triangles allocated to it whenever the intruder lies inside them.

First, we consider the case in which a single guard is allocated to a triangle of the triangulation. An allocation \(A : \mathbb{G} \rightarrow 2^{\mathbb{T}(G)}\) maps each guard to a subset of triangles in the triangulation of the polygon. Let \(\mathcal{A}\) denote the set of all possible allocations. We say that an allocation is complete if there is no triangle with no guard allocated to it. Next, we classify the triangles of the triangulation of \(P\) (the triangles in \(\mathbb{T}(G)\))
based on the number of incident guards and their position relative to the triangles. Refer to Figure 1. The red segments in the figure represent the diagonals $h_i \in \mathbb{H}$ on which the guards move. In the subsequent figures, those diagonals are labeled as $g_i$ instead.

1. Safe Triangle: A triangle $T_k \in \mathbb{T}(G)$ is called safe if $g_i$ covers $T_k$ at all times regardless of the location of $x_I$. Clearly, if there is a guard $g_i \in \mathbb{G}$ deployed on one of the edges of $T_k$, the triangle is a safe one. We use $\overrightarrow{T}_G$ to denote the set of safe triangles. In Figure 1, safe triangles are shaded in blue.

2. Unsafe Triangle: A triangle $T_k \in \mathbb{T}(G) \setminus \overrightarrow{T}_G$ is called unsafe if $|\mathbb{G}(T_k)| = 1$, where $|\cdot|$ is the cardinality operator. In Figure 1, unsafe triangles are shaded in orange.

3. Regular Triangle: A triangle $T_k$ is regular if it is neither safe nor unsafe. In Figure 1, regular triangles are shaded in white.

Fig. 1: Classification of a set of triangles in $\mathbb{T}(G)$. Notice that the regular triangles $T_1$ and $T_2$ can be covered by more than one guard.

Let $\overrightarrow{T}_G(g_i)$ denote the set of triangles that are incident to the endpoint $v_{\alpha}(i)$ of $h_i$ ($\alpha \in \{1, 2\}$). Also, let $\overrightarrow{T}_G(G) = (\mathbb{T}(G) \setminus \overrightarrow{T}_G(G))$ denote the set of non-safe (Regular+Unsafe) triangles in $\mathbb{T}(G)$ and $\overrightarrow{T}_G(g_i)$ denote the set of non-safe triangles that are incident to $v_{\alpha}(i)$. By definition, $\overrightarrow{T}_G(g_i)$ is always covered by $g_i$ for every allocation $A(\mathbb{G}) \in \mathcal{A}$. We want to assign a $g_i \in \mathbb{G}$ to every $T \in \overrightarrow{T}_G(G)$ so that $g_i$ can cover $T$ when the intruder lies in it. Clearly, only the guards in $\mathbb{T}(G)$ can be assigned to $T$. The deployment strategy proposed in Section 3.1 guarantees that $|\mathbb{G}(T)| \geq 1$ for every $T \in \overrightarrow{T}_G(G)$. Therefore, an allocation that assigns a guard to each triangle in $\mathbb{T}(G)$ exists. Moreover, every unsafe triangle can only be assigned to the single guard incident to it.

Given an allocation $A(\mathbb{G})$, there are two possibilities for each $g_i \in \mathbb{G}(T)$:

1. All the non-safe triangles assigned to $g_i$ are in $\overrightarrow{T}_G(g_i)$. In this case, $g_i$ is static, and can cover all the triangles assigned to it from $v_{\alpha}(i)$ regardless of the speed ratio $r$.

2. There are triangles assigned to $g_i$ in $\overrightarrow{T}_G(g_i)$ and $\overrightarrow{T}_G(g_i)$. This implies that $g_i$ needs to be at $v_1(i)$ ($v_2(i)$) when the intruder $I$ lies in $\bigcup_{T \in \overrightarrow{T}_G(g_i)} T$ ($\bigcup_{T \in \overrightarrow{T}_G(g_i)} T$). Therefore, $g_i$ has to move from one endpoint of $h_i$ to the other depending on $x_I$. Assume that $I$ is initially located in any triangle $T_j \in \overrightarrow{T}_G(g_i)$ assigned to $g_i$, and $I$ moves to a triangle $T_k \in \overrightarrow{T}_G(g_i)$ also assigned to $g_i$. As a result, the guard $g_i$, initially located at $v_1(i)$, moves to $v_2(i)$. Therefore, $g_i$ should reach $v_2(i)$ before the intruder can reach $T_k$. If the aforementioned condition is satisfied for every $T_j \in \overrightarrow{T}_G(g_i)$ and every $T_k \in \overrightarrow{T}_G(g_i)$ assigned to $g_i$, then $g_i$ can track $I$ when it is inside any triangle allocated to it.

To summarize the above discussion, an allocation $A(\mathbb{G})$ should satisfy the following conditions:

1. $A$ is complete.

2. Unsafe triangles are allocated to the single guard incident to them.

3. $\bar{v}_g \geq \max \bar{v}_c l_i / d(T_{safe}(g_i))$, where $T_{safe}(g_i) \subset \overrightarrow{T}_G(g_i) \subset T_{safe}(g_i)$ are the sets of triangles allocated to $g_i$ that correspond to $v_1(i)$ and $v_2(i)$ respectively.

An allocation $A$ is called feasible if it satisfies the above conditions. Given the constraint that a single guard can be allocated to a non-safe triangle, we pose the following problem:

**Problem 1:** Given $\bar{v}_c$, what is the minimum value of $\bar{v}_g$ and the corresponding allocation $A$ for which the intruder can be persistently tracked?

$G_D$ encodes the adjacency between faces of $G$ based on common diagonals. Analogous to a dual graph, we define a Guard Adjacency Graph (GAG), denoted as $G^\#$, which encodes the adjacency between faces of the triangulation graph $G$ based on common guards. For a given triangulation graph $G$ of the polygon and deployment of the guards, $G^\#$ is constructed as follows. Each vertex of $G^\#$ corresponds to a non-safe triangle in $G$. An edge $e_{j,k}(g_i)$ exists between vertices $v_j$ and $v_k$ in $G^\#$ if there exists a guard $g_i$ such that $T_j \in \overrightarrow{T}_G(g_i)$ and $T_k \in \overrightarrow{T}_G(g_i)$. The weight of the edge $e_{j,k}(g_i)$ is given by $w_{j,k}(g_i) = l_i / d(T_j, T_k)$.

Next, we define terms related to allocation of guards to triangles. Given an allocation $A(\mathbb{G}) \in \mathcal{A}$,
let \( A(g_i) \) be the set of triangles assigned to \( g_i \). Then we define the cost of the allocation of \( g_i \) as 
\[
c(A(g_i)) = \max_{T_j, T_k \in A(g_i)} \{ w_{j,k}(g_i) \}
\]

The overall cost of the allocation for \( G \) is defined as 
\[
c(A(G)) = \max_{g_i \in G} \{ c(A(g_i)) \}.
\]

Based on these definitions, Problem 1 can be formulated as the following problem on \( G \):

**UNIALLOC**: Find a feasible allocation \( A(G) \in A \) for which \( c(A(G)) \) is minimized.

In order to prove the above theorem, first we define a set of representatives from \( G^3 \). For each non-safe triangle \( T_j \in \text{safe}(G) \), we define a set \( S_0(T_j) = \{(j, g_i) : g_i \in \mathcal{G}(T_j)\} \). \( S_0(T_1), \ldots, S_0(T_{|\mathcal{V}(G)|-1}) \) is a collection of disjoint sets. For any \( x = (j, g_i) \in S_0(T_j) \) and \( y = (k, g_0) \in S_0(T_k) \) with \( j \neq k \), define \( c_2(x, y) \) as follows:

\[
c_2(x, y) = \begin{cases} w_{j,k}(g_i), & a = b \\ 0, & \text{otherwise.} \end{cases}
\]

Given sets \( S_j \) and the cost function \( c_3 \) stated above, we prove that a \( \lfloor V(G^3) \rfloor \)-clique in \( G_2 \) exists if and only if a set of \( \lfloor V(G^3) \rfloor \) representatives \( S_{\text{rep}} \subset \bigcup_{j \in \{1, \ldots, |\mathcal{V}(G^3)|\}} S_j \) such that \( c_3((j, v), (k, w)) = 0 \) for all \( (j, v), (k, w) \in S_{\text{rep}} \) can be found.

\( \Rightarrow \) First, assume that there exists such a \( S_{\text{rep}} \), but there is no \( \lfloor V(G^3) \rfloor \)-clique in \( G_2 \). By definition, for each pair \( (j, v), (k, w) \in S_{\text{rep}} \), \( j \neq k \), and \( v \neq w \) (\( v = w \) implies that \( G_2 \) has a self-loop which is not possible). Therefore, there is at least one pair \( (j, v), (k, w) \in S_{\text{rep}} \) such that \( c_3((j, v), (k, w)) = 0 \) for all \( (j, v), (k, w) \in S_{\text{rep}} \), then \( \lfloor V(G^3) \rfloor \)-clique in \( G_2 \) exists.

\( \Leftarrow \) Now assume that there is no \( S_{\text{rep}} \) which contains the aforementioned constraints, but there is a \( \lfloor V(G^3) \rfloor \)-clique in \( G_2 \). Since \( G_2 \) contains a \( \lfloor V(G^3) \rfloor \)-clique, there is a subset of \( \lfloor V(G^3) \rfloor \) vertices such that its induced subgraph \( G_2 \) is complete. Since \( G_2 \) is complete, choosing \( (j, v_j) \in S_j \) for each \( v_j \in G_2 \) as an element of \( S_{\text{rep}} \) implies that for any \( (j, v), (k, w) \in S_{\text{rep}} \), an edge exists between \( v_j \) and \( v_k \) in \( G_2 \). Therefore, \( c_3(v_j, v_k) = 0 \) for all \( (j, v), (k, w) \in S_{\text{rep}} \), which contradicts the assumption that there is no \( S_{\text{rep}} \) of size \( |\mathcal{V}(G^3)| \) such that \( c_3((j, v), (k, w)) = 0 \) for all \( (j, v), (k, w) \in S_{\text{rep}} \).

Thus, we have a polynomial-time reduction from \( K \)-clique to GAMMAREP.

**Lemma 1** GAMMAREP is NP-hard.

**Proof** We reduce the problem of finding a \( K \)-clique \([75]\) in a graph (a NP-hard problem \([70]\)) to GAMMAREP. Consider a graph without self-loops \( G_2 \) such that \|\(\mathcal{V}(G_2)\| \geq |\mathcal{V}(G^3)|\). The problem of finding a \( |\mathcal{V}(G^3)| \)-clique in \( G_2 \) is NP-hard \([77]\). For each \( v_j \in \mathcal{V}(G_2) \), we define a set \( S_j = \{(j, v_j, v_k) \in \mathcal{V}(G_2) \} \) for a pair of vertices \( v, w \in \mathcal{V}(G_2) \), we define the following cost function:

\[
c_3((j, v), (k, w)) = \begin{cases} 0, & \text{if an edge exists between } v \text{ and } \text{in } G_2 \\
1, & \text{otherwise.} \end{cases}
\]
4.1 Suboptimal Algorithm for Computing Approximate minimum speed ratio

MAXCLIQUE refers to the problem of finding the maximum-sized clique in a graph. MAXCLIQUE is NP-hard [28]. An approximation algorithm for MAXCLIQUE can be used to obtain a suboptimal solution for Problem 1. The procedure is as follows. Let $G_3$ be a graph such that $V(G_3) = \{v_j : \exists S_g(T_j) \text{ such that } (j, g_i) \in S_g(T_j)\}$. Recall that $S_g(T_j) = \{(j, g_i) : g_i \in G(T_j)\}$ for all $1 \leq j \leq |V(G^g)|$. Since $|V(G^g)| = O(n)$ ($|V(G^g)| \leq n - 2$), and $1 \leq |S_g(T_j)| \leq |G| = O(n)$, $|S_g(T_j)| = O(n)$. Therefore, $|V(G^g)| = O(n^2)$. Additionally, let $E(G^g) = \{e_{j,k} : v^a_j, v^b_k \in V(G^g) \text{ with } j \neq k\}$. Each edge $e_{j,k}$ has an associated weight defined as follows:

$$w_{j,k}^{a,b} = \begin{cases} 0, & \text{if } a \neq b \text{ or } \frac{1}{\sqrt{d_{j,k}}} \leq r \\ 1, & \text{otherwise} \end{cases}$$

We define $G_3'$ as a subgraph of $G_3$ such that $V(G_3') = V(G_3)$ and $E(G_3') = \{e_{j,k}^{a,b} \in E(G_3) : e_{j,k}^{a,b} = 0\}$. Therefore, $E(G_3')$ consists of edges in $E(G_3)$ with weights less than or equal to $r$. We can use any approximation algorithm for MAXCLIQUE to find a $|V(G_1)|$-clique in $G_3'$ for a given $r$. Since $r_{\text{min}} = \min\{r = \frac{1}{\sqrt{d_{j,k}}} : e_{j,k}(i) \in G^g\}$, we perform the aforementioned check for values of $r$ that correspond to the weights of the edges in $G^g$.

The suboptimal allocation corresponds to the minimum value of $w_{j,k}(g_i)$ for which a $|V(G_1)|$-clique is found in the graph $G_3'$. If no such $w_{j,k}(g_i)$ is found, $r_{\text{min}}$ is equal to the minimum $w_{j,k}(g_i)$. Each vertex $v^j_j \in V(G_3')$ in the $|V(G_1)|$-clique corresponds to a distinct triangle in $T_j \in \mathbb{G}^{\mathbb{G}}$, and each one of them corresponds to a $g_i \in \mathbb{G}$ which is the guard assigned to $T_j$. This gives an approximate optimal allocation $A(g_i)$ for each $g_i \in \mathbb{G}$.

It has also been shown that approximating the MAXCLIQUE within a factor of $n^\epsilon$ for some $\epsilon > 0$ is NP-hard [79,80]. In the past, probabilistic techniques [81,82,83,76] have been proposed in the literature to find largest cliques without any guarantees on the optimality. [84] presents an approximation algorithm that finds a clique with an approximation ratio of $O(n(\log \log n)^3/\log n)$ when the size of the maximum clique is between $n/\log n$ and $n/(\log n)^3$. Since $V(G_3) = O(n^2)$, where $n$ is the number of vertices of the polygon, the performance ratio in our case is $O((\log \log n)^2/\log n)$.

Next, we consider the case in which more than one guard can be assigned to each non-safe triangle. In this case, we partition $T_j$ into disjoint regions, each covered by a unique guard (incident on $T_j$). The problem of computing the minimum guard speed ($\bar{v}_3$) when multiple guards can be assigned to a single triangle is as hard as UNIALLOC which itself is NP-hard.

In the next section, we address the problem of allocating guards to triangles for a given maximum speed of the intruder and the guards.

5 Computing a Feasible Allocation for Known Guard Speed

In the previous section, we proved that the problem of finding the minimum speed of the guards that can ensure tracking is NP-hard. In this section, we propose a technique to search and compute a feasible allocation for given maximum speed of the intruder and guards. Therefore, the speed ratio $r$ is known. We address the general allocation problem in which multiple guards can be assigned to cover an unsafe triangle in the polygon. Specifically, we focus on techniques that partition the polygon, and activate guards to track an intruder located in its allocated partition.

Let $R^g(i)$ denote the region inside triangle $T_j$ assigned to the guard $g_i \in G(T_j)$ incident to vertex $v_a(i)$ ($\alpha = 1, 2$). We define $S^g_\alpha(i) = \{R^g(i) : R^g(i) \subseteq T_j \in \mathbb{G}^{\mathbb{G}}(g_i)\}$ as the set of regions assigned to $g_i$ that can be covered from $v_a(i)$. We define $\hat{U}^g_\alpha(i) = \bigcup_{R^g(i) \subseteq S^g_\alpha(i)} R^g(i)$ as the union of the regions belonging to $S^g_\alpha(i)$. Let $\mathbb{R}^{\text{alloc}}(g_i) = \hat{U}^1_R(i) \cup \hat{U}^2_R(i)$ denote the region assigned to $g_i$. The cost associated with an assignment is defined as $c(\mathbb{R}^{\text{alloc}}(g_i)) = l_i/d(U^1_R(i), \hat{U}^2_R(i))$.

In this section, we address the following problem:

**Problem 2:** For the $[n/4]$ deployment of guards inside a polygon described in Section 5.1 and a given $r$, determine $\mathbb{R}^{\text{alloc}}(g_i)$ for each $g_i \in \mathbb{G}$ such that $\max\{c(\mathbb{R}^{\text{alloc}}(g_i)) : g_i \in \mathbb{G}\} \leq r$, and the region in which the intruder lies can be covered by the guard allocated to it.

Since $c(\mathbb{R}^{\text{alloc}}(g_i)) = l_i/d(U^1_R(i), \hat{U}^2_R(i)) \leq r, d(U^1_R(i), \hat{U}^2_R(i)) \geq l_i/r$ for all $g_i \in \mathbb{G}$. Let $d^2_i = l_i/r$ denote the maximum distance traveled by the intruder during the time in which $g_i$ can travel from one endpoint of $h_i$ to the other.

In the next section, we present the procedure to find $\mathbb{R}^{\text{alloc}}(g_i)$.

5.1 Sequential computation of partitions

The allocation algorithm proposed in the next section sequentially computes $\hat{U}^g_\alpha(i)$ for each guard $g_i$. The end
point of \( h_i \) at which \( \hat{U}_R(i) \) is computed first gets the label \( \alpha = 1 \). Subsequently, \( \hat{U}_R(i) \) is computed at the other end point of \( h_i \) which is assigned the label \( \alpha = 2 \).

From the definition of \( \hat{U}_R(i) = \bigcup_{I^j(i) \in S^j(i)} R^j(i) \), it is clear that \( R^j(i) \) must be defined for every \( T_j \in T^{safe}_{g(c)}(g_i) \) for computing \( \hat{U}_R(i) \). Based on the classification of \( T_j \in T^{safe}_{g(c)}(g_i) \), we can have the following scenarios:

1. If every \( T_j \in T^{safe}_{g(c)}(g_i) \) is an unsafe triangle, \( R^j(i) = T_j \) by definition since \( T_j \) can only be covered by \( g_i \).
   
   In this case, \( \hat{U}_R(i) = \bigcup_{T_j \in T^{safe}_{g(c)}(g_i)} T_j \).

2. If \( T_j \) is a regular triangle, \( R^j(i) \) can be computed only if the region allocated to other guards incident to \( T_j \) is known. In this case, \( R^j(i) = \bigcap_{T_i \in \Gamma(T_j) \cap T_j} \hat{U}_R(i) \)
   
   where \( \hat{U}_R(i) \) is the complement of \( R^j(i) \).

The following equation allows computing \( \hat{U}_R^j(i) \) from \( \hat{U}_R(i) \):

\[
R^j(i) = \left\{ \bigcap_{p \in \hat{U}_R(i)} \beta^{-1}_{d_j} (p) \cap T_j^{free} \right\}, \quad \hat{U}_R^j(i) \neq \emptyset
\]

where \( \beta^{-1}_{d_j} (p) \) is an open ball (using the metric defined in Section 3) of radius \( d_j \) centered at \( p \), and \( \beta^{-1}_{d_j} (p) \) is its complement. Also, \( T_j^{free} \subseteq T_j \) is the region inside \( T_j \) that has not yet been assigned to a guard. In the absence of obstacles between \( R^j(i) \) and \( \hat{U}_R(i) \):

\[
R^j(i) = (\hat{U}_R(i) \bigoplus B_{d_j})^c \cap T_j^{free},
\]

where \( B_{d_j} \) is an open ball in \( \mathbb{R}^2 \), \( \bigoplus \) denotes the Minkowski sum, and \( (\hat{U}_R(i) \bigoplus B_{d_j})^c \) is the complement of the Minkowski sum. The set \( (\hat{U}_R(i) \bigoplus B_{d_j}) \) is called the offset of \( \hat{U}_R(i) \). The offset of a set bounded by a polyline curve (line-segments/arcs of circle) is a set bounded by polyline curve \[\text{[55]}.\] Appendix \[\text{[A]}\] shows that \( R^j(i) \) and \( \hat{U}_R(i) \) are sets bounded by polyline curves even in the presence of obstacles. From \( \hat{U}_R(i) \) and \( \hat{U}_R(i) \), we can compute \( \mathbb{R}^{alloc}(g_i) \) using the equation \( \mathbb{R}^{alloc}(g_i) = \hat{U}_R(i) \cup \hat{U}_R^j(i) \).

In Figure 2 there are three guards, \( g_1 \), \( g_2 \), and \( g_3 \). All the non-safe triangles incident to one endpoint of the diagonals of \( g_1 \) and \( g_3 \) are unsafe triangles. Therefore, \( \hat{U}_R(1) \) and \( \hat{U}_R(3) \) are as shown in the figure. However, that is not the case for \( g_2 \). In order to define \( \hat{U}_R(2) \), either \( g_1 \) or \( g_3 \) needs to have a region assigned to it in the triangles shared between it and \( g_2 \). In Figure 2 \( \hat{U}_R^j(1) \) (which is obtained from [4]) is used to define \( \hat{U}_R(2) \), which in turn is used to define \( \hat{U}_R^j(2) \) using [4].

Once that all the guards in \( G(T_j) \) have been assigned to a region \( R^j(i) \), it is possible to determine if no allocation was found for the given \( r \). By definition, a region \( R^j(i) \) inside \( T_j \) ensures that every location within the triangle is assigned to a guard in \( G(T_j) \). However, that may not hold if all the regions allocated to the guards in \( T_j \) are \( R^j(i) \). Every \( R^j(i) \) is constructed using [4], thereby, guaranteeing that \( g_i \) is assigned to the largest possible region within \( T_j \) such that \( c(\mathbb{R}^{alloc}(g_i)) \geq r \).

Otherwise, there is no guarantee to keep track of the intruder if it follows the shortest path between \( \hat{U}_R(i) \) and \( \hat{U}_R(i) \). Clearly, an allocation cannot be found by the above process if there is a region inside the triangle that is not assigned to any guard in \( G(T_j) \) after every \( R^j(i) \) is defined for \( T_j \). In Figure 2 there is a region \( R_0 \) inside the regular triangles shared by \( g_2 \) and \( g_3 \) that cannot be assigned to those guards. The existence of a region \( R_0 \neq \emptyset \) implies that an allocation that guarantees successful tracking was not found. The blue shaded triangles correspond to safe triangles, which by definition are covered all the time.

5.2 Guard Allocation

In this section, we present an allocation algorithm that respects the constraints in Problem 2. Algorithm 2 presents the pseudocode of the allocation technique. Given a triangulation graph \( G \) and the speed ratio \( r \) as the input, Algorithm 2 provides a partition of the polygon \( P \), and an allocation of the guards to the partitions. The initialization, update and termination step of Algorithm 2 are as follows:

- **Initialization:**
  - Initialize two queues: \( G_{ready} = G_{alloc} = \emptyset \).
If there is a $g_i \in G$ such that $h_i$ has an endpoint $v_i(i)$ at which all the non-safe triangles incident to it are unsafe triangles, then $g_i$ is added to $\mathbb{G}_{\text{ready}}$.

If no guard satisfies the previous criteria, $\mathbb{G}_{\text{ready}} = \emptyset$. A guard $g_i$ is selected using Algorithm 3, and all the triangles incident to one of its endpoints are assigned to it.

- **Update:**
  - If there is a guard $g_j \notin \mathbb{G}_{\text{ready}} \cup \mathbb{G}_{\text{alloc}}$ such that for each $T_j \in \mathbb{T}_{\text{safe}}^e(g_j)$, the regions $\mathbb{R}^i_j(k)$ are already defined and allocated for all $g_k \in \mathbb{G}(T_j) \setminus \{g_j\}$. $v_i(i)$ is called the preferential endpoint of $g_i$, and is labeled as $v_1(i)$.
  
  - A guard $g_i \in \mathbb{G}_{\text{ready}}$ is selected. Algorithm 3 is used to compute $U_R^i(i)$ and $U_R^j(i)$.
  
  - If $\mathbb{G}_{\text{ready}} = \emptyset$, a guard is arbitrarily selected using Algorithm 3. Thus, Algorithm 2 can continue as in the previous step.

- **Termination:** The algorithm can terminate in two ways described as follows.

  - Algorithm 3 finds a region $R_2(j) \neq \emptyset$ inside a non-safe triangle $T_j$. Since the region $R_2(j)$ cannot be assigned to any guard, the algorithm terminates. Therefore, no allocation is found.
  
  - Each $T_j \in \mathbb{T}_{\text{safe}}^e(G)$ is partitioned, and each partition is assigned to a guard. In this case a feasible allocation is found.

Figure 3(a) shows a polygonal environment with 21 (= n) edges and 4 (\leq \lceil n/4 \rceil) guards. There are 8 safe and 11 non-safe triangles. The corresponding graph $G^\#$ is shown in Figure 3(c). The orientation of the edges of $G^\#$ illustrates a partial order in which the vertices of $G^\#$ are allocated by the algorithm. $\mathcal{V}(G^\#)$ has 11 vertices, each one associated with an non-safe triangle. Edges exist between vertices that correspond to triangles incident to opposite endpoints of the diagonal of a guard. For example, $T_5$ and $T_11$ are incident to one endpoint of $h_4$ and $T_{10}$ is incident to the other. Hence, edges in $G^\#$ that correspond to $g_4$ are $e_{10,5}(g_4)$ and $e_{10,11}(g_4)$. In Algorithm 2 once a $g_i \in \mathbb{G}_{\text{ready}}$ is chosen (and $\mathbb{R}_{\text{alloc}}(g_i)$ is obtained by Algorithm 3), the endpoints of its diagonal can be labeled as $v_1(i)$ and $v_2(i)$.

In the example shown in Figure 3, when the algorithm starts, $\mathbb{G}_{\text{ready}} = \{g_1, g_4\}$. $g_1$ meets the definition of the $\mathbb{G}_{\text{ready}}$ queue since all the non-safe triangles incident to one endpoint of $h_1$ are unsafe triangles ($T_1$ and $T_2$). The same is true for one endpoint of $h_4$ ($T_{10}$ is an unsafe triangle). Next, $g_1$ is selected and its corresponding preferential endpoint is named $v_1(1)$. Algorithm 3 is called. It uses the procedure to compute $\mathbb{R}_{\text{alloc}}(g_i)$ from Section 5.1 for a given $g_i$. Thus, $R_1^1(1) = T_1$ and $R_2^1(1) = T_2$. $S_{\text{safe}}^e(1) = \{T_1, T_2\}$ and $U_R^1(1) = T_1 \cup T_2$. Edges $e_{1,3}(1)$, $e_{1,4}(1)$, $e_{1,5}(1)$ and $e_{1,11}(1)$ become outgoing edges of $v_1$, and edges $e_{2,3}(1)$, $e_{2,4}(1)$, $e_{2,5}(1)$ and $e_{2,8}(1)$ become outgoing edges of $v_2$. Next, the regions that correspond to the endpoint $v_2(1)$ are computed. They are $R_2^1(1) = T_3$, $R_2^2(1) = T_5$, $R_2^3(1) = T_8$ and $R_2^4(1)$ which is the region inside $T_4$ assigned to $g_i$ shown in Figure 3(b). Finally, the algorithm searches for the existence of any region $R_2^j(1) \neq \emptyset$ with $j \in \{3, 4, 5, 8\}$. Since it is not found, the process returns to Algorithm 2 where $\mathbb{G}_{\text{alloc}}$ (which contains all guards that have been allocated) and $\mathbb{G}_{\text{ready}}$ are updated to $\mathbb{G}_{\text{alloc}} = \{g_1\}$ and $\mathbb{G}_{\text{ready}} = \{g_4, g_2\}$. In the second iteration, $g_4$ is selected, and the same process is repeated. $R_1^2(1) = \hat{U}_R^2(1) = T_{10}$. Edges $e_{10,9}(4)$ and $e_{10,11}(14)$ become outgoing edges of edges of $v_1$. $R_1^1(4) = T_{11}$ and $R_2^4(1)$ (the region inside $T_5$ assigned to $g_i$ shown in Figure 3(b)) are obtained. At the end of the iteration, $\mathbb{G}_{\text{alloc}} = \{g_1, g_4\}$ and $\mathbb{G}_{\text{ready}} = \{g_2, g_3\}$. The algorithm continues until the end of the fourth iteration at which point $\mathbb{G}_{\text{alloc}} = \{g_1, g_4, g_2, g_3\}$ and $\mathbb{G}_{\text{ready}} = \mathcal{G}(\mathbb{G}_{\text{ready}} \cup \mathbb{G}_{\text{alloc}}) = \emptyset$. The allocation of all the regions of the environment is shown in Figure 3(b), and the resulting directed graph $G^\#$ is illustrated in Figure 3(c). Notice that Line 15 of Algorithm 3 is only reached when all the edges incident to a vertex $v_j$ in $G^\#$ are incoming edges. Thus, the regions $\hat{U}_R^1(g_i)$ and $\hat{U}_R^2(g_i)$ for each guard $g_i \in \mathbb{G}(T_j)$ are already defined. Therefore, the algorithm checks whether the triangle is completely covered by the regions allocated to the guards or there is a region inside the triangle that cannot be assigned to guards. In the example, the aforementioned check is performed for triangles $T_8$, $T_7$, $T_3$, $T_{11}$, $T_6$ and $T_5$. We can see that for all the other vertices, there are outgoing edges incident to them, and those edges are associated with a single guard. According to the definition of $\mathbb{G}_{\text{ready}}$, at each iteration Algorithm 2 selects a guard $g_i$ such that the regions of the other guards that can cover the triangles in $\mathbb{T}_{\text{safe}}^e(g_i)$ are already defined. It follows that Algorithm 3 selects the vertices that correspond to the triangles in $\mathbb{T}_{\text{safe}}^e(g_i)$ and orients all the edges incident to them that correspond to $g_i$ as outgoing edges of those vertices. Since all the other guards that can cover those triangles had their regions defined, it means that Algorithm 4 was called before to orient their edges so all of those edges are outgoing edges of other vertices. Hence, they are incoming edges of the vertices associated with the triangles in $\mathbb{T}_{\text{safe}}^e(g_i)$. This explains why either all the edges incident to a vertex in $G^\#$ become incoming edges, or only the edges associated to one guard become outgoing edges.
Algorithm 3 LOCALLOC

1: Input: \( g_i, \bigcup_{k=1}^m \mathcal{F}^k(g_i), G^\# \) and \( G \).
2: Output: Regions \( U^1_R(i) \) and \( U^2_R(i) \) are defined and assigned to \( g_i \) or no feasible allocation found.
3: \( E(i) \leftarrow \emptyset \)
4: for each \( T_j \in \bigcup_{k=1}^m \mathcal{F}^k(g_i) \) do
5: \( R^1_j(i) \leftarrow \bigcap_{g_i \in \mathcal{T}(T_j) \setminus \{g_i\}} \mathcal{F}_j^k(k) \cap T_j \)
6: \( E'(i) \leftarrow \) edges incident to \( v_j \) that correspond to \( g_i \)
7: orient edges \( e_j(i) \in E'(i) \) as outgoing edges of \( v_j \)
8: add edges in \( E'(i) \) to \( E(i) \)
9: end for
10: compute \( S^1_R(i) \) and \( U^1_R(i) \)
11: for each \( e \in E(i) \) do
12: \( v_j \leftarrow \) head of \( e \)
13: compute \( R^2_j(i) \) using (1)
14: if for all \( g_k \in G(T_j) \), \( R^2_j(i) \) is defined then
15: \( R^2_k(j) \leftarrow \bigcap_{g_k \in \mathcal{T}(T_j)} \mathcal{F}_j^k(k) \cap T_j \)
16: if \( R^2_k(j) \neq \emptyset \) then
17: return True
18: end if
19: end if
20: end for
21: compute \( S^2_R(i) \) and \( U^2_R(i) \)
22: return False

If \( G_{\text{ready}} = \emptyset \) and \( G_{\Omega} \neq \emptyset \) (Line 18), Algorithm 2 calls Algorithm 3 to allocate the guards in \( G_{\Omega} \). It happens when the iterative procedure cannot find a unique allocation for the non-safe triangles that can be covered by the guards in the set \( G_{\Omega} \). Let \( T_{\Omega} \) denote the triangles that can be covered by the guards in \( G_{\Omega} \) such that they contain regions that have not yet been assigned. Let \( V_{\Omega} \) be the set of vertices in \( V(G^\#) \) that correspond to the triangles in \( T_{\Omega} \). Triangles in \( T_{\Omega} \) contain regions that can be assigned to more than one guard, and hence, it is not possible to determine a unique partition. Thus, for each \( v_j \in V_{\Omega} \) there are at least two guards with non-oriented edges incident to \( v_j \).

Figure 4 shows such a scenario involving 2 guards and 6 non-safe triangles. If Algorithm 2 was executed, it could find \( R^1_j(2) = T_4 \) or \( R^1_j(2) = T_6 \), or also \( R^1_j(1) = T_4 \) or \( R^1_j(1) = T_3 \). The presence of the regular triangles \( T_2 \) and \( T_4 \) prevents Algorithm 3 to find \( U^1_R(i) \) and \( U^2_R(i) \). There is no region that has initially been assigned to any of the guards inside \( T_2 \) and \( T_4 \) which can be used to construct the region of the other guard. Therefore, \( G_{\text{ready}} = \emptyset \) and \( V_{\Omega} = \{v_2, v_4\} \). Let \( G_{I} \) be the subgraph induced by \( V_{\Omega} \). We know that there are at least two guards with non-oriented edges incident to each \( v_j \in V_{\Omega} \). Since the number of vertices of \( G_1 \) is finite (\( G_{\Omega} \) cannot be a tree with an infinite length path), each \( v_j \in V_{\Omega} \) is inside a cycle involving the diagonal of guards in \( G_2 \). Otherwise, if there is a \( v_j \in V_{\Omega} \) with only one neighbor in \( V_{\Omega} \), there is only one guard with a non-oriented edge incident to it which implies \( v_j \notin V_{\Omega} \). In
Since the number of edges in $G^\#$ that can be associated to a given guard is upper bounded by $n(n - 1)/2$, the time complexity of Algorithm 3 is $O(n^2)$. Also, since the total number of edges that can be deleted is upper bounded by $O(n^2)$, the time complexity of Algorithm 4 is $O(n^2)$. At each iteration of Algorithm 2, one guard enters $\mathcal{G}_{alloc}$ and Algorithm 2 is called. It follows that the time complexity of Algorithm 2 is $O(n^3)$.

### Algorithm 4 ARBITALLOC

1: procedure 1 of General Allocation($\mathcal{G}_{\Omega}$, $G^\#$)
2: choose $g_i \in \mathcal{G}_{\Omega}$
3: $v_i(1) \leftarrow$ arbitrary endpoint of $h_i$
4: for each $T_k \in \text{T}_i^{\text{alloc}}(g_i)$ do
5:   $\mathcal{G}_{\text{del}}(T_k) \leftarrow (\mathcal{G}_{\text{del}}(T_k) \cap \mathcal{G}_{\Omega}) \setminus \{g_i\}$
6: for each $g_j \in \mathcal{G}_{\text{del}}(T_k)$ do
7:   delete from $G^\#$ all edges incident to $v_k$ that correspond to $g_j$
8: end for
9: end for
10: end procedure

5.3 Completeness and Correctness of GENALLOC

The following lemma proves that Algorithm 2 terminates.

**Lemma 2** Algorithm 2 terminates in a finite number of steps.

**Proof** Each iteration of Algorithm 2 computes the region to be allocated to an unassigned guard. Since the number of guards is finite, Algorithm 2 terminates in finite number of steps.

The next lemma states a condition under which Algorithm 2 is complete.

**Lemma 3** If $G^\#$ is a forest, Algorithm 2 is complete.

**Proof** The proof is by contradiction. Consider a polygonal environment $P$ for which $G^\#$ is a forest, and Algorithm 2 fails to find a feasible allocation of guards even though one exists. From hereon, any variable with a symbol $\bar{\cdot}$ on top of it is associated with the feasible allocation. For example, $\bar{R}_j^2(i)$ denotes a region in $T_j$ assigned to $g_i \in T_j$ based on the feasible allocation.

Since Algorithm 2 cannot find a feasible allocation in $P$, it terminates at Line 15. Hence, there is a vertex $v_j \in \mathcal{V}(G^\#$) such that all the edges incident to $v_j$ are incoming edges, and the triangle $T_j \in \overline{T_{\text{alloc}}}$ has a region $R_0(j) \neq \emptyset$ which by definition cannot be assigned to any guard by Algorithm 3. Since a feasible allocation exists, every point inside $R_0(j)$ can be assigned...
to a guard in $G(T_j)$. Let $R^*_i(j) \subseteq R^i(j)$ denote the region that can be covered by a guard $g_i \in G(T_j)$ in the feasible allocation. Therefore, $R^*_i(j)$ exists such that $R^*_i(j) \subseteq \bar{R}^i(i)$ and $R^*_i(j) \not\subseteq R^2_i(i)$. For the feasible allocation, $d(U^2_R(i), \bar{R}^2_i(i)) \geq d^*_i$. Since $R^*_i(j) \cap \bar{R}^2_i(i) \neq \emptyset$, $d(U^1_R(i), \bar{R}^2_i(i)) < d^*_i$. By definition, $d(U^1_R(i), \bar{R}^2_i(i)) \geq d^*_i$. Therefore, a region $R^*_i(k) \subseteq T_k \in \mathbb{T}_{1}^{a}(g_i)$ exists such that $R^*_i(k) \subseteq R^1_i(i)$ and $R^*_i(k) \not\subseteq R^1_i(i)$. Therefore, $R^*_i(k) \subseteq \bigcup_{k \in \mathbb{G}(T_k) \setminus \{g_i\}} \bar{R}^2_k(a)$. Hence, there exists a guard $g_a \in G(T_k) \setminus \{g_i\}$ and a region $R^*_i(k) \subseteq R^i(k)$ such that $R^*_i(k) \subseteq \bar{R}^k_i(a)$ and $R^*_i(k) \not\subseteq R^2_i(a)$. Therefore, we can find a sequence $v_1 \rightarrow v_2 \rightarrow \cdots$ of vertices and guards in $G^\#$ such that $g_{j_k} \in G(T_{i_k}) \setminus \{g_{j_{k-1}}\}$, and corresponding regions $R^k_{i_k}(j_k) \subseteq R^i_k(i_k)$ such that $R^k_{i_k}(j_k) \not\subseteq R^2_k(i_k)$ and $R^k_{i_k}(j_k) \subseteq R^1_k(j_k)$ for each $g_{j_k}$. The sequence terminates if $\{g_{j_{k-1}}\} = G(T_{i_{k-1}})$ in which case $T_{i_k}$ is an unsafe triangle. Since $j_k$ does not exist $\bar{R}^k_{i_k}(j_k) = \emptyset \Rightarrow R^k_{i_k}(i_k) = \emptyset$. Since $R^k_{i_k}(i_k) = R^k_{i_k}(i_k)$ and $R^k_{i_k}(i_k) \not\subseteq \bar{R}^k_i(a)$, we arrive at a contradiction. If the sequence does not terminate, then there exist $i_k$ and $i_j$ such that $i_k = i_j$ since the number of vertices in $G^\#$ are finite. This implies the existence of a cycle in $G^\#$. Therefore, we arrive at a contradiction since $G^\#$ is a forest.

In the appendix (Lemma 8), we prove a more general result which shows that Algorithm 2 is complete if Algorithm 1 is never called during execution. Figure 3 shows an example in which $G^\#$ does not contain cycles. In Figure 3(a), a simple polygonal environment is shown along with the deployment of guards. There are four non-safe triangles $T_1$, $T_2$, $T_3$ and $T_4$. $T_1$, $T_2$ and $T_3$ can be covered by $g_1$, $g_2$ and $g_3$ respectively. Algorithm 2 selects the non-safe triangle $T_2$ to be assigned to the only guard that can cover it ($g_2$). Since $|G(T_2)| = 1$, $U^1_R(2) \Rightarrow g_2$ is a type 1 guard. Once $U^1_R(1) \in T_1$ is computed after a few steps, Algorithm 2 selects the non-safe triangle $T_3$ and allocates the unshaded region to $g_3$. Besides $g_3$, $T_3$ can also be covered by $g_1$ and $g_2$. Regions $U^1_R(1)$ and $U^1_R(2)$ are known. Consequently, $R^2_1(1)$ and $R^2_1(2)$ are also computed. It follows that the unshaded region in $T_3$ is labeled as $R^1_3(3)$. Moreover, since $\mathbb{T}_{1}^{a}(g_1) = \{T_3\}$, $U^1_3(3) = R^1_3(3)$. It follows that $g_3$ is an example of a type 2 guard.

5.4 Classification of Guards

In this section, we present a classification of the guards. It is based on the regions $U^\alpha_R(i) = \{1, 2\}$ constructed from Algorithm 2.

1. Type 0 guard: These are guards for which either $U^1_R(i) = \emptyset$ or $U^2_R(i) = \emptyset$. Since the region allocated to a type 0 guard can be covered from one endpoint of its diagonal, it is a static guard.

2. Type 1 guard: There are guards for which all the non-safe triangles allocated to the guard incident to one endpoint are unsafe triangles. Notice that each edge in $\mathbb{E}(G^\#)$ that corresponds to a type 1 guard is an outgoing edge of a vertex $v_j$ such that $|\mathbb{G}(T_j)|= 1$. Clearly, if $\mathbb{G}(T_j)\setminus \{g_i\} = \emptyset$ for each $T_j \in \mathbb{T}_{1}^{a}(g_i)$, then $g_i$ is a type 1 guard.

3. Type 2 guard: Any guard which is neither Type 0 nor Type 1 is a Type 2 guard.

Consider the example shown in Figure 6. Assume $T_1$, $T_2$ and $T_3$ are the only non-safe triangles incident to $v_1(1)$, $v_1(2)$ and $v_1(3)$ respectively. Algorithm 2 selects the non-safe triangle $T_2$ to be assigned to the only guard that can cover it ($g_2$). Since $|\mathbb{G}(T_2)| = 1$, $U^1_R(2) = T_2 \Rightarrow g_2$ is a type 1 guard. Once $U^1_R(1) \in T_1$ is computed after a few steps, Algorithm 2 selects the non-safe triangle $T_3$ and allocates the unshaded region to $g_3$. Besides $g_3$, $T_3$ can also be covered by $g_1$ and $g_2$. Regions $U^1_R(1)$ and $U^1_R(2)$ are known. Consequently, $R^2_1(1)$ and $R^2_1(2)$ are also computed. It follows that the unshaded region in $T_3$ is labeled as $R^1_3(3)$. Moreover, since $\mathbb{T}_{1}^{a}(g_1) = \{T_3\}$, $U^1_3(3) = R^1_3(3)$. It follows that $g_3$ is an example of a type 2 guard.
triangles incident to one of the endpoints of its diagonal. \( g_i \) is arbitrarily chosen since Algorithm 2 failed to find a unique partition of such regions. It implies that \( \hat{U}_R^1(i) \) cannot be constructed unlike in the case of a type 0, a type 1 nor a type 2 guard. However, the arbitrary allocation in Algorithm 4 assigns those regions to \( g_i \). As a result, \( g_i \) is converted to a type 1 or a type 2 guard.

5.5 Motion strategy for the Guards

In this section, we present a motion strategy for the guards to move on their diagonals. We introduce the concept of critical curves to propose activation strategies for type 1 and type 2 guards.

From the discussion in the previous section, \( \hat{U}_R^1(i) \) is the region assigned to a type 1 or type 2 guard \( g_i \). Therefore, it is the responsibility of \( g_i \) to cover the triangles incident to \( v_i(i) \) when the intruder lies in \( \hat{U}_R^1(i) \).

We define an internal critical curve, denoted by \( s_{\text{int}}^i \), as the boundary of \( \hat{U}_R^1(i) \). Corresponding to an internal critical curve, we define an external critical curve as follows:

\[
s_{\text{ext}}^1(i) = \{ p \in P \setminus \hat{U}_R^1(i) : d(p, s_{\text{int}}^i) = d_T^i \} \tag{6}
\]

Comparing the definition of \( s_{\text{ext}}^1(i) \) to the definition of \( R_T^1(i) \) (see Equation (4)), we can conclude that a part of the boundary of \( R_T^1 \) can belong to \( s_{\text{ext}}^1(i) \). We define a critical region associated with the guard \( g_i \) as follows:

\[
C_1(i) = \{ p \in P \setminus \hat{U}_R^1(i) : d(p, s_{\text{int}}^1(i)) \leq d_T^i \}, \tag{7}
\]

where \( \hat{U}_R^1(i) \) is the interior of \( \hat{U}_R^1(i) \). Note that, by definition, the boundary of \( C_1(i) \) contains both curves \( s_{\text{int}}^1(i) \) and \( s_{\text{ext}}^1(i) \), and since \( d(\hat{U}_R^1(i), \hat{U}_R^2(i)) \geq d_T^i \), it is clear that \( (\hat{U}_R^2(i) \cap C_1(i)) \subset s_{\text{ext}}^1(i) \).

Figure 7 shows the region \( \hat{U}_R^1(i) = R_T^1(i) \cup R_T^2(i) \cup R_T^3(i) \) for the guard \( g_i \). The neighboring guards are not shown for the sake of simplicity. \( s_{\text{int}}^1(i) \), the boundary of \( \hat{U}_R^1(i) \), is represented as blue segments and arcs that form the boundary of the regions \( R_T^1(i) \), \( R_T^2(i) \), and \( R_T^3(i) \). Since triangles \( T_4 \) and \( T_5 \) are safe triangles (they have \( h_i \) as an edge), there is no internal critical curve inside them. The green segments and curves denote \( s_{\text{ext}}^1(i) \), and the unshaded region inside \( P \) is \( C_1(i) \). The boundary of \( C_1(i) \) is formed by \( s_{\text{int}}^1(i) \), \( s_{\text{ext}}^1(i) \) and edges of the environment. The dark colored regions represent \( R_T^2(i) \) and \( R_T^3(i) \) which are part of \( \hat{U}_R^1(i) \).

Fig. 7: Example of the definition of the critical region of guard.

For an intruder located in \( C_1(i) \), the following equation maps the position of the intruder \((x_i)\) to the position of \( g_i \) denoted as \( x_{g_i} \) along its diagonal:

\[
x_{g_i} = x_{v_1(i)} + \frac{d(s_{\text{int}}^1(i), x_i)}{d_T^i}(x_{v_2(i)} - x_{v_1(i)}) \tag{8}
\]

where \( x_{v_1(i)} \) is the location of vertex \( v_1(i) \) and \( x_{v_2(i)} \) is an arbitrary point on \( \hat{U}_R^1(i) \). If \( x_i \in \hat{U}_R^1(i) \), \( g_i \) remains static at \( v_1(i) \). Otherwise, if \( x_i \not\in \hat{U}_R^1(i) \), \( g_i \) remains static at \( v_2(i) \).

By definition, \( d(U_R^1(i), U_R^2(i)) \geq d_T^i \) while \( d(U_R^1(i), s_{\text{ext}}^1(i)) = d_T^i \). Hence, \( [3] \) guarantees that \( g_i \) will always cover the regions assigned to it when the intruder is located in them.

Consider the case of a guard \( g_i \) such that \( g_i \) is incident to \( T_k \) at a vertex \( v_2(i) \). The motion strategy proposed in \([3] \) ensures that \( \hat{U}_R^1(i) \) is covered if there is an intruder inside \( \hat{U}_R^1(i) \) (Fig. 6). However, if the intruder is located in \( \hat{U}_R^1(i) \cup C_1(i) \), where \( C_1(i) \) is the interior of \( C_1(i) \), \( g_i \) cannot cover \( T_k \) because \( g_i \) can only be located at \( v_2(i) \) when the intruder is outside \( \hat{U}_R^1(i) \cup C_1(i) \) according to \([3] \). Now, consider the case when \( g_i \) is incident to \( T_k \) at a vertex labeled \( v_3(i) \). Since \( v_3(i) \) is a vertex of \( T_k \), \( T_k \) will be covered by \( g_i \) if there is an intruder inside it according to \([3] \). However, if the intruder lies outside \( T_k \), then \( T_k \) is not covered by \( g_i \) since \( g_i \) is not located at \( v_3(i) \) according to \([3] \). Consequently, for each \( g_i \in G(T_k) \), there exists a region which prevents \( g_i \) to cover \( T_k \) when the intruder lies inside it. This region, denoted by \( C_1(i) \), is called the extended critical region. It is given by the following expression:

\[
\hat{C}_1(i) = \left\{ \begin{array}{ll}
\hat{U}_R^1(i) \cup \hat{U}_R^2(i) & \text{a vertex of } T_k \\
P \setminus \hat{U}_R^1(i) & \text{otherwise}
\end{array} \right. \tag{9}
\]

Based on the concept of extended critical regions, Lemma 4 presents a necessary and sufficient condition for the guards to cover a non-safe triangle when an intruder lies in it.
Lemma 4 For the guards in \( G(T_k) \) (\( T_k \in T^{saF}(G) \)), \( g \) guarantees that the triangle \( T_k \) is covered when an intruder is located in it if and only if \( \bigcap_{g_i \in G(T)} (\hat{C}_1(i) \cap T_k) = \emptyset \).

Proof (\( \Rightarrow \)) Assume that \( \bigcap_{g_i \in G(T)} (\hat{C}_1(i) \cap T_k) = \emptyset \), and \( T_k \) is not covered when the intruder lies in \( T_k \). It implies that there is a location inside \( T_k \) for the intruder that prevents every \( g_i \in G(T_k) \) to cover \( T_k \). According to \( g \), such a region must belong to \( \bigcap_{g_i \in G(T_k)} (\hat{C}_1(i) \cap T_k) \) which contradicts our assumption.

(\( \Leftarrow \)) Next, assume that \( \bigcap_{g_i \in G(T_k)} (\hat{C}_1(i) \cap T_k) \neq \emptyset \) and \( T_k \) is covered when the intruder is located in it. Since \( \bigcap_{g_i \in G(T_k)} (\hat{C}_1(i) \cap T_k) \neq \emptyset \) when \( x_I \in \bigcap_{g_i \in G(T_k)} (\hat{C}_1(i) \cap T_k) \), there is no guard covering \( T_k \) according to \( g \), which is a contradiction. The lemma follows.

6 Polygons with Holes

In this section, we assume that \( P \) has polygonal holes which represent obstacles inside the polygon. Let \( Q = \{Q_1, \ldots, Q_N\} \) represent the set of polygonal holes. Let \( n = n_1 + q_1 + \cdots + n_{Q_N} \) denote the total number of vertices of \( G \), where \( n_i \) is the number of vertices of the hole \( Q_i \) and \( i \in \{1, 2, \ldots, N\} \). Figure 8 shows a polygonal environment with an internal polygonal hole \( Q_1 \).

In Theorem 5.1 of [28], it is shown that one can find an internal diagonal of the triangulation of \( P \) between any two holes (or between a hole and the outer boundary) which merges two holes (or the hole with the boundary) if a wall of thickness 0 is placed on the diagonal. This reduces the value of \( N \) by 1. See Figure 8 where \( Q_1 \) is merged with the outer boundary through the diagonal shared by triangles \( T_1 \) and \( T_2 \). Therefore, for any polygon \( P \) with \( n \) vertices and \( N \) internal polygonal holes, we can construct a simply-connected polygon \( \hat{P} \) with \( n + 2N \) vertices. We can apply all techniques proposed in the previous sections for deploying guards, and allocating them to triangles of the triangulation of \( \hat{P} \) for tracking the intruder.

7 Tracking Multiple Intruders

In this section, we analyze the performance of the proposed algorithm for multiple intruders. We assume that all the intruders have the same maximum speed \( v_e \). We use the symbol \( I (|| > 1) \) to denote the set of intruders, and the vector \( x_I(t) \in \mathbb{R}^{||} \) to denote their positions inside the polygon. We assume that the deployment of the guards and the allocation of the different regions of the environment are obtained using the techniques presented in sections 3.1 and 4, respectively.

In Section 5.5, a motion strategy for the guards was proposed for a single intruder. \( g \) is a reactive motion strategy that depends on the location of the intruder. It ensures that each guard \( g_i \) can cover \( U^j_{R}(i) \) when the intruder is inside it. In the presence of multiple intruders, the priority for each \( g_i \) is to cover \( U^j_{R}(i) \) as long as there is an intruder inside it. Hence, the motion strategy only needs to consider the intruder closer to \( U^j_{R}(i) \), i.e. if there is an intruder inside \( U^j_{R}(i) \), \( g_i \) stays at \( v_i \) regardless of the positions of other intruders.

Therefore, the motion strategy of the guard in this case is given by (10), wherein \( d(s^1_{mot}(i), x_I) \) is replaced by \( d_{min}(s^1_{mot}(i), x_I) \) defined as follows:

\[
d_{min}(i, x_I(t)) = \min_{i \in I} d(s^1_{mot}(i), x_I(t)). \quad (10)
\]

In Section 5.5 we showed that an intruder inside each \( \hat{C}_1(i) \) associated to \( g_i \in G(T_k) \) will prevent an on-safe triangle \( T_k \) from being covered by any guard incident to it. Therefore, \( |G(T_k)| \) intruders are sufficient to keep \( T_k \) uncovered. However, if for example, there are two guards \( g_i, g_j \in G(T_k) \) such that \( \hat{C}_1(i) \cap \hat{C}_1(j) \neq \emptyset \), a single intruder lying inside the intersection will prevent \( g_i \) and \( g_j \) from covering \( T_k \). Therefore, fewer than \( |G(T_k)| \) intruders can prevent \( T_k \) from being covered by any guard incident to it if there are non-empty intersections between the extended critical regions corresponding to distinct guards that can cover \( T_k \).

Consider the power set \( 2^G(T_k) \) of all guards incident to \( T_k \). Let \( S \subseteq 2^G(T_k) \) be a collection of all sets \( S \in 2^G(T_k) \) for which the extended critical regions of the guards belonging to \( S \) have a non-empty intersection. The problem of finding the minimum number of intruders that can be placed at the intersection of extended critical regions to uncover \( T_k \) is equivalent to
the problem of finding the minimum cover $C \subseteq S$ of $G(T_k)$.

Let $n_I(T_k)$ denote the maximum number of intruders that can be tracked by the guards incident to $T_k$ without uncovering $T_k$ when there is an intruder inside it. The following lemma relates $n_I(T_k)$ to $|C|$.

**Lemma 5** Let $T_k \in T_{sa}^{I}(G)$.

$$n_I(T_k) = \begin{cases} |C| & T_k \cap I(C) = \emptyset, \forall C \in C \\ |C| - 1 & \text{otherwise,} \end{cases} \quad (11)$$

where $I(C) = \bigcup_{g_k \in C} \hat{C}_1(k)$

Proof $|C|$ intruders, each placed in a distinct $I(C)$, are sufficient to prevent all guards incident to $T_k$ from covering it when an intruder is located inside $T_k$. If $T_k \cap I(C) = \emptyset \forall C \in C$, the intruder inside the $T_k$ cannot lie inside any $I(C), C \in C$. Therefore, $n_I(T_k) = |C|$ in this case. Otherwise, the intruder located inside $T_k$ can cover an $I(C)$. Therefore, $n_I(T_k) < |C|$ in this case.

Since $C$ is the minimum set cover of $G$, $|C| - 1$ intruders cannot prevent $T_k$ from being covered by at least one guard $g_k \in G(T_k)$. Therefore, $n_I(T_k) \geq |C| - 1$. The theorem follows.

In Figure 9(a) two type 1 guards $g_1$ and $g_2$, and a type 2 guard $g_3$ are shown with their corresponding external and internal critical curves. The corresponding endpoints $v_1(i)$ are shown with a green disc. The regions $U_k^1(i)$ are shaded in orange, and the safe triangles are shaded in blue. $T$ is a regular triangle that can be covered by $g_1, g_2$ and $g_3$. Therefore, the set $C$ consists of the external critical regions and has cardinality 3. Since there is no intersection between any region in $C$ and $T$ then $n_I(T) = 3$. Figure 9(b) shows the same case but for a smaller value of $r$. $s_{\text{ext}}(1) \cap s_{\text{ext}}(2) \neq \emptyset$. This implies that $C$ consists of the extended critical region of $g_3$ and the intersection of the extended critical regions of $g_1$ and $g_2$. In this case, $C$ consists of $\hat{C}_1(4)$ and $\hat{C}_1(1) \cap \hat{C}_1(2)$, so it has cardinality 2. Since none of the regions in $C$ intersects with $T$, then $n_I(T) = 2$.

**Theorem 2** The minimum number of intruders that can be tracked based on the strategy proposed in [10] is $n_s^* = \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$.

Proof Assume that $|I| > \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$. It implies that there is at least one $T_k \in T_{sa}^{I}(G)$ for which $n_I(T_k) < |I|$ and therefore, $T_k$ cannot be covered at all times according to Lemma 5 $\implies n_s^* \neq \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$. Now assume that $|I| \leq \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$. According to Lemma 5, the guards have a strategy to cover every non-safe triangle if $|I| \leq \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$. According to Lemma 5, the guards have a strategy to cover every non-safe triangle if $|I| \leq \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$. Therefore, $n_s^* = \min\{n_I(T_k) : T_k \in T_{sa}^{I}(G)\}$.

Fig. 9: (a) An instance where the regions $\hat{C}_1(i)$ do not intersect. (b) An instance where two regions $\hat{C}_1(i)$ intersect.

The set cover problem is NP-complete [75]. Several polynomial time approximation schemes (PTAS) for the set cover problem have been proposed in the literature [80][77][88][89][90][91]. Better approximation ratios can be obtained at the expense of computational complexity slightly higher than a PTAS [92]. For example, it has been shown that any $(1 - \alpha \ln n)$-approximation algorithm for the set cover problem must run in time at least $2^{n^{\alpha \ln \alpha}}$ for some small constants $0 < c < 1$ [83][93][94]. Several present some efforts to tighten the running time by reducing the value of $c$ in $2^{n^{\alpha \ln \alpha}}$. We can use either of the aforementioned approaches to obtain $C$. For a problem instance of large size, one might prefer a PTAS, whereas a moderately exponential algorithm is more preferable when the number of guards covering each $T_k$ is small enough.

**8 Conclusion**

In this work, we addressed the problem of tracking mobile intruders in a polygonal environment using a team of diagonal guards. Leveraging on deployment strategies for mobile coverage in art gallery problems, we proposed control and coordination strategies for the guards to track intruders inside a polygonal environment. At first, we formulated the tracking problem as a multi-robot task assignment problem on the triangulation graph of a polygon. We classified the guards based on their position with respect to the triangles of the triangulation. Next, we showed that the problem of finding the minimum speed of the guards to cover the triangles of the triangulation under the constraint that each triangle can only be covered by a single guard is NP-hard. Given the maximum speed of the intruder, we proposed an algorithm to find a feasible allocation of guards to the triangles of the triangulation when multiple guards are allowed to cover the triangle. We proved the correct-
ness of the proposed algorithm, and its completeness for a specific set of inputs. Based on the task allocated to a guard, we proposed control laws for the guards to move along their diagonals. Finally, we extended the algorithm to address deployment and allocation strategies for non-simple polygons and multiple intruders.

We believe that our paper is a first step towards MRS deployment for persistent tracking with provable guarantees. An important direction of future research is to address the tracking problem for guards with sensing and motion constraints, for example, edge guards, which are more constrained in their motion, or line guards, which are less constrained than diagonal guards. Another future research direction is to study the tracking problem for special polygons, for example, orthogonal polygons, monotone polygons etc. For these polygons, it has been shown that fewer guards are required for coverage. Finally, the problem of tracking with mixed team of guards (static and mobile) is an interesting direction of future research.

References

1. S. Mittal and J. K. Rai. Wadoro: An autonomous mobile robot for surveillance. In Proceedings of IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems, pages 1–6, July 2016.

2. S. Witwicki, J. C. Castillo, J. Messias, J. Capitan, F. S. Melo, P. U. Lima, and M. Veloso. Autonomous surveillance robots: A decision-making framework for networked multiagent systems. IEEE Robotics Automation Magazine, 24(3):52–64, Sept 2017.

3. T. Theodoridis and H. Hu. Toward intelligent security robots: A survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(6):1219–1230, Nov 2012.

4. D. Cicciarizzo, H. Everett, G. Gilbreath, and T. Tran. Automated security response robot. 1999.

5. G. Tuna, C. Tasdemir, K. Gulez, T. V. Mumeu, and V. C. Gungor. Autonomous intruder detection system using wireless networked mobile robots. In Proceedings of IEEE Symposium on Computers and Communications, pages 1–5, July 2012.

6. Piotr Bazydło, Janusz Kacprzyk, and Krzysztof Lasota. Global path planning for a specialized autonomous robot for intrusion detection in wireless sensor networks (wsns) using a new evolutionary algorithm. In Wojciech Mitkowski, Janusz Kacprzyk, Krzysztof Oprzędkiewicz, and Paweł Skruch, editors, Trends in Advanced Intelligent Control, Optimization and Automation, pages 503–513. Springer International Publishing, 2017.

7. G. Oriolo, G. Ulivi, and M. Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 28(3):316–333, Jun 1998.

8. Michael Montemerlo and Sebastian Thrun. Large-scale robotic 3-d mapping of urban structures. In Marcelo H. Ang and Ouassama Khatib, editors, Experimental Robotics IX, pages 141–150, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
58. Zhijun Tang and Umit Ozguner. Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21(5):898–908, 2005.

59. C. Nam and D. A. Shell. Assignment algorithms for modeling resource contention and interference in multi-robot task-allocation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2158–2163, May 2014.

60. B. P. Gerkey and M. J. Mataric. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 3862–3868, Sept 2003.

61. Lynne E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation & Soft Computing*, 5(1):5–19, 1999.

62. S. C. Botelho and R. Alami. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1234–1239, 1999.

63. Brian P. Gerkey and Maja J. Mataric. Principled communication for dynamic multi-robot task allocation. In Daniela Rus and Sanjiv Singh, editors, *Experimental Robotics VII*, pages 353–362, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

64. Anthony Stentz and M Bernardine Dias. A free market architecture for coordinating multiple robots. Technical Report CMU-RI-TR-99-42, Carnegie Mellon University, Pittsburgh, PA, December 1999.

65. L. Chaimowicz, M. F. M. Campos, and V. Kumar. Dynamic role assignment for cooperative robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 293–298, 2002.

66. Kristina Lerman, Chris Jones, Aram Galstyan, and Maja Mataric. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

67. Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. Multi-robot decentralized exploration using a trade-based approach. In *Proceedings of International Conference on Informatics in Control*, volume 2, Jan 2011.

68. Torbjorn S. Dahl, Maja Mataria, and Gaurav S. Sukhatme. MR1-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6):674 – 687, 2009.

69. H. Hanna. Decentralized approach for multi-robot task allocation problem with uncertain task execution. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 555–560, Aug 2005.

70. N. Michael, M. M. Zaslavos, V. Kumar, and G. J. Pappas. Distributed multi-robot task assignment and formation control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 128–133, May 2008.

71. Stephane Durocher and Saeed Mehrabi. Guarding orthogonal art galleries using sliding cameras: Algorithmic and hardness results. In Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013*, pages 314–324, Berlin, Heidelberg, 2013. Springer Verlag.

72. Dietmar Schuchardt and Hans-Dietrich Hecke. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41(2):261–267, 1995.

73. Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In Kyung-Yong Chwa and Oscar H. Ibarra, editors, *Algorithms and Computation*, pages 427–437, Berlin, Heidelberg, 1998. Springer Berlin Verlag.

74. Mohammad AlMahmud et al. Algorithms for minimum length sliding camera problem. Master’s thesis, Bangladesh University of Engineering and Technology, 2016.

75. Richard M. Karp. Reducibility among combinatorial problems, pages 85–103. Springer US, Boston, MA, 1972.

76. Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216 – 231, 2005.

77. Michael R Garey and David S Johnson. *Computers and intractability*, volume 29, W. H. Freeman New York, 41 Madison Avenue, New York, NY, 2002.

78. Panos M Pardalos and Jue Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, 1994.

79. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

80. Johan Håstad. Clique is hard to approximate within $\sqrt{n}$. *Acta Mathematica*, 182(1):105–142, 1999.

81. Luitpold Babel. Finding maximum cliques in arbitrary and in special graphs. *SIAM Journal on Computing*, 46(4):321–341, 1991.

82. Egon Balas and Jue Xue. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, 15(5):397–412, 1996.

83. David R Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21(5):211–217, 1997.

84. Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225, 2004.

85. Xu-Zheng Liu, Jun-Hai Yong, Guo-Qin Zheng, and Jia-Guang Sun. An offset algorithm for polynome curves. *Computers in Industry*, 58(3):240 – 254, 2007.

86. V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, August 1979.

87. Rajiv Gandhi, Eran Halperin, Samir Khuller, Guy Kortsarz, and Aravind Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences*, 72(1):16 – 33, 2006.

88. Olivier Goldschmidt, Dorit S. Hochbaum, and Gang Yu. A modified greedy heuristic for the set covering problem with improved worst case bound. *Information Processing Letters*, 48(6):305 – 310, 1993.

89. David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256 – 278, 1974.

90. L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383 – 390, 1975.

91. Petr Slavák. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):237 – 254, 1997.

92. N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation of min set cover by moderately exponential algorithms. *Theoretical Computer Science*, 410(21):2184 – 2195, 2009.

93. Dana Moshkovitz. The projection games conjecture and the NP-hardness of ln n-approximating set-cover. In Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 93–104, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
Therefore, \( \gamma \) and Lemma 6 given \( s_2 \in S(\hat{\gamma}(\hat{U}_R^1(i))) \) the presence of an obstacle implies that for all the points in \( s_2 \), such as the one illustrated as a black circle, the shortest path between \( \gamma(\hat{U}_R^1(i)) \) and \( \delta(U_R^1(i)) \) consists of two connected line segments, one with endpoints in \( s_2 \) and vertex \( v_1 \in \mathcal{V}(G) \) and other with \( v_1 \) as an endpoint and the other at \( \delta(U_R^1(i)) \). Also, for the points in \( s_3 \in S(\gamma(U_R^1)) \), the shortest path between \( \gamma(U_R^1(i)) \) and \( \delta(U_R^1(i)) \) consists of three connected line segments, one from \( s_3 \) to \( v_2 \), another from \( v_2 \) to \( v_1 \) and the last one from \( v_1 \) to \( \delta(U_R^1(i)) \). Consider the points \( p \in \gamma(U_R^1(i)) \) for which the shortest path between them and \( \delta(U_R^1(i)) \) is not a line segment due to the presence of reflex vertices of the environment. As we can see in Figure 10 for any such \( p \) points, the shortest path from \( p \) to \( \delta(U_R^1(i)) \) must visit first a reflex vertex \( v_p \in \mathcal{V}(G) \) (\( v_1 \) or \( v_2 \) for instance) of the environment. Clearly, the union of such points \( p \) for which the first segment of the shortest path between them and \( \delta(U_R^1(i)) \) has \( v_p \) as an endpoint, is a subset of the union of all points in the plane that are equidistant to \( v_p \). Thus, they form an arc of circle, which is centered at \( v_p \) with radius \( d_p^1 - d(v_p, \delta(U_R^1(i))) \), where \( d(v_p, \delta(U_R^1(i))) \) is the length of the shortest path between \( v_p \) and \( \delta(U_R^1(i)) \). Therefore, every point in \( \gamma(U_R^1(i)) \) belongs to an arc of circle, so \( \gamma(U_R^1(i)) \) is the union of a set \( S(\gamma(U_R^1(i))) \) of arcs of circle.

**Lemma 6** Given \( S(\delta(U_R^1(i))) \), its corresponding \( \gamma(U_R^1(i)) \) consists of arcs of circle.

**Proof** Trivially, if there are no obstacles between \( \delta(U_R^1(i)) \) and \( \gamma(U_R^1(i)) \), \( \gamma(U_R^1(i)) = \{ p \in P(U_R^1(i)) : d(p, \delta(U_R^1(i))) = d_2 \} \) is the offset of \( \delta(U_R^1(i)) \) (with \( d_2 \) as the offset distance). Therefore, \( \gamma(U_R^1(i)) \) must be a polyline curve [52].

The presence of obstacles between \( \delta(U_R^1(i)) \) and \( \gamma(U_R^1(i)) \) implies that the shortest path from some points in \( \gamma(U_R^1(i)) \) to \( \delta(U_R^1(i)) \) is a chain of connected line segments instead of a line segment, as in the case where there are no obstacles between \( \gamma(U_R^1(i)) \) and \( \delta(U_R^1(i)) \). In Figure 10 a region \( U_R^1(i) \) is shown as an orange triangle, its corresponding \( \delta(U_R^1(i)) \) is a black dotted segment, \( \gamma(U_R^1(i)) \) is represented as a dotted curve divided into four arcs of circle. \( s_1 \in S(\gamma(U_R^1(i))) \) illustrates the case of points in \( \gamma(U_R^1(i)) \) such that the shortest path between them and \( \delta(U_R^1(i)) \) is a line segment. Now consider \( s_2 \in S(\gamma(U_R^1(i))) \) the presence of an obstacle implies that for all the points in \( s_2 \), such as the one illustrated as a black circle, the shortest path between \( \gamma(U_R^1(i)) \) and \( \delta(U_R^1(i)) \) consists of two connected line segments, one with endpoints in \( s_2 \) and vertex \( v_1 \in \mathcal{V}(G) \) and other with \( v_1 \) as an endpoint and the other at \( \delta(U_R^1(i)) \). Also, for the points in \( s_3 \in S(\gamma(U_R^1(i))) \), the shortest path between \( \gamma(U_R^1(i)) \) and \( \delta(U_R^1(i)) \) consists of three connected line segments, one from \( s_3 \) to \( v_2 \), another from \( v_2 \) to \( v_1 \) and the last one from \( v_1 \) to \( \delta(U_R^1(i)) \). Consider the points \( p \in \gamma(U_R^1(i)) \) for which the shortest path between them and \( \delta(U_R^1(i)) \) is not a line segment due to the presence of reflex vertices of the environment. As we can see in Figure 10 for any such \( p \) points, the shortest path from \( p \) to \( \delta(U_R^1(i)) \) must visit first a reflex vertex \( v_p \in \mathcal{V}(G) \) (\( v_1 \) or \( v_2 \) for instance) of the environment. Clearly, the union of such points \( p \) for which the first segment of the shortest path between them and \( \delta(U_R^1(i)) \) has \( v_p \) as an endpoint, is a subset of the union of all points in the plane that are equidistant to \( v_p \). Thus, they form an arc of circle, which is centered at \( v_p \) with radius \( d_p^1 - d(v_p, \delta(U_R^1(i))) \), where \( d(v_p, \delta(U_R^1(i))) \) is the length of the shortest path between \( v_p \) and \( \delta(U_R^1(i)) \). Therefore, every point in \( \gamma(U_R^1(i)) \) belongs to an arc of circle, so \( \gamma(U_R^1(i)) \) is the union of a set \( S(\gamma(U_R^1(i))) \) of arcs of circle.

**Lemma 7** Every \( \hat{U}_R^1(i) \) and \( U_R^1(i) \) is bounded by arcs of circle.

**Proof** Assume that \( \hat{U}_R^1(i) \) is bounded by arcs of circle. According to \( S_1 \), \( R_3^1 \) where \( T_j \in \frac{1}{\alpha + 1} \mathcal{T}_j(g_i) \), is the intersection of \( T_j \) and the complement of the region enclosed by \( \delta(U_R^1(i)) \) and \( \gamma(U_R^1(i)) \). Hence the boundary of \( R_3^1 \) consists of arcs of circle in \( S(\gamma(U_R^1(i))) \) and the edges of \( T_j \) (arcs of circle with center at infinity). Since the boundary of \( R_3^1 \) consists of arcs of circle, it follows that \( \delta(U_R^1(k)) \) also consists of arcs of circle from the definition of \( U_R^1(i) \). Consider the base case where \( \hat{U}_R^1(i) \) is the union of unsafe triangles. \( \delta(U_R^1(i)) \) is then a set of line segments, and the lemma holds. Consider the case of an unassigned guard \( g \) such that all the guards \( g_i \in \mathcal{G}(T_j) \setminus \{g_i\} \) have their regions \( U_R^1(k) \) and \( U_R^1(k) \) defined for each \( T_j \in \frac{1}{\alpha + 1} \mathcal{T}_j(g_i) \), where \( \alpha = 1 \) to 2. We assume that the result holds for the regions of those guards

![Fig. 10: Boundary \( \gamma(U_R^1(i)) \) decomposed in four different arcs of circle.](image-url)
B Generalization of Lemma 3

Lemma 8 If Algorithm 3 is never called for a specific input, Algorithm 4 is complete for such an input.

Proof The proof is by contradiction. We assume that Algorithm 3 does not find a feasible allocation but there exists one. Additionally, we assume that Algorithm 3 is never called during execution. In the proof of Lemma 3, we show the existence of a sequence $v_{i_1} s_{i_1} v_{i_2} s_{i_2} \ldots$ of vertices and guards in $G_1$ such that $g_{i_k} \in G(T_{i_k}) \setminus \{g_{i_{k-1}}\}$ and $R^2(i_k) \subseteq R^2(i_{k-1})$ such that $R^2(i) \subseteq R^2(i_{k})$ and $R^2(i_k) \subseteq R^2(i_{k+1})$. It is also stated that the sequence terminates if $g_{i_{k-1}} = G(T_{i_k})$ in which case $T_{i_k}$ is an unsafe triangle. The problem of finding the existence of an allocation that works when Algorithm 3 fails is reduced to the problem of showing that the aforementioned sequence of vertices and guards in $G^#$ does not terminate. Since the number of vertices in $G^#$ is finite, the sequence is stuck in a cycle $C$ in $G^#$ involving the vertices and the guards of the cycle in the sequence. Now prove that such a cycle $C$ cannot exist unless Algorithm 3 was called. First, we show that for the first vertex of the cycle $C$ that appears in the sequence, $v_{i_1}$, the pair of edges in $G^#$ incident to it are both incoming edges. If $v_{i_1}$ is the vertex that corresponds to the triangle where Algorithm 3 determined that it could not find an allocation, the claim is trivially proved since all edges incident to $v_1$ are incoming edges. Otherwise, if $v_{i_1}$ is no such vertex, then it corresponds to a vertex where all the incident edges are incoming edges excepting the edges that correspond to guard $g_{i_{k-1}}$. Notice that the sequence $v_{i_1} s_{i_1} v_{i_2} s_{i_2} \ldots$ follows a direction opposite to the orientation of the edges. Therefore, when the sequence is in $v_{i_k}$, the next guard cannot be $g_{i_{k-1}}$. It implies that both edges of $C$ incident to $v_{i_k}$ do not correspond to $g_{i_{k-1}}$, so they are by definition incoming edges. The sequence then continues with a different guard $g_{j_1}$ followed by a vertex $v_{i_{k+1}}$. By definition, the edge in $G$ incident to $v_{i_{k+1}}$, that corresponds to $g_{i_k}$, is an outgoing edge, so the next edge in $C$ corresponds to a different guard $g_{i_{k+1}}$, and its corresponding edge in $C$ is an incoming edge of $v_{i_{k+1}}$. Clearly, every vertex in $C$ has an outgoing and an incoming edge. Since $C$ is a cycle, vertex $v_{i_1}$ is eventually reached. However, it does not have an incoming and an outgoing edge in $G^#$ (both are incoming edges), which is not possible. Therefore, such a cycle does not exist in $G^#$. This contradicts the initial definition of $G^#$. Thus, at least one edge of $C$ was removed during the execution of Algorithm 3. This implies that Line 8 of Algorithm 3 was reached, which is impossible since Algorithm 3 was never called. The result follows.

C List of Variables

| Variable | Definition |
|----------|------------|
| $P$ | Polygon |
| $n$ | Number of vertices of $P$ |
| $\bar{v}_n$ | Maximum speed of intruder |
| $x_I$ | Location of intruder |
| $t$ | Time |
| $G$ | Set of guards |
| $\bar{v}_g$ | Maximum speed of guards |
| $r$ | Triangulation graph |
| $\bar{v}_1$ | Vertex set of graph $G$ |
| $E(G)$ | Edge set of graph $G$ |
| $T(G)$ | Set of guards incident to $T$ |
| $d(\cdot,\cdot)$ | Distance function |
| $A(g_1)$ | Non-safe triangles allocated to $g_1$ |
| $A(G)$ | Allocation of all non-safe triangles |
| $\bar{v}_i$ | Set of all $A(G)$ |
| $v_{\alpha}(i)$ | Set of non-safe triangles incident to $v_\alpha(i)$ |
| $s_{\alpha}(i)$ | Set of non-safe triangles |
| $w_{\alpha}(i)$ | Guard adjacency graph |
| $e_{\alpha}(g_1)$ | Edge of $G^#$ |
| $w_{\alpha}(k(i))$ | Weight of $e_{\alpha}(g_1)$ |
| $S_{\alpha}(i)$ | Set of representatives of $\bigcup_{\alpha \in \{1,\ldots,\|V(G^*)\}\} S_{\alpha}(T_{i})$ |
| $T_{\alpha}(i)$ | Region inside $T_{i}$ assigned to $g_{i}$ incident to $v_{\alpha}(i)$ |
| $S_{\alpha}(i)$ | Region allocated to $g_{i}$ |
| $R_{\alpha}^n(i)$ | Region allocated to $g_{i}$ |
| $U_{\alpha}^n(i)$ | Region inside $T_{k}$ that has not been assigned |
| $S_{\alpha}(i)$ | Region inside $T_{k}$ that cannot be assigned |
| $G_{\text{ready}}$ | Set of guards ready to be allocated |
| $G_{\text{alloc}}$ | Set of allocated guards |
| $G_{\text{alloc}}$ | Set of guards that cannot be allocated |
| $V_{\Omega}$ | Set of vertices corresponding to non-safe triangles that can be covered by guards in $G_{\text{alloc}}$ |
| $s_{\alpha}^{\text{internal}}(i)$ | Internal critical curve |
| $s_{\alpha}^{\text{external}}(i)$ | External critical curve |
| $C_1(i)$ | Critical region |
| $C_\Omega(i)$ | Extended critical region |
| $\|$ | Set of intruders |