Sensor Deployment Using Billiards Algorithm

Bilal Al-Qudah  
Dept. of Computer Science & Engineering  
Southern Methodist University  
Dallas, TX, 75275, USA  
972-437-6928  
balqudah@smu.edu

Rabie A. Ramadan  
Department of Systems and  
Computer Engineering Al-Azhar University  
Cairo, Egypt.  
+201-6-764-3227  
raramada@engr.smu.edu

Abstract

This paper proposes a Billiards algorithm for two sensor deployment problems, random and deterministic deployment. The deployment considers both homogenous and heterogeneous sensors. The algorithm aims to maximize the coverage of a given monitored field with obstacles. The coverage is maximized by avoiding sensors overlapping and minimizing the uncovered areas. By adjusting the expansion ratio, the algorithm was able to find the best sensing ranges that maximize the field coverage. The conducted experiments point out the effect of expansion ratio, iterations between expansions, number of collisions, and mobility, on the overall coverage. At the same time, Billiards algorithm shows significant improvement in the coverage performance from the initial field’s coverage.

1. Introduction

Sensor deployment is a challenging task due to sensors limitations and hostility nature of the monitored fields. Sensors usually suffer from limited energy, storage, and low processing capabilities. In spite of their limitations, sensors could be classified onto two types, homogeneous and heterogeneous. Heterogeneous sensors in this context mean that sensors differ in their initial energy, adjustable sensing ranges, and mobility capabilities while homogeneous sensors share the same characteristics. In addition, there are two types of the monitored fields which are hostile and non-hostile fields. In the latter, the access to the monitored field is usually granted and sensors can be deployed manually or using mobile robots. On the other hand, in a hostile field, sensors have to be “sprayed” using flying robots, helicopters and/or unmanned vehicles [8][9]. In such case, optimizing the initial placement of the sensors will certainly enhance the coverage and prolong sensor network lifetime.

Due to the limited space, we review only the most related deployment algorithms to our work. The most related deployment algorithms are those proposed by Zou et al. [13], Xue et al. [11], and Lam et al. [7]. Zou et al. in [13] proposes a virtual force (VF) algorithm for the deployment of mobile sensors. Sensors are assumed to have two types of forces, attractive and repulsive. Based on these forces, the algorithm virtually moves the sensors from one place to another. After a number of iterations, sensors will be rearranged to maximize the coverage taking into consideration sensors energy. Xue et al. in [11] proposed an improved VF algorithm named “virtual force directed co-evolutionary particle swarm optimization (VFCPSO)”. The algorithm combines swarm optimization and virtual force for mobile and stationary sensors as well. Both VF and VFCPSO showed reasonable enhancement in the coverage over the initial deployment. However, they are proposed only for homogeneous sensors; in addition, sensors energy limitation is considered after the termination of the algorithms. In other words, after the best final position of the sensors are identified, the algorithm does one of the following to satisfy sensors energy limitations: 1) sensors are allowed to move only a distance that is equivalent to their deployment energy, 2) the algorithm searches the sensors coverage history and assigns them to the best positions within their maximum capabilities. In both cases, this restriction is expected to affect the overall coverage of the sensor network.

Lam et al. in [7], propose a deployment algorithm for homogenous and heterogeneous sensors, respectively, using circle packing. The algorithm is mainly based on the tangency among the sensors and adjusting sensing ranges of the internal sensors iteratively until their angles sums approaches $2\pi$. The algorithm shows significant enhancement in the final coverage over the initial coverage. However, it assumes sensors with enough energy to perform all of the required computations as well as the movement which is not the case in all mobile sensors. Sensors usually suffer from limited energy and they are expected to consume only a small percentage of their energy during the deployment process. In addition, the algorithm assumes the mobility of all the deployed sensors;
while in a real sensor network, a combination between mobile and stationary sensors could be required due to the cost of the mobile sensors.

In this paper, we study the deployment of homogenous and heterogeneous sensors in a field with obstacles. We consider different characteristics and capabilities of the sensors such as their adjustable sensing ranges, mobility, and initial energy during the deployment process. Our proposed algorithm tries to utilize mobile sensors for the benefit of the field coverage. At the same time, it increases the lifetime of the sensors by using only their best sensing ranges. The best sensing range of a sensor is a percentage from its maximum range that maximizes the coverage. In addition, two types of sensor deployment are considered, deterministic and random deployment. Our target is to maximize the coverage of the monitored field without overlapping among sensors areas. We propose using billiards algorithm that adapts the concepts of billiards physics and collisions laws. An extensive set of experiments are conducted to show the performance of the proposed algorithm.

The paper is organized as follows: in the next section, the two deployment problems are formally defined, in section 3, principles of the billiards algorithm are explained, Sensor Billiards Algorithm (SBA) is detailed in section 4, section 5 studies the performance of the proposed algorithm, finally, the paper concludes in section 6.

2. Problem Statement

In this section, we define two deployment problems. The first problem considers the deterministic deployment of the sensors on a field with obstacles. The access to the field is assumed granted and/or the number of sensors is relatively small. The second problem handles the deployment of sensors on a field with obstacles while the monitored field is assumed hostile and/or the number of sensors is relatively large. In both problems, homogenous and heterogeneous sensors are considered.

2.1. Deterministic Deployment Problem

\[ D \text{ is a sensor deployment to } |S| \text{ number of sensors in a monitored field } F, \text{ where sensors' sensing ranges } r \in \{r_1, r_2, r_3, \ldots, r_{|S|}\}, D((x_1, y_1), \ldots, (x_{|S|}, y_{|S|})) \]

\[ \sqrt{(x_i, y_j)^2} \geq (r_i + r_j) \in [0, F], 1 \leq i, j \leq |S|, \] and \( x_j, y_i \in F \). Also, a number of obstacles \(|O| \) are distributed in \( F \). The objective is to deploy \(|S|\) where \(|S| \geq 2\) in the field \( F \) without overlapping and minimize \( F \) uncovered area.

Given a number of sensors \(|S|\) to be deployed in a field \( F \). Sensors sensing range is represented by a disk (circle) with radius \( r_i \), where \( i \) is the sensor’s identifier. The deployment field \( F \) is defined by a length \( l \) and a width \( w \). A set of obstacles \( O \) are assumed to be distributed in \( F \). In addition, sensors could be homogenous or heterogeneous; homogenous sensors sensing ranges are equal \((r_i = r_j)\), while heterogeneous sensors sensing ranges may differ. The final placement of a sensor \( i \) at a position \( P(x_i, y_i) \) should maximize the coverage of the field. The coverage is maximized by preventing the overlapping areas among the sensors and minimizing the uncovered areas.

2.2. Random Deployment Problem

Assuming \( D \) is a sensor deployment in a monitored field \( F \) with sensors’ sensing ranges \( r \in \{r_1, r_2, r_3, \ldots, r_{|S|}\} \), initial positions \( p \in \{p_1, p_2, p_3, \ldots, p_{|S|}\} \), initial energy \( E \in \{E_1, E_2, E_3, \ldots, E_{|S|}\} \), speed \( v \in \{v_1, v_2, v_3, \ldots, v_{|S|}\} \), and the allowed consumed energy during the deployment process \( e \in \{e_1, e_2, e_3, \ldots, e_{|S|}\} \), where

\[ D((x_1, y_1), \ldots, (x_{|S|}, y_{|S|})) \]

\[ \sqrt{(x_i, y_j)^2} \geq (r_i + r_j) \in [0, F], 1 \leq i, j \leq |S| \]. Also, a number of obstacles \(|O| \) are placed in the field \( F \). The objective is to deploy the number of sensors \(|S|\) where \(|S| \geq 2\) in the field \( F \) without overlapping and with minimizing \( F \) uncovered area.

In random deployment, the given number of heterogeneous sensors \(|S|\) is initially deployed in a field \( F \). Their initial positions are defined by \( x_i \) and \( y_i \) coordinates within the field borders. In addition, the initial set of energy \( E \) of all sensors and the allowed deployment energy set \( e \) is given. The allowed deployment energy \( e \) is defined as a percentage of the sensors initial energy \( E \). Moreover, sensors could be mobile or stationary; a mobile sensor \( i \) can move from one place to another with speed \( v_i \) while a stationary sensor speed is assumed zero.
Nevertheless, some of the obstacles $|O|$ might be scattered in the field. Sensors are not allowed to be deployed at the same place of an obstacle. The objectives of this deployment are similar to the deterministic deployment objectives. However, the given constraints such as the mobility and energy are considered during the deployment process.

3. Principles of Billiards Algorithm

The fundamental idea of the billiards algorithm is based on the collision between two or more moving objects with given velocities. According to the mechanical laws, the collided objects repel each other with different velocities. The collisions of $n$ objects can simply be determined by the following equation:

$$m_i x_i(t) = C(t, x_i(t), x_j(t)) \Delta x_j(t) = q_j, x_i(0) = p_i, i = 1, n$$

Where $x_i \in R^d$ (d > 1) and $m_i$ are the position and mass of object $I$, respectively. $C(t, x, v)$ represents the object external force located at $x$ with velocity $v$ and time $t$. The collision between any two objects $i$ and $j$ at time $t_c$ is subject to $||x_i(t_c) - x_j(t_c)|| = r_i + r_j$, where $r_i$ and $r_j$ are the radii of objects $i$ and $j$, respectively.

In billiards algorithm, objects are assumed to move in straight lines after the collision. This assumption is only used to simplify the collision detection mechanism. It also allows computing the collision time $t_c$ before the actual collision occurs. For instance, if we assume the positions of $i$ and $j$ at time $t$ is given by

$$x_i(t) = q_i + v_i t \text{ and } x_j(t) = q_j + v_j t$$

where the objects positions are $q_i, q_j \in R^d$, and their velocities at time $t_0$ are $v_i, v_j \in R^d$. By taking the square of both sides of equation (1), the next collision time $t_c$ can be computed from the following equation:

$$||A||^2 r_i^2 + 2(A, v) + ||A||^2 = \sigma^2$$

where $A = v_i - v_j, A = q_i - q_j$, and $\sigma = r_i + r_j$.

The idea of using Billiards algorithm in scientific applications is not new. It has been used in many scientific applications such as fluid suspension [10] and object packings [1]. However, to our knowledge, this is the first work that considers the billiards concept for sensor deployment. Our deployment algorithm is inspired from the object packings applications, namely circle packing in the field of computational geometry. The algorithm is used to pack equal and non-equal size circles in various shapes. In the following section, we show how the billiards algorithm can be adapted to solve the deployment problem.

4. Sensor Billiards Algorithm (SBA)

In this section, we describe the details of the adapted version of billiards algorithm for sensor deployment. First, we assume that a centralized node runs the billiards algorithm; then, it informs each node by its final position. In case of deterministic deployment, the centralized node could be a dedicated or non-dedicated machine while in random deployment it has to be a dedicated machine since the nodes are already deployed. Each sensor sends its coordinates, sensing range, and the amount of energy that it is willing to spend in the deployment process to the centralized node via one or more hop. We assume a hierarchal network with cluster heads communicating directly to the centralized node.

Using billiards algorithm, each sensor is considered as an object with a maximum radius equal to the sensor’s sensing range. The object mass and velocity are generated as a percentage of its radius. The number of objects in such case is equal to the number of sensors plus the number of obstacles in the field. Obstacles are considered as stationary objects with velocity equal to zero.

In the deterministic deployment, objects initial positions are generated based on a uniform distribution function taking into consideration the field’s borders $[0,F]$. To avoid the collision among objects in their initial states, their radii are set to zero. Also, all objects are assumed mobile but the obstacles and there is no restriction on their movement. Objects radii are iteratively expanded by a percentage of their maximum radii. After each expansion, they are allowed to collide with each other, the obstacles, and the field’s borders for a number of iterations. After certain number of expansions, objects will be “jammed” and their movement will be almost zero.

As shown in Algorithm 1, the algorithm starts by initializing all of the required parameters including the stopping conditions. The algorithm terminates when the maximum radii of all objects are reached, objects are jammed and no significant movement is detected, and/or the required coverage is satisfied. In step 2, the objects radii are expanded gradually by $\Delta_i = r_i + \sigma$, where $\sigma$ is the given percentage. The
Collision among the objects, obstacles, and the borders are detected in steps 2.3.1 and 2.3.2. These steps run for \( I \) number of iterations; then the stopping criteria is checked in Step 3. If at least one of the stopping criteria is met, the algorithm terminates and objects positions as well as the best object radii are reported.

In the random deployment, sensors are already deployed in the monitored field and their positions are assumed to be known to a centralized node. Again, sensors are represented by objects within the borders of the field and their radii initially are assumed zero to avoid the collision. Objects radii are expanded iteratively and the objects are allowed to collide for \( I \) number of iterations. However, the movement of the mobile objects is restricted by the allowed deployment energy \( e_i \). For instance, when a collision occurs between two objects \( i \) and \( j \), both objects change their velocities and directions. If \( i \) reach its maximum energy (distance), the algorithm assumes a virtual obstacle in its path and treats such cases as a new collision. This process continues until the algorithm becomes stable and no significant movement is detected. The random deployment algorithm is shown in Algorithm 2; it is similar to the algorithm presented in Algorithm 1 with adding the energy limitations check in steps 2.3 and 2.4.

**Algorithm 1:**

**Step 1: Initialization**

1. Distribute the given objects in the field and determine each object coordinates \( x_i \) and \( y_i \).
2. Set \( r_i = 0 \) \( \forall i \in |S| \) , where \(|S|\) is the number of objects.
3. Define the obstacles positions \( O_{(j,x,y)} \) and \( O_{(j,y)} \) \( \forall j \in [0,F] \).
4. Divide the field into \( N \) number of sectors.
5. Set the stopping criteria
   - All objects reached their maximum radii
   - Objects are jammed
   - The required coverage percentage is reached.

**Step 2: Expansion**

1. Expand the radius of each object by a percentage \( \Delta_i = r_i \star \delta \), where \( \delta \) is a given percentage.
2. **While** (iterations < \( I \))
   1. Check collision with neighbors
   2. Get nearest collision in time.
   3. iterations++
   End While

**Step 3: Stopping Criteria**

If the stop condition is met go to step 3.2 otherwise go to step 2.

Terminate and report objects final positions and their radii.

**Algorithm 2:**

**Step 1: Initialization**

1. Determine objects coordinates \( x_i \) and \( y_i \).
2. Set \( r_i = 0 \) \( \forall i \in |S| \) , where \(|S|\) a set of sensors.
3. Define the obstacles positions \( O_{(j,x,y)} \) and \( O_{(j,x)} \) \( \forall j \in [0,F] \).
4. Divide the field into \( N \) number of sectors.
5. Counter = 0
6. Set the stopping criteria
   - All objects reached their maximum radii
   - Objects are jammed
   - The required coverage percentage is reached.

**Step 2: Expansion**

1. Expand the radius of each object by a percentage \( \Delta_i = r_i \star \delta \), where \( \delta \) is a given percentage.
2. **While** (iterations < \( I \))
   1. Check collision with neighbors
   2. Get nearest collision in time.
3. **IF** ( \( (d_i \geq \text{Max}(r_i)) \) AND (Counter != \( I \)) ) \( \forall i \in |S| \)
   THEN
   4. Add virtual object.
   5. ELSE if (Counter < \( I \))
   **ELSE** go to step 2.2
   6. Update Objects
   7. go to step 3.

**Step 3: Stopping Criteria**

If the stop condition is met go to step 3.2 otherwise go to step 2.

Terminate and report objects final positions and their radii.

Collision Detection, as shown in the Algorithms 1 and 2, is the core of both the deterministic and the random billiards algorithms. In fact, it defines the computational complexity of the billiards algorithm. In this paper, we adapt the collision detection mechanism introduced by Lubachevsk in [6]. The main idea of this mechanism is to simulate the collisions using event-driven instead of time-driven approach. In event-driven, objects states are updated only when a collision occurs. In addition, the deployment field is divided into \( N \) sectors and each object checks the collision with only its neighbors in that sector. In our implementation, this division occurs only at the initialization step of the billiard algorithm as in 1.4 in Algorithms 1 and 2. A double buffer technique is also implemented to keep track of objects current and next state. The algorithm chooses the nearest collision(s) in time and advances the objects based on this event. The computational complexity of the billiards algorithm is shown in Algorithm 2.
cost of the collision detection step as given in [6] is \( O(\log N) \) which is repeated for \( I \) number of iterations per expansion.

5. Experimental Results

In this section, a set of experiments are conducted to show the performance of the sensor billiards algorithm (SBA). A common monitored field \( F \) of 400m width and 400m length is used throughout these experiments. Obstacles are distributed based on a uniform distribution function within the borders of the field.

5.1. Deployment Examples

A set of experiments are conducted to show some of the deployment examples using billiards algorithm. In the first example, shown in Figure 1, 28 homogenous sensors are randomly deployed in the field \( F \). The maximum sensing range of each sensor is set to 100m. The algorithm is configured to run for 100 iterations between each expansion and the expansion ratio is set to 1% of the sensors maximum sensing range. As can be seen, the algorithm shows a successful deployment to the sensors without overlapping. It also terminated when the field is jammed as well as reported the final sensing range (80m per sensor). In addition, Figure 1 shows different arrangement to the sensors. For instance, in Figure 1a and b, sensors are arranged in rows with 5 sensors in each row but 4 sensors in the second and the third row from the top, respectively. In Figure 1c, sensors are arranged in columns with 5 sensors in each column but 4 sensors in the second column from the left. The algorithm stabilized after 2639, 2437, and 2301 collisions to reach the arrangements in Figures 1a, 1b, and 1c, respectively.

The second example shown in Figure 2 illustrates the deployment of 34 heterogeneous sensors in the same field. Sensors are deployed randomly and their initial positions are identified. Other parameters such as sensing range, energy, speed, and mass are also generated randomly based on a uniform distribution function. After 3250 collisions, sensors are jammed and the algorithm terminated with the deployment schema shown in Figure 2a. In this case, 83% of the field area is covered. Another example is explored where 33 sensors are deployed randomly at the field \( F \) along with one obstacle. The obstacle is represented, for simplicity, by a stationary sensor marked as a black circle in Figure 2b. The algorithm terminates after 3755 collisions with 78% coverage of the field’s area.

5.2. Effect of the Expansion Ratio on the Overall Coverage

The previous examples show successful deployment schemes of heterogeneous as well as homogenous sensors. However, heterogeneous deployment as shown in Figures 2 indicates the importance of the expansion ratio where some of the sensors are forced to go beyond the deployment field borders. In such case, the overall coverage of the monitored field might be affected. Therefore, in this section, we study the effect of the expansion ratio on the coverage percentage. Different expansion ratios, as given in Figure 3, range from 1% to 50% of the sensors maximum sensing range are selected. As shown in the figure, the average results over a set of experiments with different problem settings demonstrate that increasing the expansion ratio with small percentage produces high coverage. For instance, at 1% expansion, the average coverage is almost 83% of the deployment field while at 20% expansion; the average coverage is 74%. At the same time, the results show that using more than 20% expansion ratio does not affect the coverage percentage.

Based on our observations, using small expansion ratio, the resulted collisions due to expansion process is less than those when large ratio (i.e. 30%) is used. This guarantees a better object rearrangement. The new arrangement postpones sensors jamming from occurring early; therefore the resulted coverage will be higher since the gaps is covered.
5.3. Effect of the Number of Iterations Between Expansions on the Coverage

In the previous set of experiments, the expansion ratio seems to have a significant effect on the overall coverage. However, the number of iterations between expansions may also affect the coverage where sensors rearranged after each iteration. In this section, a large number of experiments are conducted to study this effect. The expansion ratio is set to 1% of the sensors sensing ranges while other parameters are randomly generated. The number of iterations per expansion is varied from 50 to 500 iterations. Sensors are also varied from 100 sensors to 500 sensors.

Figure 4 shows the average results of these experiments. It shows that in most cases, as the number of iterations increases, the coverage percentage increases. For example, the coverage percentage reaches 75% when 50 iterations per expansion is used while it is 85% when 450 iterations per expansion is used. However, in some cases, the increase in the number of iteration results in insignificant increase in the coverage percentage. For instance, 3% is the difference in the coverage percentage between using 300 and 500 iterations per expansion.

5.4. Effect of the Sensors Energy on the Coverage

In this section, we study the affect of sensors deployment energy on the overall coverage. Figure 7 shows the relationship between the sensors allowed deployment energy and the coverage percentage of the monitored field. A set of 200 sensors are chosen to be deployed randomly in the monitored field. Sensors energy is assumed fixed for all sensors and converted into a percentage of the field distance. For instance, if the allowed deployment energy is 10%, the sensor is allowed to move in a circle from its initial position unless it collides with a border or another sensor. As shown in the figure, increasing the deployment energy increases the field coverage. However, increasing the energy to 50% results a significant coverage percentage. After 50% of the energy, the change in the coverage percentage is a minor. Therefore, we recommend using only energy that allows sensors to move up to 50% of the monitored field width and/or height.

5.6. Effect of Mobility on the Coverage

In this set of experiments, 500 sensors are used to study the affect of mobility on the overall coverage. A combination of stationary and mobile sensors are deployment in the monitored field $F$ while the number of mobile sensors are controlled. The number of the mobile sensors vary from 10% to 100% of the total number of sensors. As shown in figure 10, when the number of the mobile sensors is 10% of the total number of sensors, the average coverage is almost 60%. While increasing the mobile sensors percentage to 50% leads to 80% coverage. Increasing the mobility percentage beyond the 50% does not have much affect on the coverage. Therfore, based on these results, using the number of mobile sensors percentage up to 60% of the total number of sensors is recommended.

6. Conclusion

In this paper we proposed a Billiards algorithm for sensor deployment. We studied both random and deterministic deployment, where the sensors can be homogeneous or heterogeneous. In addition, hostile and non-hostile fields have been taken into consideration. The Billiards algorithm contemplated both mobile stationary sensor deployments where sensors simulated as Billiards objects with ability to expand. The next challenge is to provide a comprehensive set of key factors that can generate optimal deployment schemes. Our future work will also include the deployment of connected sensors while maximizing the coverage.
Figure 7: Effect of sensor deployment energy on the coverage percentage

Figure 8: Effect of sensors speed on the coverage percentage

Figure 10: Effect of mobility on the coverage percentage

7. References


