International Journal of Innovative Computing, Information and Control
Volume **13**, Number **3**, June **2017**

ICIC International ©2017 ISSN 1349-4198
pp. 921–940

GENERALIZED COVERAGE SOLVER USING HYBRID EVOLUTIONARY OPTIMIZATION

Igi Ardiyanto[1] and Jun Miura[2]

[1]Department of Electrical Engineering and Information Technology
Universitas Gadjah Mada
Jl. Grafika No. 2, Yogyakarta 55281, Indonesia
igi@ugm.ac.id

[2]Department of Computer Science and Engineering
Toyohashi University of Technology
Tenpaku-cho, Hibarigaoka 1-1, Aichi-ken 441-8580, Japan
jun.miura@tut.jp

Received September 2016; revised January 2017

Abstract. *This paper addresses a novel algorithm called Generalized Hybrid Evolutionary Coverage Solver (GHEC-Solver) for solving the coverage problem. We extend and generalize the concept of the Art Gallery Problem to accommodate several applications, ranging from a set of security camera placements to the robot scanning positions for covering the entire environment. We first simplify the environment map into a polygon and build guard candidates utilizing the topological features. The polygon arrangement of each guard visibility is established and an optimization based on a non-unicost Set Covering Problem (SCP) is performed to obtain an optimal set of the guard candidates. Unlike the original Art Gallery Problem, our algorithm takes account of the cost of each guard imposed from the environment or the problem settings (such as vertex guards, interior guards, and cost function-based guards) directly in the non-unicost SCP, so that it can cope with many classes of the coverage problem. The algorithm then alternates the non-unicost SCP and a probabilistic evolutionary optimization to obtain the optimal guards. Several tests using various environment map-based polygons are conducted, and the results support the robustness of our proposed algorithm.*

Keywords: Sensor coverage, Robot guards, Set Covering Problem, Evolutionary optimization, Surveillance

1. Introduction.

Suppose we want to put several surveillance cameras to see the entire building. Due to the budget limitation, we also want to reduce the number of cameras as minimum as possible. The problem now becomes how many cameras should be used and where it should be placed. This question is basically the essence of the *Art Gallery Problem*.

The Art Gallery Problem [1], as appearing in several textbooks (e.g., [2-4]), is a classical problem which inquires the minimum number of guards which should be placed in a polygon ensuring the full coverage of the entire polygon. It is closely related to the sensor coverage and sensor placement problem, so we are not surprised for finding several works which blend both problems into one topic, such as [5-8].

There are a lot of practical cases in the real world which rely on the Art Gallery or sensor coverage problem and its variants. The surveillance camera placement above is one example. The other related cases are how to determine the efficient sensor placement and coverage problem in a *sensor network* (e.g., [9-11]) and *multi agent deployment* for the building inspection [12].

Each problem above has a unique characteristic. For instance, the cameras can be located at any place of the building, even on the edge of the walls or the corners, depending on the camera model. Yet, there are some occasions that we prefer to put the camera, such as the omnidirectional camera, on the middle of the room's ceiling. While those problems are usually solved for each case (such as vertex guards [13], edge guards, and interior guards [14]), here we aim to establish a generalized framework which can be applied for every case which includes the placement preferences.

1.1. **Our contributions.** Our objective is to develop a unified framework for solving the coverage problem in different kinds of settings and applications as mentioned in the beginning of this paper, by generalizing the original Art Gallery Problem. At the same time, we want to overcome the drawback of the existing methods. Here we propose *Generalized Hybrid Evolutionary Coverage Solver* (*GHEC-Solver*) to achieve those purposes.

Main contributions of this work are three-fold. First, our approach serves a generalized and comprehensive framework for the Art Gallery Problem which binds a broad range of coverage applications. Second, we introduce the utilization of the topological features to fetch the potential guard candidates. Lastly, we also initiate combination of the *non-unicost* SCP and the *probabilistic evolutionary* optimization for obtaining the optimal set of guards. To the best of our knowledge, it is the first method pursuing such unified coverage problem, including the usage of evolutionary algorithm for solving the coverage problem.

1.2. **Text organization.** The rest of this paper is organized as follows. Several related works are exhibited in Section 2. A strategy for taking advantages from the topological features is presented in Section 3. Section 4 explains the optimization technique for different cases of the coverage problem. We then verify our proposed approach on various experiments in Section 5. Lastly, we give the conclusion and some possible future directions of this work.

2. Related Works.

The early result of the Art Gallery Problem was published by Chvatal [15] in which it is proclaimed that $\left(\frac{n}{3}\right)$ guards are adequate for covering any polygon with n vertices, which later was proved by Fisk [16]. Using Fisk's proof, Avis and Toussaint [17] then developed an $O(n \log n)$ algorithm for assigning the guard positions. These classical works assume a simple polygon without holes.

Recently, a vast effort has been promoted to deal with the Art Gallery Problem and coverage problem variants. Pinciu [18] proposed a coloring algorithm to find the connected guards in an art gallery. González-Banos and Latombe [5] presented an algorithm for a restricted version of the sensor placement problem (with a connection to the Art Gallery Problem) using a randomized approach, by applying a large set of guards and optimizing them using *hitting set* approach.

Later, heuristic-based approaches were then introduced. The authors in [19] adopted a greedy strategy to form a set of heuristic-based algorithm with the polygon partition methods. Bottino and Laurentini [14] then created a new heuristic for the Art Gallery Problem; however, the algorithm was restricted to cover only the edges of a polygon.

The most recent works made attempts for seeking an exact solution of the Art Gallery Problem via Integer Linear Programming (ILP) approach. Couto et al. [13] introduced an exact algorithm for the problem limited to the vertex guards. In [20], the authors utilized a finite set of so-called witnesses and guard candidates, and employed a linear relaxation of the primal-dual formulations iteratively to find the integer solution of the Art Gallery Problem. It was then improved in [21] by introducing the combination of Linear Programming (LP) and Difference of Convex (DC) programming. While both approaches performed good results in several instances, they failed to converge to the integer solutions in many other samples. Moreover, their approaches were basically restricted to the original issue of the Art Gallery Problem which did not take account of the guard positions.

In short, the existing algorithms for solving the Art Gallery Problem and sensor coverage above suffered from the following drawbacks:

a) They are applied to a simple or certain class of polygon, e.g., a simple polygon without holes or a convex polygon;
b) They are only applicable for limited problems, e.g., either a solution only for the vertex guards or just covering the edge of environment;
c) Lack real applications on the real environment conditions, since most of previous works are only a geometrical proof without any example on the real situations.

3. Generalized Hybrid Evolutionary Coverage Problem.

This section describes the framework of our approach for solving the coverage problem. The term "generalized" here is used for emphasizing our intention to develop a coverage algorithm which binds diverse applications, in contrast to the original Art Gallery Problem.

3.1. Algorithm overview.

Our proposed method consists of two large portions: extracting the location of the guard candidates, and optimizing them to obtain the optimal position of the guards. The first part involves the process of simplifying the given environment or map, from which we draw up a set of guard candidates. The topological features are then employed to guarantee the full coverage of the entire environment, by using the concept of visibility polygon.

The second part incorporates the optimization procedures of the obtained guard candidates. Here we propose the usage of a *non-unicost* Set Cover Problem alternated by a probabilistic evolutionary optimization technique, to ensure the optimality of the guards in accordance with the various problem settings (i.e., different applications of the coverage problem). Figure 1 shows the outline of our proposed algorithm.

3.2. Planar polygon extraction from the environment map.

Let $\mathcal{M} \subseteq \mathbb{R}^2$ denote a typical occupancy grid map representing the environment which is obtained either by a *Simultaneous Localization and Mapping* (SLAM) or a semantic map labeling algorithm.

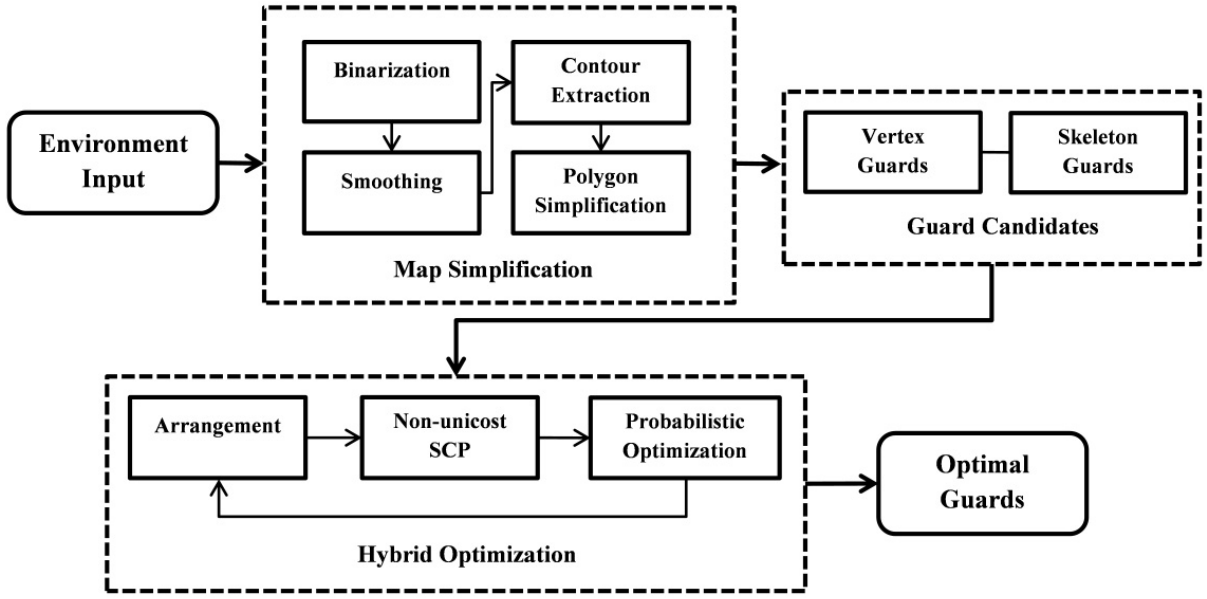

FIGURE 1. Block diagram of the proposed algorithm for the coverage problem

The map $\mathcal{M}$ is decomposed into two separable area, $\mathcal{M} = \{\mathcal{M}_w, \mathcal{M}_g\}$. Intuitively, $\mathcal{M}_w$ can be perceived as the "wall" area, i.e., the area which is *impenetrable* (in sensor terms) or *non-passable* (in robot terms), while $\mathcal{M}_g$ has the opposite meaning.

As the environment map tends to have a complex shape, our objective is to relax the map $\mathcal{M}$ into a simpler 2D planar polygonal form. We implement a set of the image processing procedures for simplifying $\mathcal{M}$ (the nearly similar procedures can also be found in [22]), as follows.

a) **Binarization**. Each point $m \in \mathcal{M}$ is mapped to a binary map $\mathcal{I}(m)$ as follows

$$\mathcal{I}(m) = \begin{cases} 1 & \text{for } \forall m \in \mathcal{M}_g \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

b) **Smoothing**. We perform a morphological operation using *opening* and *closing* operators for reducing noises in the map.

c) **Contour extraction**. An algorithm introduced by Suzuki and Abe [23] is then employed for extracting the contour from the binary map $\mathcal{I}(m)$. It yields an outer contour $\oint B$ and (possibly) k-inner contours $\oint B_k^H$, and both are a set of closed segment chains.

d) **Line segments simplification**. Lastly, we use Douglas-Peucker algorithm [24] for simplifying the contours $\oint B$ and $\oint B_k^H$. The output is a closed, connected polygon P with the outer boundary δP and k-inner boundaries δH_k where k denotes the number of holes inside P. Since H_k can be seen as the polygonal holes inside P, a point p is said to be the *interior point* of P if it satisfies

$$\left\{ p \in \left\{ P \cap \neg \left(\bigcup^{k} H_k \right) \right\} \right\}. \tag{2}$$

For the sake of simplicity, from now Equation (2) is written by $p \in P$ to describe the interior point p.

Figure 2 shows the example of simplified map created from the original 3D environment. The white area in Figure 2(c) represents the interior of the polygon from which we want to obtain the guards.

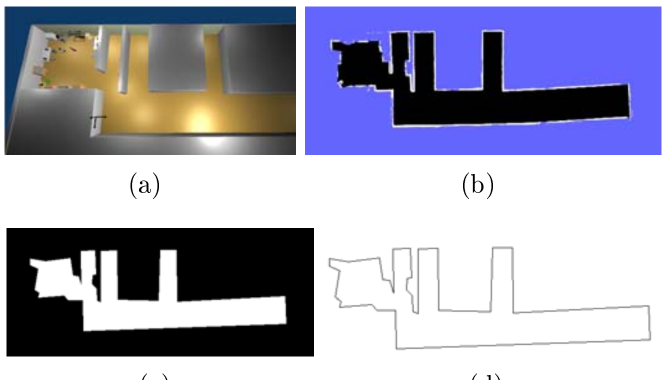

FIGURE 2. Simplifying environment map: (a) original environment, (b) map from SLAM algorithm, (c) binarized map, and (d) extracted polygon

3.3. **The concept of visibility polygon.** In the coverage problem, calculating the visibility polygon from a point (see Figure 3) becomes the most fundamental problem.

Definition 3.1. *Given a point $p \in P$, another point $q \in P$ is considered visible from p if $\overline{pq} \subset P$, where $\overline{pq}$ is a line segment.*

This basic notion leads to the establishment of the visibility polygon $\mathcal{V}(p)$, which is defined as

$$\mathcal{V}(p) = \{\forall q \in P \mid \overline{pq} \subset P\}. \tag{3}$$

Here we follow the work of [25] which utilizes a *triangular expansion algorithm* for implementing our visibility polygon. The interested readers are encouraged to directly refer to the original paper [25].

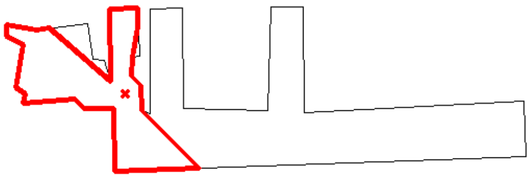

FIGURE 3. Visibility polygon of a point. The bold line segments are the visibility polygon of the point (bold cross).

3.4. **Guard candidates from the ensemble features.** The term "guard" is defined as a location on which we set an entity for "seeing" or covering the area in the environment (in this case, the polygon). The physical form of the entity can be a sensor, camera, base station, or even a robot. Intuitively, a person will put the guard at the position which has a wide view, or at the location which is difficult to see. For instance, a security camera is usually placed on the top corner of a room, or, if we have an omnidirectional camera, we will place it on the middle of ceiling which has maximum field-of-view (e.g., at the intersection of corridors).

For solving the coverage problem, we first extract the possible location of the guards using the human intuition as mentioned above. Here the topological features of the environment are adopted and selected as the candidates, as follows.

3.4.1. *Vertex guards.* The polygon vertices are among the important features in a polygon. Several previous researches on the Art Gallery Problem also employed these features, such as [14,18,26]. In reality, a vertex of a polygon represents a corner of a room on which the sensor or camera is placed. To obtain the vertex guards, we simply take out all vertices of the polygon's boundary, including its holes (if any).

3.4.2. *Skeleton vertices as the interior guards.* In the real problem setting such as a building, we can naturally determine the place in which we will get a wider view. Intuitively, a person will say that an intersection of corridors in a building grants wider view, compared with the wall or the corner of the room. Based on this reason, we deem it is necessary for us to take account of the topological shape of the environment for the coverage problem. Thus, we use skeletonization technique for capturing topology of the polygon.

For obtaining the skeleton vertices, we first build a skeleton map using Laplacian of distance transform technique [22] (in contrast to the straight skeleton in [27]). We construct a distance transform map $\mathcal{D}$ (Figure 4(a)) given by

$$\mathcal{D}(p) = \begin{cases} \|p - p'\| & \text{for } p \in P,\ p' \in \{\delta P, \delta H\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where p' is the nearest non-passable point to p.

We then apply a Laplacian filter to $\mathcal{D}$, to get the skeleton map $\mathcal{K}$, denoted by

$$\mathcal{K}(p) = \frac{\partial^2 \mathcal{D}}{\partial p_x^2} + \frac{\partial^2 \mathcal{D}}{\partial p_y^2}, \tag{5}$$

where p_x and p_y respectively denote the x-axis and y-axis of the point p. $\mathcal{K}$ is then binarized by a threshold. The skeleton guards are then acquired by taking out all junctions and endpoints of the skeleton (Figure 4(b)). Both types of guards (vertices and skeleton) are then coalesced, producing a set of guard candidates $\mathcal{G}$.

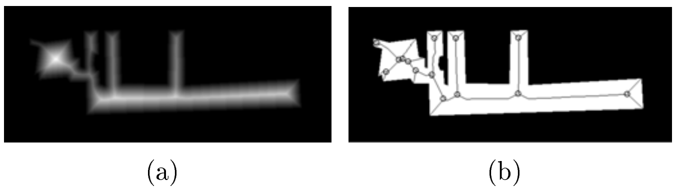

FIGURE 4. Skeleton guards extraction: (a) distance transform of the map, (b) obtaining skeleton vertices. The circles denote the skeleton guards.

3.4.3. *Coverage guarantee.* Before an optimization process is carried out for the set of guard candidates above, we want to show that the mixture of the guard candidates itself has already been able to cover the entire polygon, even it is not the optimal one. In other words, this property is giving a clue to the optimization part that it should always return a solution (i.e., full coverage of the polygon). The following proposition is used for exhibiting the coverage guarantee of the guard candidates.

Proposition 3.1. *All vertices (including the holes, if any) of a planar polygon are always adequate for covering the entire polygon.*

Proof: One of possible ways to prove this proposition is by using the set theory over the established theorems. For the polygon with holes, O'Rourke [2] stated $\frac{n+2h}{3}$ **vertex guards** are sufficient for covering a polygon with n vertices and h holes (see Theorem 5.1 of [2]). Let $\mathcal{G}_{vt}$ be a set of all vertices (including the holes) of a polygon with cardinality n, and $\mathcal{G}_{rk}$ be the set of covering vertices in O'Rourke theorem with cardinality $\frac{n+2h}{3}$. We begin with proving the set cardinality, $\frac{n+2h}{3} \leq n$. The problem can be written as

$$\begin{aligned} \frac{n+2h}{3} \leq n \Longleftrightarrow \frac{2h}{3} \leq \frac{2n}{3}, \\ \Longleftrightarrow h \leq n. \end{aligned} \tag{6}$$

- If $h = 0$, since $n \geq 3$ (a polygon is composed by at least three vertices), then $h \leq n$ is held.
- If $h > 0$, since $n > 3h > h$ (a hole has at least three vertices), then Equation (6) is held.

Thus, the inequality holds for any number of holes. Here we have proved that the statement $\frac{n+2h}{3} \leq n$ is true. As the consequence, $\mathcal{G}_{rk} \subset \mathcal{G}_{vt}$. It suggests there exists a set of element in $\mathcal{G}_{vt}$ which covers the entire polygon. Notice that for $h = 0$, the problem becomes the Chvatal theorem [15]. □

Consequently, the guard candidates $\mathcal{G}$ which is a combination of the vertices and skeleton guards, retains the same coverage guarantee.

Proposition 3.2. *The coverage of the combination of guard candidates $\mathcal{G}$ over the polygon P is always guaranteed.*

Proof: Let $\mathcal{G}_{vt}$ and $\mathcal{G}_{sk}$ respectively be the vertex and skeleton guards. Subsequently, the guard combination $\mathcal{G}$ can be denoted as $\mathcal{G} = \mathcal{G}_{vt} \bigcup \mathcal{G}_{sk}$. From Proposition 3.1, we know that $\forall g_{vt} \in \mathcal{G}_{vt}$ holds the coverage guarantee. Using the *set theory*, since $\mathcal{G}_{vt} \subset \mathcal{G}$, it implies $\forall g_{vt} \in \mathcal{G}$. Thus, $\mathcal{G}$ retains the same coverage guarantee as $\mathcal{G}_{vt}$. □

4. Guard Optimization with Hybrid Evolutionary Strategy.

After the guard candidates are determined, the next step is to optimize the number of guards for covering the entire area of P. Even for the original issue of the Art Gallery Problem which does not consider the placement of the guards; it is already an NP-hard problem [28]. Here we try to evade the problem by applying a *hybrid optimization* approach. First, we transform the coverage problem into a *Set Covering Problem*, a family of *Integer Linear Programming*. A heuristically *probabilistic evolutionary optimization* is then administered to obtain the optimal guard positions.

4.1. Arrangement of the guard's visibility.

Before we bring the coverage problem into a *Set Covering Problem*, we need to understand its prerequisite of the transformation, that is the *polygon arrangement*. We borrow the definition of the *arrangement* from [29]. Given a finite set of guard candidates $\mathcal{G}$, from which we acquire a set of visibility polygon $\mathcal{V}(\mathcal{G})$, the *arrangement* $\mathcal{A}(\mathcal{G})$ is then defined as the subdivision of the plane created by the intersection of all vertices of $\mathcal{V}(\mathcal{G})$, such that $\mathcal{A}(\mathcal{G}) : \mathcal{V}(\mathcal{G}) \mapsto \mathcal{F}_c$. Each subdivision area of $\mathcal{A}(\mathcal{G})$ is called a *face*, denoted by $f_c \in \mathcal{F}_c$. Figure 5 shows the definition of the arrangement and *face*.

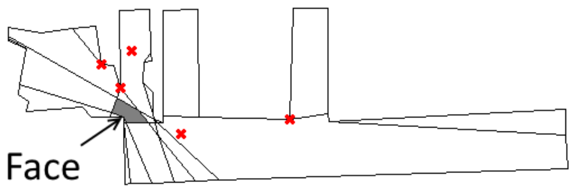

FIGURE 5. Arrangement of a set of guards. The bold crosses represent the guards. All edges are the result of combining the visibility polygon of all guards. The gray area is one of the *face* created by the arrangement.

Reciprocally, we can construct the visibility polygon of a guard $\mathcal{V}(g_1)$ as a set of *faces* f_c "seen" by the guard $g_1 \in \mathcal{G}$, such that

$$\mathcal{V}(g_1) \approx \left\{ \bigcup_{\forall f_c \in \mathcal{F}_{c1}} f_c | \mathcal{F}_{c1} \subset \mathcal{F}_c \right\}, \tag{7}$$

where

$$\mathcal{F}_{c1} = \{\forall f_c \in \mathcal{V}(g_1)\}. \tag{8}$$

It then raises a definition of the polygonal coverage, as follows.

Definition 4.1. *The coverage of a polygon P by a finite set of guards $\mathcal{G}$ is guaranteed under circumstances*

$$P = \bigcup_{\forall f_c \in \mathcal{F}_c} f_c, \tag{9}$$

where $\mathcal{F}_c$ are composed by the arrangement of $\mathcal{V}(\mathcal{G})$.

In other words, all *faces* will always cover the polygon as long as its composing set of guard $\mathcal{G}$ also has a full coverage. Now we aim for the optimal number of $\mathcal{G}$ which satisfies Definition 4.1.

4.2. **The Art Gallery Problem as the Set Covering Problem.** Equations (7) and (9) give two consequences:

a) There may exist a *face* "seen" by more than one guard, or geometrically

$$\mathcal{V}(g_1) \bigcap \mathcal{V}(g_2) \approx \mathcal{F}_{c1} \bigcap \mathcal{F}_{c2} \neq \emptyset, \quad \{g_1, g_2\} \in \mathcal{G}. \tag{10}$$

It simply means some guards may cover the same area.

b) Summation of all *faces* in $\mathcal{F}_c$ should cover the entire polygon P, in order to comply with Definition 4.1.

These consequences lead to the problem of assigning the smallest set of guards $\mathcal{G}$ such that its summation of *faces* in the $\mathcal{A}(\mathcal{G})$ satisfies Equation (9). Accordingly, we are able to bring the coverage problem into an *Integer Linear Programming* formulation, more precisely, a *Set Covering Problem.*

Given an $M \times N$ matrix $\mathbf{A}$, the *Set Covering Problem* (SCP) is defined as a problem of discovering a subset of the columns of $\mathbf{A}$ which covers all rows at a minimum cost [30]. Using the SCP formulation, the original Art Gallery Problem can be defined as follows

$$\text{Minimize} \quad \sum_{n=1}^{N} g_n, \quad \text{for } g_n \in \mathcal{G} \tag{11}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} a_{mn} g_n \geq 1, \quad \{m = 1, \ldots, M\} \tag{12}$$

$$g_n \in \{0, 1\}, \quad \{n = 1, \ldots, N\} \tag{13}$$

$$a_{mn} \in \{0, 1\}, \quad a_{mn} \in \mathbf{A}. \tag{14}$$

Here, M and N respectively denote the number of *faces* of the arrangement and guard candidates.

From the above minimization, the SCP formulation for the Art Gallery Problem is obtained by making the guards be the sets used for covering and imposing the *faces* of the arrangement as the elements to be covered. The guard placement inside the polygon is modeled by testing it using a binary condition ($g_n = 1$ if the guard is included into the set). Henceforth, a face $row(a_{mn})$ is set to 1 if it is seen by the guard g_n (see Equation (14)). Inequality of Equation (12) assures that a certain row must be covered by at least one column, or in other words, a *face* should be covered by at least one guard. It will guarantee that the entire polygon P is fully covered.

4.3. **Non-unicost Set Covering Problem.** According to Equation (11), each guard candidate is treated the same, i.e., it does not matter where the selected guards are chosen from, whether it lies at the interior or the vertices of the polygon, as long as it can cover the entire at the most minimum number. This is exactly what the original Art Gallery Problem aims for.

Such formulation of Equation (11) is often called *unicost* Set Covering Problem, indicating each guard is eligible to be chosen with the same cost. Nevertheless, in many cases of the coverage problem, some guards may become more alluring than the others. An interesting instance will be that a network provider should calculate the different land cost for placing each Base Tranceiver Station (BTS), while still covers the whole area.

In order to achieve a broad range of the coverage application, here we propose the usage of *non-unicost* Set Covering Problem. It allows us to assign different cost for each guard. Subsequently, we modify Equation (11) as follows

$$\text{Minimize} \qquad \sum^{N} c(g)g, \quad \text{for } g \in \mathcal{G}. \tag{15}$$

Equation (15) introduces a cost function $c(g)$, from which we can manage how important a guard will be. The formulation in Equation (11) becomes the special case of Equation (15) where all costs are equal. We will describe the cost function $c(g)$ further in the experimental section along with the examples, including how it will affect and alter the coverage problem.

4.4. **Hybrid evolutionary guard optimization.** The output of the *non-unicost* SCP is self-explanatory, i.e., it attains the optimal combination among the input set $\mathcal{G}$. However, it does not imply that the result is also the optimal one for the coverage of a polygon. This matter arises since there is no guarantee whether the optimal guards are already in the input set $\mathcal{G}$ or not; we only ensure the coverage as suggested by Proposition 3.2.

We then come up with a strategy to cope with it. Essentially, we break down the optimization process into two parts: *initial* and *probabilistic evolutionary* optimization. In the initial optimization, we reduce the guard candidates composed in 3.4 using the steps mentioned in Sections 4.1 to 4.3, yielding an *initial upper bound* of the guards. Afterward, we make attempt to reduce the guards further. Here we examine the area which has mutual guards coverage using a probabilistic evolutionary optimization, alternated by the *non-unicost* SCP. This technique is expected to cut down the mutual guards coverage by an alternative point guard.

4.4.1. *Initial optimization.* Let $\mathcal{G}_0$ be the initial set of guard candidates obtained in Section 3.4. We first optimize the guard candidates by

a) Constructing the arrangement $\mathcal{A}(\mathcal{G}_0)$;
b) Building the cost function of the guards $c(\mathcal{G}_0)$;
c) Solving the *non-unicost* SCP to get a set of *pre-optimized* guards $\mathcal{G}_{opt}$. The cardinality of $\mathcal{G}_{opt}$ is then called *initial upper bound*[1].

The *initial upper bound* $|\mathcal{G}_{opt}|$ of the possible optimal guards from the set is acquired using the above steps. To make it compact, the initial upper bound $|\mathcal{G}_{opt}|$ is now denoted by U. Since the algorithm is continued by an iterative optimization, the notation $\mathcal{G}_{opt}$ will be constantly used to show the set of the optimal guards found so far.

4.4.2. *Iterative probabilistic evolutionary optimization.* Our basic idea is to examine each optimized guard for further possible reduction, by analyzing the *faces* of its arrangement. Given a *face* $f_c \subset \mathcal{A}(\mathcal{G}_{opt})$ "seen" by a set of guards $\mathcal{G}_{par} \subset \mathcal{G}_{opt}$, the following definition is then applicable.

Definition 4.2. *For a face f_c seen by all $g \in \mathcal{G}_{par}$, it implies that all point guard $g \in \mathcal{G}_{par}$ are inside the visibility polygon of any point $p \in f_c$,*

$$\{\forall g \in \mathcal{G}_{par} | f_c \subset \mathcal{V}(g)\} \implies \{\forall p \in f_c | (\forall g \in \mathcal{G}_{par}) \in \mathcal{V}(p)\}. \tag{16}$$

The above definition is an extension of Definition 3.1, from which we want to show the possibility of "seeing" a set of guards by a point inside a *face*. It does not necessarily imply that all area of $\mathcal{V}(\mathcal{G}_{par})$ will be covered by a point $p \in f_c$, yet it exhibits the likelihood

[1]This naming convention suggests an attempt to decrease the cardinality, lower than this upper bound.

of "replacing" several guards to one point. Hence, Definition 4.2 becomes the stepping stone for our proposed approach for finding the optimal guards.

By making use of the definition above, here we build a *face cost map* which shows how the *faces* are parameterized by the guards, i.e., a *face* will have a higher value when it is covered by more number of guards. Let $h(f_c)$ be the *face cost map function* defined by

$$h(f_c) = |\mathcal{G}_{par}|, \quad \forall f_c \in \mathcal{F}_c. \tag{17}$$

Subsequently, we use $h(f_c)$ as a distribution function for sampling a set of *auxiliary guard candidates*. We currently sample the auxiliary guard candidates $\mathcal{G}_{sampling}$ from $\mathcal{F}_c$ using $h(f_c)$, as much as two times of $|\mathcal{G}_{opt}|$. It is expected that we will obtain more samples on the *face* covered by more guards. Figure 6 shows the definition of the *face cost map*.

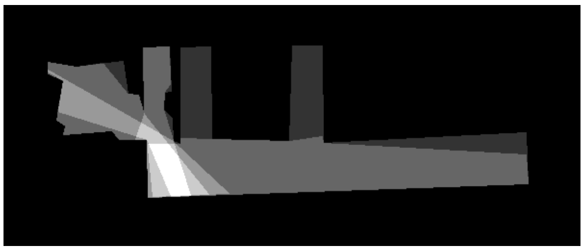

FIGURE 6. Face cost map of the arrangement in Figure 5. Brighter *face* means it is seen by more guards.

Both $\mathcal{G}_{opt}$ and $\mathcal{G}_{sampling}$ are concatenated forming a combined set of guards, from which the *non-unicost* SCP is then solved using the same steps as the one in the *initial optimization*, yielding a new set of optimized guards $\mathcal{G}_r$. The above processes of constructing the face cost map, sampling the *auxiliary guard* candidates, and solving the *non-unicost* SCP are accordingly alternated for the iterative optimization procedures.

A heuristic approach is then exerted for examining the optimality of the guards produced under current iteration. In principal, we wish the reduction of the cardinality (guard number), or a better aggregated guard cost $c(g_n)$ as demanded by Equation (15). We commence from the *non-unicost* SCP results (i.e., $\mathcal{G}_r$). There are three possible outputs which correspond to the cardinality of optimized guards $\mathcal{G}_r$ and its leverage to the next iteration.

a) $|\mathcal{G}_r| < U$. This is what we expect for, subsequently the next iteration will start using this new guard set $\mathcal{G}_r$ and the upper bound U is lowered to $|\mathcal{G}_r|$.
b) $|\mathcal{G}_r| > U$. It means the minimization in Equation (15) generates a set of guards which has lower aggregated cost, despite of having a higher cardinality. Please note that this type of output unlikely happens when the uniform cost function is used (e.g., the original AGP), considering Proposition 4.1 which will be presented later.
c) $|\mathcal{G}_r| = U$. It means $\mathcal{G}_r$ has a better aggregated cost with the same cardinality. We will particularly discuss this type of output later.

We empirically found that after several iterations, the last type of the output of the *non-unicost* SCP above appears in most of the cases. It suggests the cardinality of the guards for the coverage problem is gradually converged. Thus, a *Hausdorff metric* is adopted to ascertain the stopping condition of the iterative optimization process, defined as

$$\begin{aligned} D_{hd}(\mathcal{G}_r, \mathcal{G}_{opt}) &= \max\left\{d(\mathcal{G}_r, \mathcal{G}_{opt}), d(\mathcal{G}_{opt}, \mathcal{G}_r)\right\}, \\ \text{where } d(\mathcal{G}_r, \mathcal{G}_{opt}) &= \max_{g_r \in \mathcal{G}_r} \min_{g_{opt} \in \mathcal{G}_{opt}} \|g_r - g_{opt}\|. \end{aligned} \tag{18}$$

Here, Equation (18) has a physical meaning that is when the algorithm converges, there should not be much change on the position of the optimized guards *iteration-by-iteration.*

For the last two types of the output of the *non-unicost* SCP above, we merge the $\mathcal{G}_{opt}$ and $\mathcal{G}_r$ to be used in the next iteration. By this strategy, we basically feed the SCP solver all viable set of candidates found so far (i.e., has the same cardinality or cost) and let it discover the best one as the solution. It is expected to avoid the alternating result of the optimal guards[2] and speed up the optimization process.

Using this merging technique, we evoke the following proposition, to prove our statement in the second type of the output above.

Proposition 4.1. *For uniform cost function, the cardinality of subsequent optimal guards $|\mathcal{G}_r|$ in the iterative optimization should not exceed the initial upper bound U.*

Proof: (Proof by Contradiction) Assume $|\mathcal{G}_r| > U$ is true. For the iteration $i = 0$ (initial optimization), $\mathcal{G}_r$ is basically equal to $\mathcal{G}_{opt}$, so that $|\mathcal{G}_r|$ is equal to U. In the next optimization process, yielded $\mathcal{G}_r$ will be merged with $\mathcal{G}_{opt}$ and the algorithm uses $\mathcal{G}_{opt} \bigcup \mathcal{G}_{sampling}$ as the guard candidates $\mathcal{G}$, which is then utilized for solving the SCP (Equation (15)). It means $\mathcal{G}_r \subset \mathcal{G}$. Since Equation (15) is a minimization problem with uniform $c(g)$ and the previous $\mathcal{G}_r$ is in the set $\mathcal{G}$, the possible maximum cardinality must be U, by means of the Set Covering Problem. However, it contradicts the initial assumption. Therefore, we have to conclude that $|\mathcal{G}_r| \leq U$ for the subsequent iteration. $\square$

Someone may wonder that the metric in Equation (18) only considers the norm between two sets, regardless of the cost function $c(g)$. This matter is clarified by the following proposition.

Proposition 4.2. *Under all conditions, the iterative optimization in the probabilistic search always produces*

$$\sum_{g_r \in \mathcal{G}_r} c(g_r) \leq \sum_{g_r^- \in \mathcal{G}_r^-} c(g_r^-), \tag{19}$$

where $\mathcal{G}_r^-$ is the optimal solution in the previous iteration.

Proof: (Proof by Contradiction) The proof construction is basically the same with Proposition 4.1. Assume $\sum_{g_r \in \mathcal{G}_r} c(g_r) > \sum_{g_r^- \in \mathcal{G}_r^-} c(g_r^-)$ is true. In the iterative optimization, $\mathcal{G}_r^-$ is in the set of $\mathcal{G}$ used for minimization in Equation (15), since $\mathcal{G}_r^-$ is merged with $\mathcal{G}_{opt}$ and $\mathcal{G} = \mathcal{G}_{opt} \bigcup \mathcal{G}_{sampling}$. Again, it suggests the minimization result has the total objective value which is equal to at most $\sum_{g_r^- \in \mathcal{G}_r^-} c(g_r^-)$. Yet, it contradicts the initial assumption. Hence, we draw a conclusion that the proposition is true. $\square$

The above proposition ensures that our proposed *hybrid probabilistic guard optimization* is probabilistically converged towards the optimal solution. We then call off the iterative process when $D_{hd}(\mathcal{G}_r, \mathcal{G}_{opt})$ is below a threshold.

The probabilistic evolutionary optimization is summarized as follows

a) Initialize the cardinality of the given guard candidates;
b) Establish the *face cost map*;
c) Sample a set of auxiliary guard candidates using the *face cost map*;
d) Solve the *non-unicost* SCP;
e) Check the convergence using the *Hausdorff metric*, and repeat.

The complete process of our proposed optimization is shown by Algorithm 1.

[2]A condition where two sets of the optimal result show up alternately, which may create an infinite loop.

Algorithm 1 Probabilistic Evolutionary Guard Optimization

```
Require:
 1: $\mathcal{G}_0$: initial set of guard candidates
 2: $|\mathcal{G}_0|$: cardinality of $\mathcal{G}_0$
 3:
Ensure:
 4: $\mathcal{G}_{opt}$: optimized guards
 5:
 6: procedure ProbEvoGuardOpt($\mathcal{G}_0$)
 7:     // Initial optimization
 8:     $\mathcal{G}_{opt} \leftarrow$ Init($\mathcal{G}_0$)
 9:     $U \leftarrow |\mathcal{G}_{opt}|$                                  ▷ initial upper bound
10:     // The real loop starts here
11:     faceCostMap($\mathcal{G}_{opt}$)                                  ▷ Equation (17)
12:     $\mathcal{G}_{sampling} \leftarrow$ sample($\mathcal{G}_{opt}, 2|\mathcal{G}_{opt}|$)
13:     $\mathcal{G}_r \leftarrow$ solveSCP($\mathcal{G}_{opt} \bigcup \mathcal{G}_{sampling}$)          ▷ Equation (15)
14:     if $|\mathcal{G}_r| > U$ then                                   ▷ Proposition 4.1
15:         $\mathcal{G}_{opt} = \mathcal{G}_{opt} \bigcup \mathcal{G}_r$
16:         go to 12
17:     else if $|\mathcal{G}_r| < U$ then
18:         $\mathcal{G}_{opt} = \mathcal{G}_r$
19:         $U \leftarrow |\mathcal{G}_{opt}|$
20:         go to 11
21:     else
22:         if $D_{hd}(\mathcal{G}_r, \mathcal{G}_{opt}) >$ threshold then              ▷ Hausdorff
23:             $\mathcal{G}_{opt} = \mathcal{G}_{opt} \bigcup \mathcal{G}_r$
24:             go to 11
25:         else
26:             $\mathcal{G}_{opt} = \mathcal{G}_r$
27:             break
28:         end if
29:     end if
30:     return $\mathcal{G}_{opt}$
31: end procedure
```

4.5. **Remarks.** The reader may notice that the result of the *non-unicost* SCP in each iteration is examined by its cardinality. It then raises a question, there exist possibilities to have the $\mathcal{G}_r$ which has more guards than previous $\mathcal{G}_{opt}$, but with a smaller cost. In the real cases, the cost of a guard is often not so cheap (e.g., sophisticated camera, and robot). Therefore, we consider minimizing the number of guards as our priority.

Another notable thing is that the size of $\mathcal{G}_{opt}$ will grow due to the merging technique $\mathcal{G}_{opt} \bigcup \mathcal{G}_r$. Still, we have never empirically undergone any "bloated number of guard" problem, since the optimization process is done on a bounded area (polygon interior) with the *Hausdorff metric*. We can expect the distance between the optimal set on the current iteration and $\mathcal{G}_{opt}$ will gradually decrease.

5. **Experiments and Results.** The implementation of the GHEC-Solver is done on a Windows PC (i7 2.4 GHz, 16 GB RAM) using C++ programming language. We extensively utilize the Computational Geometry Algorithms Library (CGAL) [31] for

performing the visibility calculation, the arrangement building, and some other geometric computations. To solve the *non-unicost* Set Covering Problem, we use both open source (GLPK [32] and SCIP [33]) and commercial[3] (CPLEX [34] and Gurobi [35]) solvers.

5.1. **Experimental settings.** We want to evaluate the effectiveness and generality of our proposed algorithm. For accomplishing that goal, we prepare six different environments as shown by Figure 7, grouped into:

a) Artificial 2D/3D maps without and with holes (Figures 7(a), 7(b), 7(d), and 7(e));
b) A *star-shaped* synthetic polygon (Figure 7(c), which is also used in [5]);
c) A complex real building at our university (Figure 7(f)).

The environments represent distinct types of the coverage problem. The environment in Figures 7(a), 7(b), 7(d), and 7(e) elucidates the ability of the proposed method to handle the coverage of various classes of polygon, as well as how to transform the environment itself into a polygon. To be more specific, Figures 7(a) and 7(b) exemplify the coverage problem in the environment without holes, while Figures 7(d) and 7(e) show the opposite one.

The star-shaped polygon (Figure 7(c)) is the special case of the coverage, from which we want to exhibit how far the guard optimization can be carried out by our algorithm. Intuitively, a guard located exactly in the middle of the polygon should cover the entire environment.

Lastly, an attempt to solve the coverage of a real and complex building is demonstrated, showing the feasibility of our algorithm to be directly adopted in the real environment. Here we use a hall room inside our university (see Figure 7(f)). Table 1 shows the polygon complexity of each environment, where it varies from 24 to 102 vertices.

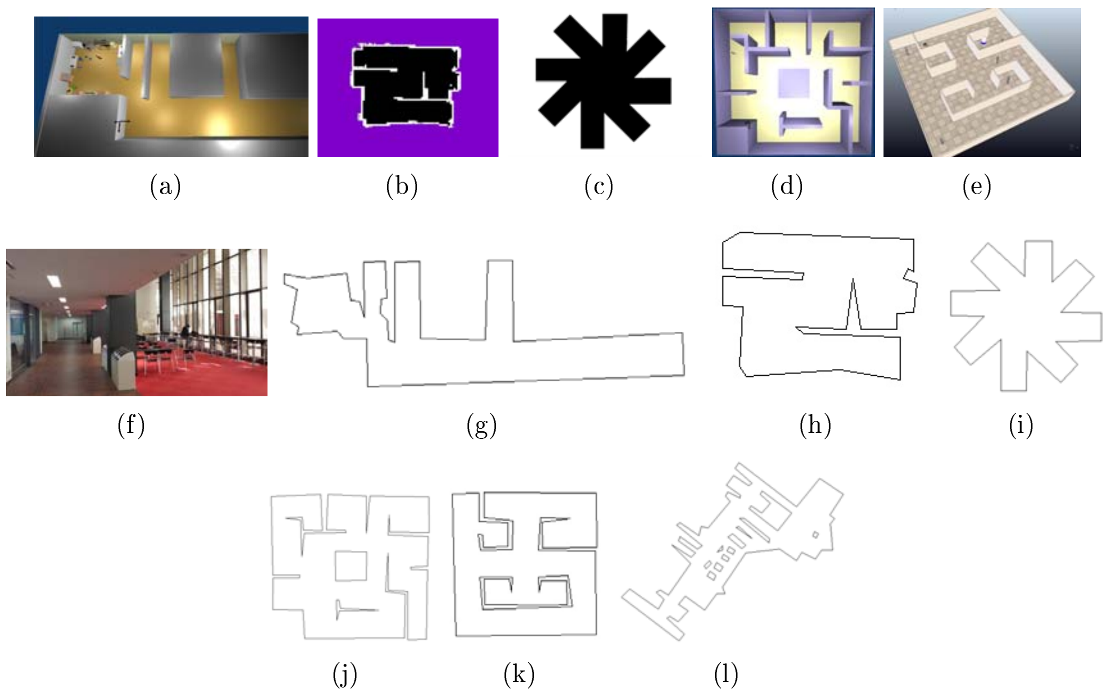

FIGURE 7. Environment map used for the coverage problem: (a)-(f) the real environment map used as input, (g)-(l) the simplified polygon of its respective map. The map-polygon pairs are (a)-(g), (b)-(h), (c)-(i), (d)-(j), (e)-(k), and (f)-(l).

[3]We use the academic version of CPLEX and Gurobi.

TABLE 1. Polygon complexity of the environment

Environment	Generic Map Name	# vertices	# holes
Figure 7(a)	A	32	0
Figure 7(b)	B	26	0
Figure 7(c)	C	24	0
Figure 7(d)	D	51	2
Figure 7(e)	E	38	1
Figure 7(f)	F	102	11

Generality of the proposed algorithm is then confirmed by experiments in the following sections, which represent different kinds of the coverage problems. We want to show that it can be achieved by simply changing the cost function $c(g)$.

5.2. **Art Gallery Problem.** In this experiment, we evaluate the capability of our proposed algorithm to solve the original issue of the Art Gallery Problem. The goal is obvious to minimize the number of guards without bothered by any guard preference and placement. It can be realized by making the cost function $c(g)$ in Equation (15) to be uniform.

Here we analyze the result of our approach applied into all environments mentioned in Section 5.1. First, we examine the guards quality generated by our algorithm. Table 2 represents the optimal guards produced by the initial and iterative optimization (as have been explained in Sections 4.4.1 and 4.4.2), and its computational time respectively. The computational time shown in the table refers to the method using SCIP [33] as the *non-unicost* SCP solver, which performs the best among all solvers. Further analysis about the SCP solvers will be clarified later. Thus, the optimal guard positions can be qualitatively perceived in Figures 8 and 10.

TABLE 2. Number of optimal guards for the Art Gallery Problem and its computational time

Map Name	# Guard Candidates	# Optimal Guards			
		Initial Optim.	Time (s)	Evolutionary Optim.	Time (s)
A	62	5	1.442	4	15.166
B	50	4	0.792	3	6.451
C	46	1	0.883	1	0.883
D	104	8	3.290	8	18.308
E	76	7	1.163	7	10.213
F	224	11	40.102	11	206.182

From Table 2, we observe that the initial optimization reduces the number of guards significantly with a fast computational time. It can even be remarked as “near optimal” compared with the final results. It clarifies the merit of the guard candidate features which assists the GHEC-Solver in converging quickly.

The iterative probabilistic evolutionary optimization also plays a good role. In maps A and B, the guards are further optimized to obtain the optimal solution. Figure 8(a) *vs* 10(a) and 8(b) *vs* 10(b) show the *head-to-head* comparison of the initial and iterative optimization. The other maps yield the same cardinality even after going through the iterative optimization process. An example of the evolution of the guard positions during

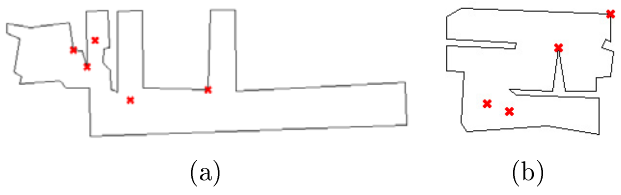

FIGURE 8. Initial optimization result of the Art Gallery Problem. The guard positions are marked by the bold crosses.

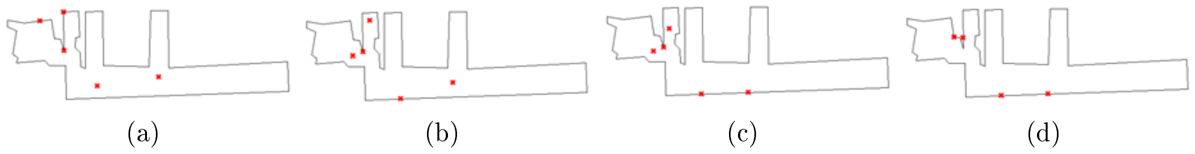

FIGURE 9. Evolution of the guards during the iterative optimization process for map A (from left to right)

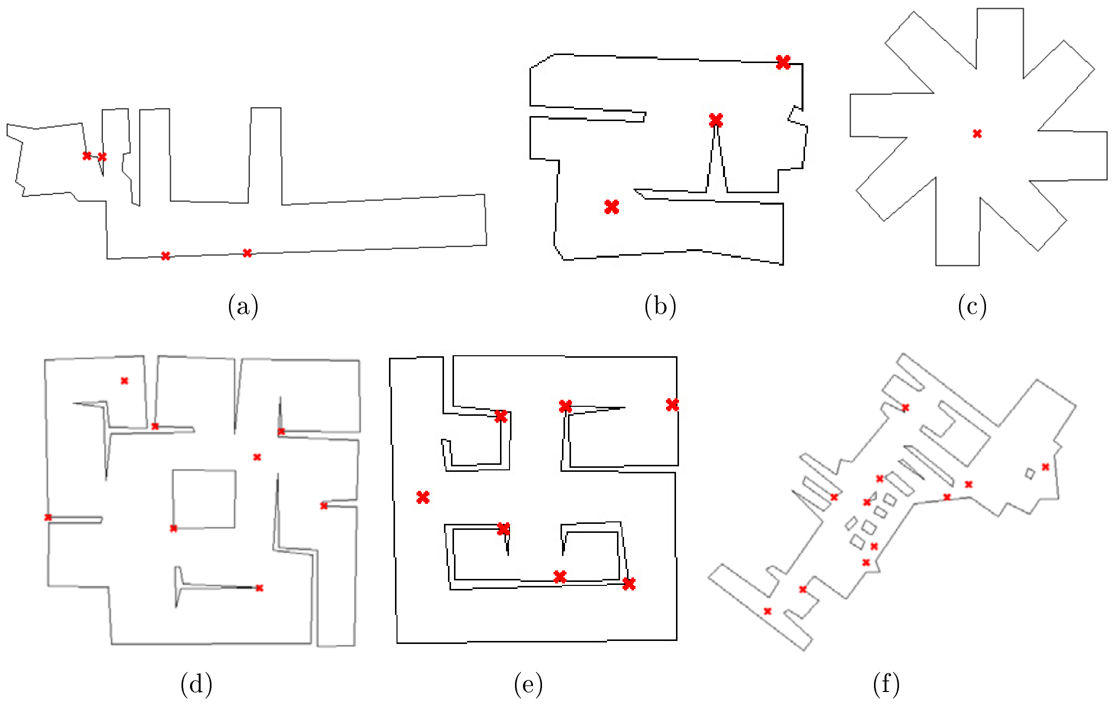

FIGURE 10. Optimal guards for the original Art Gallery Problem. The guard positions are marked by the bold crosses.

this iterative optimization is shown by Figure 9. Here we can make a suggestion to the user which concerns with the computational speed and is willing to get considerably "good" cardinality, employing the result from the initial optimization of the GHEC-Solver has already served the purpose.

5.3. **Coverage problem with guard preferences.** In this section, we investigate the performance of our algorithm when the guards are restricted based on the user preferences, for instance, it can only be placed at the vertices. It realizes the problem mentioned in the following example.

Example 5.1. Vertex only guards: *This type of problem resembles most of the existing Art Gallery Problem, such as [15,17,26]. Here we need to put the guard exactly on the vertex of a polygon. Following cost function is then used.*

$$c(g) = \begin{cases} 1, & \textit{for } g \in \textit{vertices}, \\ \infty, & \textit{otherwise}. \end{cases} \tag{20}$$

Actually, this problem can be simply achieved by removing all guard candidates which are not located at the vertices of the polygon. In our case, however, we want to exhibit how this problem can be solved by simply modifying the cost function, which shows the generality of the GHEC-Solver. Figure 11 and Table 3 demonstrate its optimization results.

We notice that the guard cardinality is higher than the one in the Art Gallery Problem. It is because we lost some information from the non-vertex guards which is restricted by

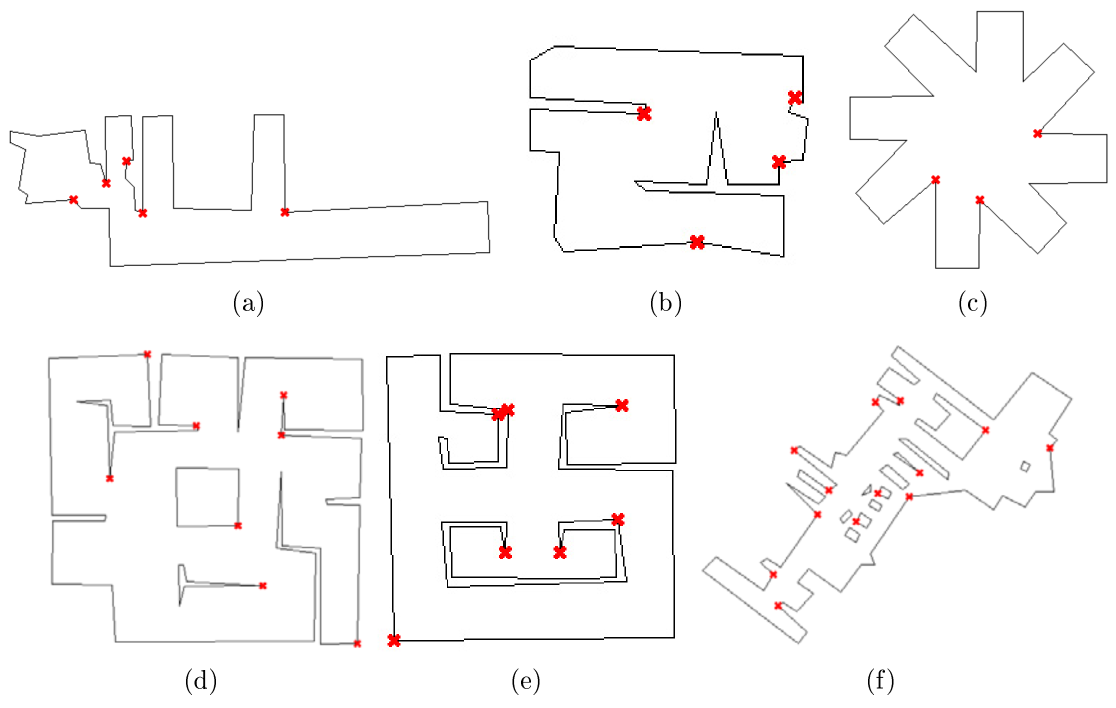

Figure 11. Optimal guards for the vertex coverage problem. The guard positions are marked by the bold crosses.

Table 3. Optimal guards for the coverage problem with vertex only guards

Map Name	# optimal guards			
	Initial Optimization	Time (s)	Evolutionary Optimization	Time (s)
A	5	1.520	5	11.600
B	4	0.703	4	5.139
C	3	0.770	3	6.770
D	8	3.590	8	11.891
E	7	1.025	7	7.780
F	13	35.400	13	100.632

the cost function. It also signifies how the topological features play a great role to render an optimal solution for the coverage problem. An obvious example is the result of map C (Figure 11(c)) where now it needs three vertex guards to cover the entire polygon, compared with one guard in the previous problem.

5.4. **Coverage problem with arbitrary cost function.** As the continuation of the above section, we now present the realization of the coverage problem with the arbitrary cost function, which becomes the main concern of the GHEC-Solver. We set up a case using a set of mobile robot formations to cover the environment, as follows.

Example 5.2. Robot guards: *Here we need to consider the physical limitation of the mobile robot to do the coverage tasks. One example of this limitation is that the robot is supposed to be located not too close with the walls, to avoid a collision. We then accommodate this problem using a distance function as follows*

$$c(g) = e^{-\|g-\delta P\|}, \tag{21}$$

where δP represents the boundary of the polygon (in this case, the walls). Equation (21) tells us that we prefer to locate the robot far from the walls.

Table 4 and Figure 12 show the optimization results of the coverage problem using the cost function in Equation (21). We can observe that there is no guard which lies at the vertices nor the boundary of the polygon, as suggested by the usage of the distance function. Most of the guards are located at the skeleton or the middle area of the polygon. Physically, it means the guard robots take place in the middle of room or the intersection of corridors, which is favorable according to the collision properties mentioned in above example.

Table 4. Optimal guards for the coverage problem with arbitrary cost function

Map Name	# optimal guards			
	Initial Optimization	Time (s)	Evolutionary Optimization	Time (s)
A	7	2.881	6	21.859
B	6	1.471	5	10.747
C	1	1.024	1	12.488
D	10	10.849	9	29.538
E	7	1.820	7	13.417
F	14	73.668	13	212.976

One alluring result is that this coverage problem produces relatively higher cardinality of the optimal guards than the one in the Art Gallery Problem. This matter is plausible since the cost function in Equation (21) leads to a lower total cost as pointed by Equation (15), even though it has more number of guards.

5.5. **Computation time.** Now we evaluate the computation time needed for each section of the proposed algorithm, as well as the influence of using different solvers for *the non-unicost* SCP. Most of the time are allocated for establishing the arrangement and *face*, including the visibility polygon calculation. In average, it takes around 74.25% of the total time using the best performing SCP solver. The map simplification and generating the guard candidates spend around 17.45%, while the non-unicost SCP only takes 8.3% of the total time.

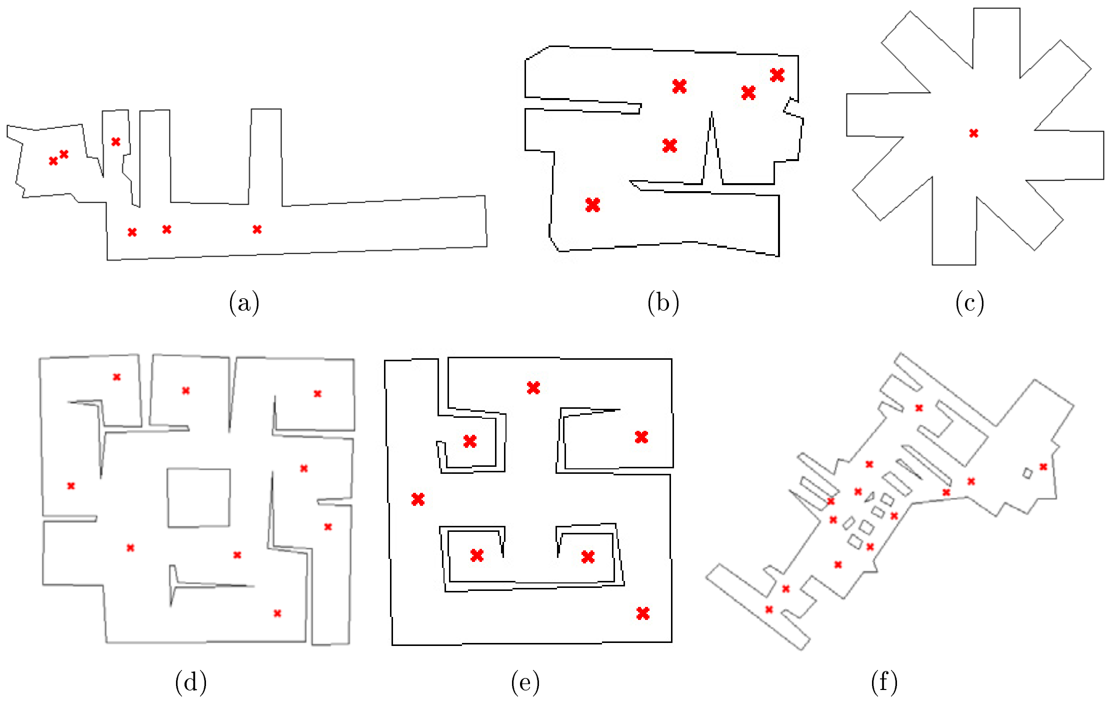

Figure 12. Optimal guards for the coverage problem using arbitrary cost map function. The guard positions are marked by the bold crosses.

Table 5. Calculation time of different SCP solvers

Map Name	Solving time (seconds)			
	GLPK	SCIP	GUROBI	CPLEX
A	0.35	**0.059**	**0.059**	0.09
B	0.12	**0.03**	**0.03**	0.059
C	0.04	0.04	**0.02**	0.052
D	4.099	**0.232**	0.32	0.708
E	0.089	**0.03**	**0.03**	0.11
F	N/A	**10.342**	13.667	19.417

As a side result, we also report the effect of using different SCP solvers. Table 5 shows the calculation time needed for each SCP solver in the initial optimization stage. We select the initial optimization for comparing the performance of each solver because the huge number of guard candidates to be optimized are lied on it. As implied by Table 5, the commercial solvers like CPLEX [34] and GUROBI [35] are superior to the open source solver such as GLPK [32]. For the map F, GPLK fails to retrieve the optimal solution. Surprisingly, SCIP [33] which is an open source solver has a comparable, or even better speed than the commercial one. Due to this reason, all results of the coverage problem mentioned in the previous section are accomplished using the SCIP.

6. Conclusion.

We have presented the generalized framework for solving the coverage problem. The algorithm utilizes the geometric topology features for determining the guard candidates. A combination of the *non-unicost* SCP and the *probabilistic evolutionary optimization* is then performed to obtain the optimal combination of the guards. We also have shown that several classes of the Art Gallery and coverage problem can be solved by one proposed framework.

While our algorithm shows remarkable results, there are some open problems which can direct the future of this research. One of them is to include the *field-of-view* (FOV) limitation of the guards into the system. This constraint is very interesting since it will broaden the type of guards which can be used in the framework, e.g., we can use any sensor or camera widely sold in the market for setting up the surveillance system. Another possible direction is to find an exact formulation and solution for this generalized Art Gallery Problem.

REFERENCES

[1] M. Ernestus, S. Friedrichs, M. Hemmer, J. Kokemller, A. Krller, M. Moeini and C. Schmidt, Algorithms for art gallery illumination, *Journal of Global Optimization*, pp.1-23, 2016.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[3] S. L. Devadoss and J. O'Rourke, *Discrete and Computational Geometry*, Princeton University Press, 2011.

[4] J. Urrutia, Art gallery and illumination problems, in *Handbook of Computational Geometry*, J. Sack and J. Urrutia (eds.), Elsevier, 2000.

[5] H. González-Banos and J. C. Latombe, A randomized art-gallery algorithm for sensor placement, *Proc. of the 17th Annual Symposium on Computational Geometry*, pp.232-240, 2001.

[6] J. Ivanchev, H. Aydt and A. Knoll, Information maximizing optimal sensor placement robust against variations of traffic demand based on importance of nodes, *IEEE Trans. Intelligent Transportation Systems*, vol.17, no.3, pp.714-725, 2016.

[7] B. Li and A. D. Kiureghian, Robust optimal sensor placement for operational modal analysis based on maximum expected utility, *Mechanical Systems and Signal Processing*, vol.75, pp.155-175, 2016.

[8] V. Ramaswamy and J. R. Marden, A sensor coverage game with improved efficiency guarantees, *2016 American Control Conference (ACC)*, pp.6399-6404, 2016.

[9] P. K. Agarwal, E. Ezra and S. K. Ganjugunte, Efficient sensor placement for surveillance problems, *Proc. of the 5th IEEE Int. Conf. on Distributed Computing in Sensor Systems*, pp.301-314, 2009.

[10] A. A. Ismail, S. Herdjunanto and P. Priyatmadi, Ant system algorithm for finding optimal path on travelling salesman problem with path restriction, *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol.1, no.3, pp.43-48, 2012.

[11] B. Wang, Coverage problems in sensor networks: A survey, *ACM Comput. Surv.*, vol.43, no.4, pp.1-53, 2011.

[12] K. J. Obermeyer, A. Ganguli and F. Bullo, Multi-agent deployment for visibility coverage in polygonal environments with holes, *International Journal on Robust and Nonlinear Control*, vol.21, no.12, pp.1467-1492, 2011.

[13] M. C. Couto, P. J. de Rezende and C. C. de Souza, An exact algorithm for minimizing vertex guards on art galleries, *International Transactions in Operational Research*, vol.18, no.4, pp.425-448, 2011.

[14] A. Bottino and A. Laurentini, A nearly optimal algorithm for covering the interior of an art gallery, *Pattern Recognition*, vol.44, no.5, pp.1048-1056, 2011.

[15] V. Chvatal, A combinatorial theorem in plane geometry, *Journal of Combinatorial Theory, Series B*, vol.18, no.1, pp.39-41, 1975.

[16] S. Fisk, A short proof of chvtal's watchman theorem, *Journal of Combinatorial Theory, Series B*, vol.24, no.3, pp.374-375, 1978.

[17] D. Avis and G. Toussaint, An efficient algorithm for decomposing a polygon into star-shaped polygons, *Pattern Recognition*, vol.13, no.6, pp.395-398, 1981.

[18] V. Pinciu, A coloring algorithm for finding connected guards in art galleries, *Discrete Mathematics and Theoretical Computer Science, ser. Lecture Notes in Computer Science*, vol.2731, pp.257-264, 2003.

[19] Y. Amit, J. S. B. Mitchell and E. Packer, Locating guards for visibility coverage of polygons, *Proc. of the 9th Workshop on Algorithm Engineering and Experiments*, pp.120-134, 2007.

[20] A. Kröller, T. Baumgartner, S. P. Fekete and C. Schmidt, Exact solutions and bounds for general art gallery problems, *J. Exp. Algorithmics*, vol.17, 2012.

[21] A. Kröller, M. Moeini and C. Schmidt, A novel efficient approach for solving the art gallery problem, *ALCOM: Algorithms and Computation, ser. Lecture Notes in Computer Science*, vol.7748, pp.5-16, 2013.

[22] I. Ardiyanto and J. Miura, Visibility-based viewpoint planning for guard robot using skeletonization and geodesic motion model, *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, pp.652-658, 2013.
[23] S. Suzuki and K. Abe, Topological structural analysis of digitized binary images by border following, *Computer Vision, Graphics, and Image Processing*, vol.30, no.1, pp.32-46, 1985.
[24] D. H. Douglas and T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol.10, no.2, pp.112-122, 1973.
[25] F. Bungiu, M. Hemmer, J. Hershberger, K. Huang and A. Kröller, Efficient computation of visibility polygons, *Proc. of the 30th European Workshop on Computational Geometry*, 2014.
[26] S. K. Ghosh, Approximation algorithms for art gallery problems, *Canadian Information Processing Society Congress*, pp.429-434, 1987.
[27] O. Aichholzer, F. Aurenhammer, D. Alberts and B. Grtner, A novel type of skeleton for polygons, *The Journal of Universal Computer Science*, vol.1, no.12, pp.752-761, 1995.
[28] D. Lee and A. Lin, Computational complexity of art gallery problems, *IEEE Trans. Information Theory*, vol.32, no.2, pp.276-282, 1986.
[29] E. Fogel, D. Halperin and R. Wein, *CGAL Arrangements and Their Applications – A Step-by-Step Guide*, in ser. Geometry and Computing, 2012.
[30] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, New York, 2011.
[31] CGAL, *Computational Geometry Algorithms Library*, Version 4.3, http://www.cgal.org.
[32] GLPK, *Gnu Linear Programming Kit (GLPK)*, Version 4.34, http://www.gnu.org/software/glpk /glpk.html.
[33] T. Achterberg, Scip: Solving constraint integer programs, *Mathematical Programming Computation*, vol.1, no.1, pp.1-41, 2009.
[34] CPLEX, *Cplex Optimizer*, Version 12.6, http://www-01.ibm.com/software/commerce/optimization/ cplex-optimizer/index.html.
[35] GUROBI, *Gurobi Optimizer*, Version 5.6.2, http://www.gurobi.com/.