Guarding a Polygon Without Losing Touch

Barath Ashok[1], John Augustine[1(✉)], Aditya Mehekare[2], Sridhar Ragupathi[2], Srikkanth Ramachandran[2], and Suman Sourav[3]

[1] Indian Institute of Technology Madras, Chennai, India
augustine@iitm.ac.in
[2] National Institute of Technology, Tiruchirappalli, India
[3] Advanced Digital Sciences Center, Singapore, Singapore

Abstract. We study the classical *Art Gallery Problem* first proposed by Klee in 1973 from a mobile multi-agents perspective. Specifically, we require an optimally small number of agents (also called guards) to navigate and position themselves in the interior of an unknown simple polygon with n vertices such that the collective view of all the agents covers the polygon.

We consider the *visibly connected setting* wherein agents must remain connected through line of sight links – a requirement particularly relevant to multi-agent systems. We first provide a centralized algorithm for the visibly connected setting that runs in time $O(n)$, which is of course optimal. We then provide algorithms for two different distributed settings. In the first setting, agents can only perceive relative proximity (i.e., can tell which of a pair of objects is closer) whereas they can perceive exact distances in the second setting. Our distributed algorithms work despite agents having no prior knowledge of the polygon. Furthermore, we provide lower bounds to show that our distributed algorithms are near optimal.

Our visibly connected guarding ensures that (i) the guards form a connected network and (ii) the polygon is fully guarded. Consequently, this guarding provides the distributed infrastructure to execute any geometric algorithm for this polygon.

Keywords: Art gallery problem · Mobile agents · Swarm robotics · Visibility · Line of sight communication

1 Introduction

The *Art Gallery Problem* is a classical computational geometry problem that seeks to minimize the number of guards (or agents in our context) required to guard an art gallery (represented by a simple polygon P comprising vertices $\{p_1, p_2, \ldots, p_n\}$). To successfully guard the gallery, every point inside the polygon

The authors are listed in alphabetical order.

A. W. Richa and C. Scheideler (Eds.): SIROCCO 2020, LNCS 12156, pp. 91–108, 2020.
https://doi.org/10.1007/978-3-030-54921-3_6

must be visible to at least one guard, i.e., for every point in the polygon, there must exist at least one guard such that the segment joining the point to the guard does not intersect the exterior of the polygon. This computational geometry problem was first posed by Klee in 1973, and thereafter has been widely studied over the years (see [16,26,30,31]).

In this paper, we investigate a variation of the problem called the *visibly connected art gallery problem* from a distributed multi-agents perspective. We require an optimally small number of agents (also called guards) with omni-directional vision to navigate an unknown simple polygon with n vertices in a coordinated manner and position themselves in its interior such that the collective view of all the agents covers the polygon. Additionally, for the connected art-gallery problem (as in [25]), it is required that the agents maintain line-of-sight connectivity. More precisely, the visibility graph [22,25] comprising agents as nodes and edges between agents that are within line of sight of each other (unobstructed by polygon edges) must be a connected graph.

The visibly connected art gallery problem was studied as early as 1993 by Liaw *et al.* [22] in the centralized setting, but only for the special case of spiral polygons. Hernandez-Penalver [19] considered simple polygons and showed that $\lfloor n/2 \rfloor - 1$ guards are sufficient and sometimes necessary. Pinciu [28] presented a centralized algorithm based on iteratively processing the dual graph of the polygon's triangulation. Although [28] lacks the analysis, one can infer that it runs in time linear in n, the number of vertices of the polygon. However, the algorithm is somewhat complicated and not amenable for parallel or distributed computing. Obermeyer et al. [25] provided a distributed algorithm that is capable of handling polygons with holes, but unfortunately requires $O(n^2)$ rounds.

Our work is motivated by recent advancements in unmanned aerial vehicles (UAVs), especially those capable of automated sensing (either through photogrammetry or LiDAR) and communication (typically through line-of-sight electromagnetic radio waves). Such UAVs are typically deployed into unknown territories from which they are required to navigate, learn, and perform useful tasks in a coordinated manner. We model these UAVs as point agents that start from a common starting point assumed without loss of generality (w.l.o.g.) to be p_1. Agents operate in synchronous rounds during which they can look, compute, communicate and move. They are required to coordinate with each other and achieve full visibility coverage of the polygon while maintaining line-of-sight connectivity with each other.

A connected visibility graph ensures that there exists a path between every pair of guards. So visibly connected guards can simultaneously maintain coverage of the polygon and execute distributed computing protocols through line of sight communication.

Our Contributions. We begin with a description of a centralized algorithm in Sect. 3 that takes a simple polygon P with n vertices as input and produces a placement of at most $\lfloor n/2 \rfloor - 1$ guards that satisfy the requirements of the visibly connected art gallery problem. This algorithm only requires $O(n)$ time. Here, we introduce a notion of triplets (three connected nodes) in the weak dual

graph $\mathcal{D}$ (defined formally in Sect. 2). Informally, $\mathcal{D}$ is the graph whose nodes are triangles of a triangulation of P and arcs connect pairs of triangles that share an edge. We show that $\mathcal{D}$ can be decomposed into $O(n)$ triplets that are connected and cover $\mathcal{D}$, and then we compute a set of visibly connected guards by placing guards – one per triplet – positioned strategically within the triangles pertaining to each triplet. Although Pinciu [28] has already presented an $O(n)$ algorithm, we believe that our algorithm is simpler and, more importantly, amenable to parallel computing. In particular, we show that our algorithm can be adapted to run in the PRAM model in time linear in the diameter of $\mathcal{D}$.

Next, we turn our attention to distributed computing models in which the agents are independent mobile computing entities that must interact with each other to solve the visibly connected art gallery problem. We define two model variants based on agents' perception capabilities. The *depth perception* variant wherein the agents can accurately perceive depth (i.e., distances) is inspired by UAVs with LiDAR technology [23]. On the other hand, the *proximity perception* variant only provides the agents with relative proximity. For concreteness, we limit the proximity perception variant to being able to sense which of any two objects (i.e., edges or vertices of the polygon) is closer. It is inspired by photogrammetry [1], which is cheaper and only guarantees coarser perception.

We present algorithms for both cases. The algorithm for the proximity perception variant, presented in Sect. 4, operates by exploring visible territories within the polygon (formally defined in Sect. 2). We describe two forms of exploration, one in a breadth-first manner and the other in a depth-first manner. Since the polygon structure is completely unknown to the agents, for each level of the breadth-first exploration, the nodes must communicate to the root in order to ensure that a sufficient number of agents are provisioned to explore that level. Our algorithm – taking the best out of both explorations –runs in $O(\min(\tilde{d}^2, n))$ rounds. The term $\tilde{d}$ is a natural notion of diameter of P called *minimal v-diameter* that we define formally in Sect. 4. Informally, it is the largest diameter among all visibility graphs pertaining to minimal placement of connected guards that cover all vertices in P. The candidate placements are minimal in the sense that the removal of guards either leads to lack of coverage or loss of connectivity.

When the agents can perceive depth, we exploit this capability to place agents based on the medial axis of the polygon P (defined formally in Sect. 2), which is a well-known tree-like structure that captures the "shape" of the polygon. Since depth perception is more powerful than proximity perception, we can take the best of all options to ensure a running time of $O(\min(\tilde{d}^2, D^2, n))$ time, where D is the (unweighted) diameter of the medial axis tree. Each of our algorithms require at most $O(n)$ agents for an initial placement, but a subsequent post-processing ensures that at most $\lfloor n/2 \rfloor - 1$ agents are placed in the polygon, which is optimal.

To complement our complexity claims, we consider the weaker problem wherein the robots are not required to be placed in a visibly connected guarding position, but rather just that the entire polygon must be explored by the agents.

The exploration problem only requires that for every point in P, some agent must have been within line of sight of that point at some time instant during the course of the algorithm. Clearly, any solution to the visibly connected guard placement problem will also be a solution for the exploration problem. Thus, we focus on showing a lower bound for the exploration problem. Specifically, we show that, for any deterministic algorithm operating on a polygon (with $\tilde{d}, D \in o(\log n)$) to even centrally coordinate $\Theta(n)$ agents to explore an initially unknown polygon, we can construct a polygon that requires $\Omega(D^2)$ (or $\Omega(\tilde{d}^2)$) communication rounds even with depth perception.

Unfortunately, due to space limitation, we have deferred some of our proofs to the full version [2].

Related Work. The classical art gallery problem was first introduced by Klee in 1973. Chvátal [7] showed by an induction argument that $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and occasionally necessary for any simple polygon with n vertices. Fisk [12] proved the same result via an elegant coloring argument. Lee and Lin in [20] proved that determining a set of minimum number of guards that can guard a given polygon is NP-hard. Consequently, researchers have focused on approximate solutions starting from an $O(\log n)$ approximation provided by Ghosh [17] in 1987, along with a conjecture that the problem admitted a polynomial time constant approximation algorithm. However, Eidenbenz et al. [9] showed that the problem was APX-hard, thereby precluding the possibility of a PTAS unless P = NP. After several improvements over the years, in 2017, Bhattacharya et al. [4] have reported constant factor approximation algorithms for the classical art gallery problem as well as for several well-studied variants.

In literature, based on the different restrictions placed on the shape of the galleries or the powers of the guards, several variations of "art gallery problems" have been studied. See [26,30], and [31] for details.

The *connected art-gallery problem* was first introduced by Liaw et al. in 1993 [22], where they refer to the problem as *minimum cooperative guards problem* and study it on k-spiral polygons (polygons with a maximal chain of k consecutive reflex vertices, i.e., vertices with internal angle >180°). It was also shown [22] that this problem is NP-Hard for simple polygons but can be solved in linear time in spiral and 2-spiral polygons (also see [29] for results on k-spiral graphs). For simple polygons with n vertices, Hernández-Penalver [19] proved by induction that $\lfloor \frac{n}{2} \rfloor - 1$ guards are always sufficient to obtain a connected guarding. Moreover, they also show that $\lfloor \frac{n}{2} \rfloor - 1$ guards are necessary for some polygons. The same result was also shown by Pinciu via an elegant coloring argument in [27].

In [21], Liaw et al. relax the strong connectivity condition from [22] and consider the case where there are no isolated vertices in the guards visibility graph. This problem of guarded guards where the overall connectivity of the guards visibility graph is non-essential has also been studied in [24,28].

In the distributed setting, Obermeyer et al. [25] study this problem in polygonal environment with holes. They first design a centralized incremental partition algorithm (defined therein) and from that obtain the distributed deployment algorithm by a distributed emulation of the centralized algorithm. The authors

give a deployment of agents that is guaranteed to achieve full visibility coverage of the polygon with n vertices and h holes in $O(n^2 + nh)$ time, given that there are at least $\lfloor \frac{n+2h-1}{2} \rfloor$ agents. This work closely relates to our work. While the scope of [25] includes polygons with holes, their algorithm is not optimized for time and their ideas lead to algorithms that require $O(n^2)$ communication rounds even for simple polygons without holes. Notable prior works with ideas leading to [25] can be found in [10,13–15].

Organization of the Paper. In Sect. 2, we present some preliminary definitions including several geometric definitions pertaining to polygons as well as formal definitions of the distributed computing models. In Sect. 3, we provide a centralized algorithm to solve the visibly connected art gallery problem and then show how to parallelize it. In Sect. 4 and Sect. 5, we present distributed algorithms under proximity perception and depth perception, respectively. We complement our upper bounds with a lower bound that is proved in Sect. 6. We then conclude with some remarks and future works in Sect. 7.

2 Preliminaries

Let P be a simple polygon with $n \geqslant 4$ vertices; we use P to refer to the polygonal region including both the interior and the boundary. In general, for a polygonal region $K \subseteq P$, we use ∂K for the set of vertices of P that lie on the boundary of this polygonal region. The ordered list of vertices of P are denoted $p_1, p_2, \ldots, p_n$, and thus, $\partial P \triangleq \{p_1, p_2, \ldots, p_n\}$. Each open line segment connecting p_i to $p_{i(\mathsf{mod}\ n)+1}$, $1 \leqslant i \leqslant n$, is denoted e_i. We assume that the vertices are in general position, i.e., (i) no three vertices are collinear, and (ii) no four vertices are co-circular. We use the term *object* to refer to either a vertex or an edge. Thus, the objects of P are $\{p_i\}_i \cup \{e_i\}_i$.

We use the notation $g \in P$ for some point g to indicate that g can either be a vertex, lie on an edge, or lie in the interior of P. Two points $g_1 \in P$ and $g_2 \in P$ are said to be in line of sight of each other if the open line segment $\overline{g_1 g_2}$ lies entirely within P. We use V_g^P (or just V_g when P is clear from context) to denote the visibility polygon of a point $g \in P$, which is defined as the subset of P that contains all points that are in line of sight from g. See Fig. 1 for an illustration.

We also borrow a useful definition from Obermeyer *et al.* [25] for *vertex-limited visibility polygon* $\bar{V}_g^P$ for a point $g \in P$ w.r.t. P, which is a modified form of V_g^P. Notice that V_g^P could have vertices that are not vertices in P; call such vertices *spurious vertices*. To get $\bar{V}_g^P$, perform the following operation repeatedly until there are no more spurious vertices: pick a spurious vertex v with predecessor p and successor s and crop the visibility polygon by cutting along the line segment $\overline{ps}$ and removing the portion that lacks the point g from further consideration. Note that either the predecessor or the successor may themselves be spurious. In Fig. 1, note the first vertex v that is clipped has successor s that is itself a spurious vertex. An edge that is in $\bar{V}_g^P$ but not in V_g^P is called a *gap*

edge. We also define a way to crop a polygon (cf. Fig. 1). Formally, for any pair of vertices p_i and p_j such that $\overline{p_ip_j}$ lies entirely within the interior of P and $c \in P \setminus \overline{p_ip_j}$, we define $\mathsf{crop}(P, c, p_i, p_j)$ to be the subset of P obtained by cutting along $\overline{p_ip_j}$ and discarding the part that contains c.

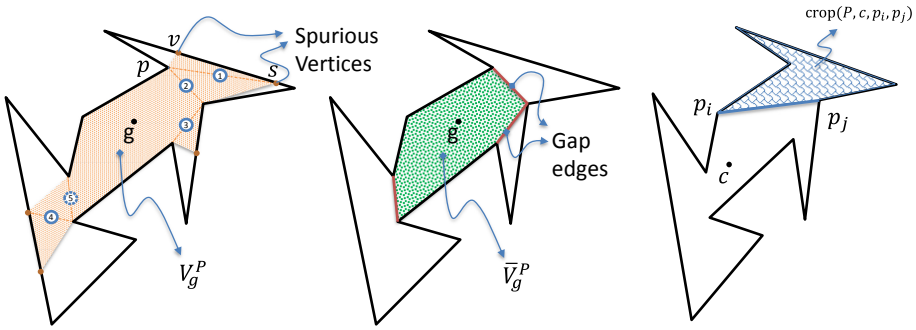

Fig. 1. Visibility polygon (left) and vertex-limited visibility polygon (at the center). The numbered line segments refer to one possible repeated sequence of cuts to arrive at vertex-limited visibility polygon. The polygon on the right depicts a cropped polygon.

Definition 1. *Let $G = \{g_1, g_2, \ldots\}$ be a set of points in P. We say that the points in G guard polygon P if $\cup_{g \in G} V_g = P$. In this context, we call the points in G as guards of P.*

The classical art gallery problem seeks to find a smallest possible set G of points that guard P, with variants including vertex and edge guarding.

In this paper, we consider a variant called the *connected art gallery problem* that, to the best of our knowledge, was introduced first by Liaw et al. [22]. In this variant, guards are connected in a suitable way, which we now formalize. We define the *visibility graph* of a set of points G within (and with respect to) P, denoted $\mathcal{G}_G^P$ (or just $\mathcal{G}$ when clear from context) as the graph with vertex set G. Two points in G are connected by an edge in $\mathcal{G}$ iff they are visible to each other within P. A set G of points in P is said to be connected (w.r.t. P) if $\mathcal{G}_G^P$ is connected. In the connected art gallery problem, we are required to compute a set G of points that guards P *and* the additional requirement that $\mathcal{G}_G^P$ is connected. It is well-known [3] that at most $\lfloor n/3 \rfloor$ guards are always sufficient to guard any polygon with n vertices. However, this bound does not hold under connected guarding.

Claim (consolidated from [27] *and* [28]*).* There exist orthogonal polygons with n vertices that require at least $n/2 - 2$ number of connected guards even if we only require them to guard the vertices of the polygon [28]. This bound increases mildly to $\lfloor n/2 \rfloor - 1$ for simple polygons with non-orthogonal edges [27].

We now define several structures associated with any polygon P. A line segment joining two vertices is said to be a *diagonal* if its interior lies entirely within the interior of P. It is easy to see that a maximal set of diagonals that do not intersect each other decomposes the polygons into a set of $n-2$ triangles called a *triangulation* of P [3]. A famous result by Chazelle [6] shows us how to find such a triangulation in $O(n)$ time. Given a triangulation T for a polygon P, the *weak dual graph* $\mathcal{D}_T^P$ (or just $\mathcal{D}$ when clear from context) is the graph (or more informatively, a tree) whose nodes are the triangles in T with edges in $\mathcal{D}_T^P$ between pairs of triangles that share a common triangle edge. Note that the weak dual graph is a tree where each tree node has a degree of at most 3.

Recall that, the term *object* refers to either a vertex or an edge of the polygon P. We define the medial axis M of the polygon P to be the (infinite) collection of points within P that are equidistant from at least two distinct objects of P (see Fig. 2(iii) for an illustration).

Claim. For any simple polygon P, the medial axis is a tree whose leaves are convex vertices of P, i.e., vertices with internal angle being less than 180°.

Note that reflex vertices (i.e., vertices with internal angles greater than 180°) cannot be nodes in the medial axis tree. An edge in M is a non-empty maximal set of points that are equidistant between the same set of objects. When the two objects are of the same type (either both polygonal edges or both vertices), the corresponding medial axis edge will be a straight line segment. On the other hand, if one of the objects is a vertex and the other is a polygonal edge, the corresponding medial axis edge will be a parabolic arc. The endpoints of medial axis edges are the *medial axis nodes* or just nodes. Since the number of leaves is at most n, we get:

Claim. The number of nodes in the medial axis of P will be $O(n)$.

We use D^P (or just D when clear from context) to refer to the unweighted diameter of the medial axis M. More precisely, D^P is the maximum number of edges over all paths in the medial axis tree of P.

2.1 Computational Models

We focus on the connected art gallery problem in the classical sequential setting. In this case, we assume that the sequence of vertex points $(p_1, p_2, \ldots, p_n)$ are given in order, say, as an array of points. However, for the connected art gallery problem as inspired by mobile agents that operate in a spatially distributed setting, we employ a distributed computing model based on the work by Obermeyer *et al.* [25]. For clarity, we assume a synchronous model with time discretized into a sequence of rounds and the agents executing a look-communicate-move cycle in each synchronous round; local computation is interspersed between the look-communicate-move cycles.

We assume that there are $\Theta(n)$ agents. Agents (modeling mobile robots) can be represented as points in the plane and as a result multiple agents can be

co-located at the same point. Without loss of generality, assume all agents start at the same vertex somewhere in P. Furthermore, agents can only move from one vertex p_i to another vertex p_j provided $\overline{p_i p_j}$ is a diagonal in P (i.e., p_i and p_j have direct line of sight to each other). We assume that agents have unique IDs from $\{1, 2, \ldots, n\}$. Each agent g performs the following tasks in each round.

Look. The agent g first orients itself to start from a particular direction (in the direction of another vertex called its *orientation vertex*) and perform a 360° clockwise sweep during which it creates a view of V_g^P. The level of information that the agent can gather depends on whether the agents have depth perception or not. With depth perception, the view is simply the full visibility polygon V_g^P. Without depth perception, however, the view is limited to a sequence of alternating vertices and edges (possibly gap edges) starting from its orientation vertex. In both cases, g can also see other agents that are inside V_g.

Communicate. Two agents can communicate as long as they are visible to each other (which includes co-located agents). Communication is via message passing. Each agent can send at most one message to each agent that it can see.

Move. This step again differs based on whether agents can perceive depth or not. Let us first consider the case when agents can perceive depth. Based on the outcome of the communication and computation, each agent g chooses to move from its current location to a new location within its current visibility polygon. We assume that the agent – once it reaches its destination location – can "remember" its source position in the sense that it can spot the source location in its view after it reaches its destination. The only restriction when agents cannot perceive depth is that they are limited to moving to vertices of the polygon. For this reason, we always assume that agents will be on polygonal vertices (even at the start of time) when they cannot perceive depth.

3 Centralized Sequential and Parallel Algorithms

We first present a centralized sequential algorithm for the connected art gallery problem and then briefly show how it can be parallelized. Our approach is to decompose the weak dual graph into a suitable set of at most $\lfloor n/2 \rfloor - 1$ connected triplets and then assign a guard for every triplet. The high-level steps are outlined in Algorithm 1. Consider the weak dual graph $\mathcal{T}$, which is of course a tree with maximum degree 3. Root the tree at some node r that is of degree 1. A *triplet* is any set of three nodes in the tree that are connected. We now show a simple procedure (cf. Algorithm 2 and Fig. 2(ii) for an illustration) to decompose $\mathcal{T}$ into triplets. Subsequently, we will prove some properties of these triplets that will immediately lead us to the required centralized algorithm for the connected art gallery problem.

Lemma 1. *When two triplets share at least one common node, their associated guards can see each other.*

Algorithm 1. Centralized algorithm for the connected art gallery problem.

1: Compute a triangulation T of the polygon P [6] and then compute the weak dual graph.
2: Decompose the weak dual graph into at most $\lfloor n/2 \rfloor - 1$ triplets, i.e., groups of three connected nodes in $\mathcal{T}$, as described in Algorithm 2.
3: Each triplet corresponds to three triangles arranged in such a way that there is a middle triangle that shares two edges, say a and b, with the other two triangles. Placing a guard at the common vertex between a and b for every triplet is the required solution. (See Figure 2(i) for an illustration.)

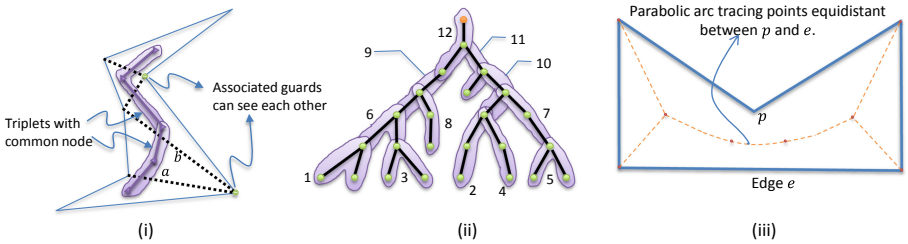

Fig. 2. (i) Illustrated placement of guards associated with triplets. Notice that the guards are associated with the two triplets sharing a common node. Consequently, they can see each other. (ii) Sequence of triplets computed at the end of each iteration of the `for` loop in Algorithm 2. (iii) Medial axis of a polygon.

Algorithm 2. Algorithm to decompose $\mathcal{T}$ into triplets.

Require: A tree $\mathcal{T}$ rooted at a node r of degree 1, with max degree three, and depth L. Note that r is assumed to be at level 0, so there are L levels from r to the farthest leaf (inclusive).
Ensure: A collection of triplets.
1: Color all internal nodes red and all leaves orange.
2: **for** $\ell \leftarrow L$ down to 2 (decrementing by 1 every iteration) **do**
3: **while** $\exists$ orange node v at level ℓ **do**
4: **if** v has a sibling v' that is also colored orange **then**
5: Form a new triplet comprising v, v', and their common parent node p.
6: Color parent p orange.
7: Color v and v' green.
8: **else**
9: Form a triplet comprising v, parent p of v, and the grandparent p' of v.
10: Color p' orange.
11: Color v and p green.
12: **end if**
13: **end while**
14: **end for**
15: **if** the root is not part of some triplet **then**
16: Form a triplet comprising the root, its child, and an arbitrarily chosen grandchild. Color all three nodes green.
17: **end if**

Having presented the algorithm to solve the connected art gallery problem in this centralized setting, we move on to analyze the algorithm. Our main focus will be on analyzing Algorithm 2. We make a series of observations formalized as lemmas and then derive the result as a consequence. For a given set of triplets, we define the *triplets graph* to be the graph with the triplets as vertices and edges between pairs of triplets that share at least one edge. We say that the set of triplets covers the tree $\mathcal{T}$ if every node is part of at least one triplet. The proof of the following lemma is deferred to the full version.

Lemma 2. *Given* $\mathtt{t}$ *is the number of nodes in* $\mathcal{T}$*, we claim that Algorithm 2*

1. *forms a set of triplets that covers* $\mathcal{T}$*,*
2. *guarantees that the triplets graph is connected,*
3. *guarantees that the number of triplets formed is at most* $\lfloor \mathtt{t}/2 \rfloor$*, and*
4. *runs in* $O(\mathtt{t})$ *time.*

Recalling that $\mathtt{t} = n - 2$, we can conclude that the sequential algorithm runs in $O(n)$ time.

Finally, we remark that the algorithm described above can be implemented in parallel specifically in the Parallel Random Access Machine (PRAM) model. In PRAM, we have several processors that operate on a shared addressable memory space. Typically, concurrent reading (i.e., multiple processors reading a word simultaneously) is acceptable but writing requires exclusivity (i.e., ensuring that at most one processor can write to a word in one time step); this is the concurrent read exclusive write (CREW) version of PRAM. Goodrich [18] has already shown how to triangulate P in $O(\log n)$ time under CREW PRAM. We logically assign one processor per triangle and ensure the parent-child relationship between triangles is extended to the processors. Then, each iteration of the `for` loop in Algorithm 2 (comprising several `while` loop iterations) can be executed in parallel. Redundant triplets that can occur when we form triplets connecting two orange siblings (see line number 4 in Algorithm 2). This can be avoided in such situations by only allowing processors corresponding to the left children to form the triplet.

Theorem 1. *Supported by Algorithm 2, Algorithm 1 solves the connected art gallery problem with at most* $\lfloor n/2 \rfloor - 1$ *guards in time that is linear in* n*. Moreover, can be solved in the CREW PRAM model with at most* $\lfloor n/2 \rfloor - 1$ *guards in time that is linear in the diameter of the weak dual graph associated with the triangulation of* P*.*

4 Distributed Guarding with Proximity Perception

In this section, we consider the case where each agent is able to distinguish the proximity or relative distances (without knowing the actual distances) between the various objects associated with the polygon as well as with other agents, etc. which are in its visibility polygon at any specified moment. Specifically, the agents' `"look"` paradigm is reflective of real-world sensing techniques where

absolute distances to objects in the scene are unavailable whilst their relative distances can be inferred such as with photogrammetric vision in drones.

We give a distributed solution that solves the connected art-gallery problem and runs in $O(\min(\tilde{d}^2, n))$ rounds, where $\tilde{d}$ is the minimal v-diameter that we formally define later in this section. Our solution comprises two algorithms that are executed in parallel, one in a breadth first manner and the other in a depth first manner. Our final solution is to take the best out of both explorations. Due to space limitation, we will describe the breadth first algorithm that runs in $O(\tilde{d}^2)$ rounds in detail. We subsequently present a brief overview of the depth first exploration and defer further details to the full version [2]. Finally, we conclude with some remarks on how our algorithm can form the basis for solving other polygon problems on P.

Assuming that the vertices and the edges defining the polygon P are in general position, the agents start at some vertex in P, which we can assume w.l.o.g. to be p_1. The algorithm operates in phases. At the end of a particular phase ℓ, a subset S_ℓ of the agents have "settle" into their final positions while establishing a connected guarding of the subset P_ℓ of the polygon. For any agent i, its settled position is a vertex in P and is denoted s_i.

Moreover, the settled agents are arranged in the following hierarchical manner. W.l.o.g., let the root be agent 1, settled at p_1. We define the territory of the root, i.e., agent 1, to be $\mathsf{territory}(1) \triangleq \bar{V}^P_{p_1}$. Every other settled agent j has a parent agent $\mathsf{parent}(j)$. If agent $i = \mathsf{parent}(j)$, then we say that j is the child of i denoted as $\mathsf{child}(i)$. Each parent agent i has one child agent j per gap edge in its $\mathsf{territory}(i)$ and the child is located at one of the end points, say p_a, of a the gap edge (p_a, p_b). Thus, $s_j = p_a$. The other end of the gap edge p_b is denoted $\mathsf{orient}(j)$; intuitively, j settles at s_j and orients itself towards $\mathsf{orient}(j)$ for performing "look" operations.

Furthermore, $\mathsf{territory}(j) \triangleq \bar{V}^P_{s_j} \cap \mathsf{crop}(P, s_{\mathsf{parent}(j)}, s_j, \mathsf{orient}(j))$ i.e., $\mathsf{territory}(j)$ is the portion of s_j's vertex limited visibility polygon not containing s_j's parent and truncated by the gap edge that originated it. (See Fig. 3.) Intuitively, each agent j is only responsible for guarding its $\mathsf{territory}(j)$. Notice that by definition, territories of a parent and its child do not overlap (they share a bordering edge that is a gap edge seen originally by the parent). Moreover, territories of children of a given parent do not overlap as well (at most they share a vertex) as this would imply that the polygon contains holes, i.e., it is non-simple. For correctness, we need $\cup_j \mathsf{territory}(j) = P$, which is immediate from our algorithm.

Next, we define a *territory tree* to be a tree in which nodes are territories and edges are pairs of territories that share a common diagonal (gap) edge. Let T^* be the set of all possible territory trees that can be achieved given all possible options for the starting vertex p_1 and all possible choice of placement of child agents. Then, we define d as the maximum diameter of all such territory trees, i.e., $d \triangleq \max_{T \in T^*} \mathsf{diameter}(T)$.

Initially, $S_0 = \{1\}$ (w.l.o.g.), $s_1 = p_1$, and P_0 is simply $\bar{V}^P_{s_1}$. We are now ready to present the steps to be performed within each phase ℓ; notice that there cannot

be more than d phases to the algorithm hence, $1 \leqslant \ell < d$. Intuitively, in each phase ℓ (see Algorithm 3), we incrementally construct the territories at level ℓ of the territory tree.

Algorithm 3. Phase $\ell \geqslant 1$ of the distributed algorithm for the connected art gallery problem that may use more than $\lfloor n/2 \rfloor - 1$ guards. This description assumes phases 0 to $\ell - 1$ have completed and each agent $i \in S_{\ell-1}, \ell > 1$, remembers one marked vertex. (This marking scheme ensures that a child and grandparent are not visible to one another.)

1: Every settled agent $i \in S_{\ell-1}$ performs a *look* operation into its $\mathsf{territory}(i)$ and counts the number of gap edges in its $\mathsf{territory}(i)$. Call this count b_i. Each settled agent i now up-casts b_i to the root with intermediate settled agents aggregating the quantities by adding up the numbers sent by their children.
2: Notice that at the end of the up-casting, the root will know the total number b of gap edges. The root apportions b new agents and sends them to its children according to the numbers sent by each child. Subsequently, whenever a settled agent notices new agents reaching its position, it will apportion the agents according to numbers sent by its children and the new agents will move to their assigned child of the current settled agent.
3: Each settled agent $i \in S_{\ell-1}$ whose $\mathsf{territory}(i)$ has some $b_i > 0$ gap edges gets exactly b_i new agents. Agent i assigns each of those new agents j to an unmarked vertex of each such gap edge and consequently, agent j marks the other vertex of that gap edge.

Lemma 3. *Repeating Algorithm 3 until all levels of the territory tree are explored, we get a distributed algorithm that, with no more than n agents, ensures that the agents position themselves in a manner that solves the connected art gallery problem. The round complexity is* $O(d^2)$.

Now, we introduce the notion of *minimal visibility connected vertex guarding* (henceforth referred to as *minimal v-guarding*) which is pivotal in this case for developing algorithmic bounds on the running time. Let P be a simple polygon with n vertices. Recall that, given a set of labelled guards $G = \{g_1, g_2, \ldots, g_k\}$ of P, we associate with G a unique graph $\mathcal{G}$ with a vertex set of size k such that when two guards are visible to each other, then they are connected by an edge in this graph $\mathcal{G}$ i.e., $e = \{i, j\} \in E[\mathcal{G}] \iff g_i$ is visible to g_j. We say that G is a *minimal v-guarding* or a *minimal v-configuration* of P whenever the following holds, $\forall v \in \mathcal{G}$, *at least* one of the following two conditions applies:

1. $\mathcal{G} - v$ has more than a single component.
2. The vertex guarding $G_{-v} := G \backslash \{g_v\}$ of P is incomplete, i.e., $\cup_{g \in G_{-v}} \partial V_g \subsetneq \partial P$ where ∂P is the set of vertices of polygon P and ∂V_g refers to the set of vertices of P visible from g.

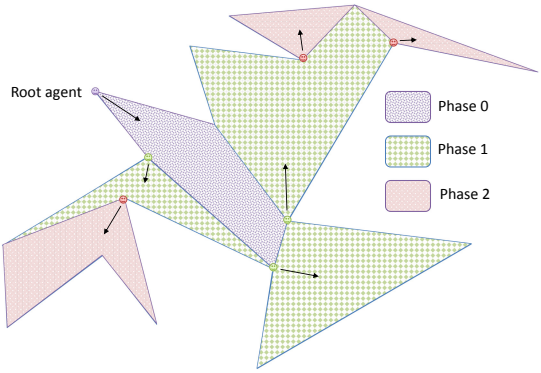

Fig. 3. Depicts the placement of agents and their respective territories at three levels. The arrows point each agent into its territory.

We then define the notion of *minimal v-diameter* $\tilde{d} := \max_{\tau \in \mathcal{M}} \mathsf{diameter}(G_\tau)$ where $\mathcal{M}$ is the set of all possible minimal v-configurations of P and G_τ denotes the associated visibility graph of such a guarding τ from $\mathcal{M}$.

In order that we may successfully compare the efficiency of our algorithms with one another we use $\tilde{d}$ which is a polygonal parameter pertinent to our guarding problem.

Lemma 4. *$d = O(\tilde{d})$, i.e., the diameter of any territory tree on P is asymptotically bounded by the minimal v-diameter of the Polygon P.*

Using Lemma 3 along with Lemma 4 yields the following theorem:

Theorem 2. *There exist a distributed algorithm that solves the connected art gallery problem in $\mathcal{O}(\tilde{d}^2)$ time using no more than n agents limited to proximity sensing capability, where $\tilde{d}$ refers to the minimal v-diameter of P and n is the number of polygon vertices.*

An $O(n)$ Round Algorithm. While most real-world polygons may have small diameters, it is nevertheless conceivable that $\tilde{d} \in \Omega(n)$ in some cases like spirals. The algorithm that we have presented above will unfortunately require quadratic in n number of rounds for such situations, which is undesirable. In the full version [2], we describe a depth first procedure that only requires $O(n)$ rounds.

Remark 1. Both the $O(\tilde{d}^2)$ algorithm and the depth first $O(n)$ algorithm will maintain agents connected by line of sight. So we can start both algorithms simultaneously. When one of them – the winner – finishes, the other can be terminated by a terminate message that will reach all agents in at most $O(\min(\tilde{d}, n))$.

General Problem Solving Given a Visibly Connected Guard Placement. In the breadth first scenario, since the agents are settled into a visibly

connected guarding position and are aware of their respective territories in the territory tree, they can gather at the root's position in a bottom-up fashion. Thus, the root, in $O(\tilde{d})$ rounds, can collect the views of all the agents and perform computation using the collective views of the agents. Similarly, taking into account the depth first scenario, we get the following generalized theorem.

Theorem 3. *Suppose $\mathcal{P}$ is a computationally tractable problem that takes a polygon P as input and either*

- *outputs information in the form of bits*
- *or requires placing available agents in positions within P.*

Then, $\mathcal{P}$ can be solved in our distributed context in $O(\min(\tilde{d}^2, n))$ rounds.

5 Distributed Guarding with Depth Perception

In this section, we give a distributed algorithm that solves the connected art gallery problem in $O(D^2)$ rounds, where D is the (unweighted) diameter of the medial axis. This algorithm's advantage is that its running time depends on D, which is more well-known than $\tilde{d}$. However, as opposed to the previous section, in this case, agents require the ability to perceive depth. The key idea of the algorithm here is if agents were placed on all internal nodes of the medial axis and some specially chosen vertices, they cover the entire graph as well as remain visibly connected.

Computing Adjacent Nodes in the Medial Axis. Imagine an agent at any point x on the medial axis. The agent can simulate the creation of a maximal disc at x to find the objects (vertices or edges) that determine x, i.e., the objects due to which x is a part of the medial axis. There would be at least two such objects that determine x. If x is determined by multiple objects (>2), it implies that x itself is a node on the medial axis, and we can consider any two consecutive objects determined by the **look** operation. For example, if a, b, c, d are 4 objects, that determine x and are ordered in accordance with the look operation, we consider the pairs ab, bc, cd and da only. Note that, only the consecutive object pairs determine the medial axis edges incident at x, and hence only those are considered.

For each pair of objects ob_1 and ob_2, there can only be three possible cases; either both are vertices, both are edges, or one of them is a vertex while the other is an edge. For all the cases, the agent at x is aware of the structure of the medial axis from x. Thus, if both ob_1 and ob_2 are vertices, then the next node of the medial axis lies on the perpendicular bisector of the line segment (ob_1, ob_2). If both ob_1 and ob_2 are edges, then the next node of the medial axis lies on the angle bisector of ob_1 and ob_2. Lastly, w.l.o.g. if ob_1 is a vertex and ob_2 is an edge, then the next node of the medial axis lies on the parabola determined by ob_1 and ob_2. Since agents have infinite computing power and depth sensing ability, they can progressively simulate maximal discs along the medial axis structure (perpendicular bisector, angle bisector or parabola) until the maximal

disc encounters a new object (say ob_3). The center of the maximal disc at this instance determines the next adjacent node in the medial axis. We define the set of new adjacent nodes obtained in phase i as A_i.

Agent Placement. As in Algorithm 3, when an already placed agent a determines its adjacent set of positions on which new agents are to be placed, then a upcasts the request of the required number of agents up to the root (the spot initially containing all the agents) with intermediate agents aggregating the quantities by adding up the numbers sent by their children. The root serves the request by assigning the required number of agents. The assigned agents trace back the path to a and thereafter get placed in their determined spot.

Algorithm 4. An $O(D^2)$ time algorithm for the connected art gallery problem.

1: Starting from the initial given vertex v where all the agents are placed, a medial axis point m is determined. If v is convex, then m is given by v's adjacent node in the medial axis. Alternatively, if v is a reflex vertex, pick the nearest visible new object ob_3 (determined by the depth sensing ability of the agents) not including v and choose the center of the maximal disc determined v and ob_3 as the point m on the medial axis.
2: Consider m as the root. All agents are moved here.
3: Determine all the adjacent medial axis nodes from m (i.e., the set A_1). * *This marks the end of the first phase. Each iteration of the loop represents a subsequent phase. The algorithm continues until the entire medial axis is uncovered.* *\
4: **for** each new adjacent node $x \in A_{i-1}$ determined in the previous phase, that is not a leaf node of the medial axis tree **do**
5: Compute set A_i (current set of new adjacent medial axis nodes of x) in parallel.
6: Place an agent at each node $y \in A_i$ except when y corresponds to a convex vertex of the polygon (leaf node of the medial axis tree).
7: **if** y is a part of a parabola determined by a polygon vertex and an edge and the polygon vertex does have an agent on it **then**
8: Place an agent on the reflex vertex determining the parabola.
9: **end if**
10: **end for**

Lemma 5. *Algorithm 4 gives a visibly connected guard placement while ensuring that the entire polygon is guarded/covered.*

Theorem 4. *There exists a distributed algorithm that solves the connected art gallery problem in $\mathcal{O}(D^2)$ time using no more than n agents, where D refers to the medial axis diameter and n is the number of polygon vertices.*

To reduce the final number of guards placed, we use similar procedure as described in Sect. 4. This gives us the following theorem.

Theorem 5. *There exists a distributed algorithm that uses no more than n agents to compute the placement of at most $\lfloor n/2 \rfloor - 1$ guard agents in a visibly connected manner, when the agents have depth sensing ability. Moreover, this algorithm takes at most $O(D^2)$ communication rounds.*

6 Lower Bound

In this section, we give lower bounds for a slightly weaker polygon exploration problem that requires for every point in P, that some agent must have been within line of sight of that point at some time instant during the course of the algorithm. Clearly, any solution to the visibly connected guard placement problem will also be a solution for the exploration problem. The lower bounds highlight the criticality of parameters like the medial axis diameter D and the minimal v-diameter $\tilde{d}$ for solving the connected art-gallery problem. The main result is summarized by the following Theorem.

Theorem 6. *For every deterministic distributed guard placement algorithm A with depth perception (resp., proximity perception) there exists a polygon P with medial axis diameter $D \in o(\log n)$ (resp., with minimal v-diameter $\tilde{d} \in o(\log n)$) such that A requires $\Omega(D^2)$ time (resp., $\Omega(\tilde{d}^2)$ time) to place the guards even when A is provisioned with a number of guards that is $\Theta(n)$.*

We show a reduction from any instance of the well-studied tree exploration game [8] to the problem of exploring a polygon, which our guarding problem subsumes. We sketch our approach here and defer details to the full version [2].

1. Firstly, we embed the tree in the Euclidean plane such that no two adjacent edges form an angle of 180°.
2. We *thicken* the edges of the embedded tree and form a simple polygon.
3. With the embedding and thickening transformations, we reduce the problem of collaborative exploration of the underlying tree to the guard placement in the obtained polygon.

7 Conclusion and Future Works

In this paper, we have presented centralized and distributed algorithms for computing a visibly connected guard placement. Crucially, our algorithms take time that is quadratic in a couple of different notions of diameters of P, i.e., $\tilde{d}$ and D. We believe that $\tilde{d} \in O(D)$, thereby obviating the need for precise depth perception, but the proof has eluded us. It would be nice to establish this formally as this would lend to understanding the trade off between LiDAR and photogrammetry (see [5,11] for example). We also remark that our algorithms have been explained in the synchronous setting for the purpose of clarity, but they can be easily extended to the asynchronous setting. Additionally, it will be interesting to extend our works to polygons with holes or polyhedra in higher dimensions.

Acknowledgements. Barath Ashok and John Augustine were supported in part by DST/SERB Extra Mural Grant (file number EMR/2016/00301) and DST/SERB MATRICS Grant (file number MTR/2018/001198). Suman Sourav was supported in part by the National Research Foundation, Prime Minister's Office, Singapore under the Energy Programme and administrated by the Energy Market Authority (EP Award No. NRF2017EWT-EP003-047). Part of this work was done when Aditya Mehekare, Sridhar Ragupathi, Srikkanth Ramachandran and Suman Sourav visited IIT Madras. We thank Rajsekar Manokaran for pointing out LiDAR and Photogrammetry.

References

1. Aber, J.S., Marzolff, I., Ries, J.B., Aber, S.E.: Principles of photogrammetry. In: Aber, J.S., Marzolff, I., Ries, J.B., Aber, S.E. (eds.) Small-Format Aerial Photography and UAS Imagery, pp. 19–38. Academic Press, Cambridge (2019). (Chapter 3)
2. Ashok, B., Augustine, J., Mehekare, A., Ragupathi, S., Ramachandran, S., Sourav, S.: Guarding a polygon without losing touch (2020). https://arxiv.org/abs/2005.05601
3. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77974-2
4. Bhattacharya, P., Ghosh, S.K., Pal, S.: Constant approximation algorithms for guarding simple polygons using vertex guards (2017)
5. Buczkowski, A.: Drone lidar or photogrammetry? Everything you need to know. https://geoawesomeness.com/drone-lidar-or-photogrammetry-everything-your-need-to-know
6. Chazelle, B.: Triangulating a simple polygon in linear time. Discrete Comput. Geom. **6**(3), 485–524 (1991). https://doi.org/10.1007/BF02574703
7. Chvátal, V.: A combinatorial theorem in plane geometry. J. Comb. Theory, Ser. B **18**(1), 39–41 (1975)
8. Disser, Y., Mousset, F., Noever, A., Skoric, N., Steger, A.: A general lower bound for collaborative tree exploration. Theor. Comput. Sci. **811**, 70–78 (2018)
9. Eidenbenz, S., Stamm, C., Widmayer, P.: Inapproximability results for guarding polygons and terrains. Algorithmica **31**(1), 79–113 (2001). https://doi.org/10.1007/s00453-001-0040-8
10. Fekete, S.P., Kamphans, T., Kröller, A., Mitchell, J.S.B., Schmidt, C.: Exploring and triangulating a region by a swarm of robots. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 206–217. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22935-0_18
11. Filippelli, S.K., Lefsky, M.A., Rocca, M.E.: Comparison and integration of lidar and photogrammetric point clouds for mapping pre-fire forest structure. Remote Sens. Environ. **224**, 154–166 (2019)
12. Fisk, S.: A short proof of Chvátal's watchman theorem. J. Comb. Theory, Ser. B **24**(3), 374 (1978)
13. Ganguli, A.: Motion coordination for mobile robotic networks with visibility sensors. Ph.D. thesis, University of Illinois at Urbana-Champaign (2007)
14. Ganguli, A., Cortes, J., Bullo, F.: Distributed deployment of asynchronous guards in art galleries. In: 2006 American Control Conference, p. 6, June 2006
15. Ganguli, A., Cortes, J., Bullo, F.: Visibility-based multi-agent deployment in orthogonal environments. In: American Control Conference, July 2007

16. Ghosh, S.: Visibility Algorithms in the Plane. Cambridge University Press, Cambridge (2007)
17. Ghosh, S.K.: Approximation algorithms for art gallery problems. In: Proceedings of Canadian Information Processing Society Congress, pp. 429–434 (1987)
18. Goodrich, M.T.: Triangulating a polygon in parallel. J. Algorithms **10**(3), 327–351 (1989)
19. Hernández-Penalver, G.: Controlling guards. In: CCCG, pp. 387–392 (1994)
20. Lee, D., Lin, A.: Computational complexity of art gallery problems. IEEE Trans. Inf. Theory **32**(2), 276–282 (1986). https://doi.org/10.1109/TIT.1986.1057165
21. Liaw, B., Lee, R.C.T.: An optimal algorithm to solve minimum weakly cooperative guards problem for 1-spiral polygons. Inf. Process. Lett. **52**(2), 69–75 (1994)
22. Liaw, B.C., Huang, N.F., Lee, R.C.T.: The minimum cooperative guards problem on k-spiral polygons. In: CCCG (1993)
23. McManamon, P.: LiDAR Technologies and Systems. SPIE Press, Bellingham (2019)
24. Michael, T.S., Pinciu, V.: Art gallery theorems for guarded guards. Comput. Geom. Theory Appl. **26**(3), 247–258 (2003)
25. Obermeyer, K.J., Ganguli, A., Bullo, F.: Multi-agent deployment for visibility coverage in polygonal environments with holes. Int. J. Robust Nonlinear Control **21**(12), 1467–1492 (2011)
26. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, Oxford (1987)
27. Pinciu, V.: A coloring algorithm for finding connected guards in art galleries. In: Calude, C.S., Dinneen, M.J., Vajnovszki, V. (eds.) DMTCS 2003. LNCS, vol. 2731, pp. 257–264. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45066-1_20
28. Pinciu, V.: Connected guards in orthogonal art galleries. In: Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P. (eds.) ICCSA 2003. LNCS, vol. 2669, pp. 886–893. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44842-X_90
29. Deogun, J.S., Sarasamma, S.: On the minimum co-operative guard problem. J. Comb. Math. Comb. Comput. (JCMCC) **22**, 161–182 (1996)
30. Shermer, T.C.: Recent results in art galleries (geometry). Proc. IEEE **80**(9), 1384–1399 (1992). https://doi.org/10.1109/5.163407
31. Urrutia, J.: Art gallery and illumination problems. In: Handbook of Computational Geometry, North-Holland, pp. 973–1027 (2000). (Chapter 22)