A Distributed Coverage Optimization Approach for Mobile Sensor Networks

Jalal Habibi, Hamid Mahboubi and Amir. G. Aghdam

Abstract

A distributed Voronoi-based sensor deployment approach is proposed to optimize the sensor network coverage. It is assumed that each sensor can construct its Voronoi polygon using the information it receives from the neighboring sensors. To increase the local coverage of the sensors, a point is obtained in each polygon, in such a way that if the corresponding sensor is placed there, then its covered area within the polygon is maximized. A nonlinear optimization algorithm is proposed based on the gradient projection to find the underlying optimal point. The algorithm can be implemented in a distributed fashion with minimum communication among the sensors. Examples are provided to demonstrate the effectiveness of the proposed approach in terms of convergence rate, coverage performance, and energy consumption.

I. INTRODUCTION

Wireless sensor networks have attracted considerable attention in recent years, as a means to gather information efficiently in a wide variety of applications. Such applications range from biotechnology to rescue missions to target tracking and surveillance [1], [2], [3], [4], [5]. Researchers in diverse disciplines have made significant contributions to the field by developing mathematical models for the operation of the system [6], designing cost-effective resource management techniques [7], and deriving efficient deployment algorithms to increase network coverage [8], [9].

Network coverage is an application of special interest, where it is desired to deploy a group of sensors to achieve maximum coverage of an environment. In this type of system, there is often no a priori knowledge of the target environment, and also the initial positions of the sensors in the field are typically unknown [10]. On the other hand, due to the distributed nature of the network, it is more desirable to have a decentralized decision-making structure for the whole network, as opposed to a centralized one. Each sensor in this type of network has limited communication and sensing ranges, and cannot communicate with a central server in general [11].

In [12], location services (which are concerned with obtaining the location information of the destination) for mobile ad-hoc networks are discussed. A new coverage model, namely, surface coverage, is proposed in [13]. Then, two important problems regarding (i) expected coverage ratio with stochastic deployment, and (ii) optimal deployment strategy with planned deployment are studied. Furthermore, three approximation algorithms with provable approximation ratios are introduced in [13]. The basic protocol approach is proposed in [14], where the sensors find their final destination using an iterative procedure. An alternative method, namely virtual movement protocol, is proposed in [9] which does not require the sensors to move physically unless the communication cost is too high, or the final destinations are determined. In both of the above techniques, the coverage holes are found using the Voronoi diagram [15], [16]. In [9], three distributed self-deployment algorithms are proposed to determine the final destination of the sensors: (i) VEC (vector-based algorithm), (ii) VOR (Voronoi-based algorithm), and (iii) Minimax. In the VOR algorithm, the distance of each sensor from the vertices of its corresponding Voronoi polygon is obtained, and the desired location for the sensor to move to is calculated accordingly. VEC, on the other hand, is a "proactive" algorithm which aims at properly relocating the sensors to achieve even distribution in the field. In the Minimax algorithm, each sensor moves (more smoothly compared to other algorithms) to a point inside its corresponding Voronoi polygon such that its maximum distance from the vertices of the polygon is minimized. While the above techniques prove effective in network coverage, they suffer from a number of shortcomings. For example, in the VOR and Minimax approaches if a sensor is located close to a narrow edge in its Voronoi polygon, it may not be required to move far enough to increase network coverage properly. Furthermore, network coverage achieved by the VEC algorithm may not be satisfactory compared to other methods, when there is a relatively large number of sensors in the network.

A distributed Voronoi-based coverage maximization strategy is proposed in the present article which is entitled as Max-area or MaxS. The main contribution of the proposed algorithm is that in order to maximize the total coverage, each sensor at each round maximizes its own coverage in its Voronoi cell. Finding the optimum point inside each polygon is a complicated task and is solved here using some approaches form nonlinear optimization theory.

The proposed algorithm here also provides the solution to an important facility location problem which deals with locating inside a geometric shape a disk which maximize the intersected area of these two. This problem has been studied in many articles but to the best knowledge of authors no optimization-based approach is proposed yet to solve this problem.

The paper is organized as follows. Some preliminaries and problem statement is outlined in the next section. As a main element of proposed coverage optimization algorithm, the solution to so-called sensor location problem is presented in

This work has been supported by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN-262127-07. J. Habibi, H. Mahboubi, A. G. Aghdam are with the Department of Electrical and Computer Engineering, Concordia University, 1455 de Maisonneuve Blvd. W., EV005.139, Montréal, Québec, H3G 1M8, CANADA. {jalal,h_mahbo,agham}@ece.concordia.ca
section III. The overall multi-sensor coverage optimization algorithm is described in section IV along with presentation of simulation results. Section V concludes the paper.

II. PRELIMINARIES

We assume that the target field is a flat two-dimensional space with known boundaries and contains no obstacle and at the initial point the sensors are randomly located in the field. We denote the target field by \( \mathcal{F} \) and the sensors by \( S_i (i \in \{1, \ldots, n\}) \) with equal sensing range of \( R_s \) and communication range of \( R_{com} \). A general point in \( \mathbb{R}^2 \) space is shown by \( q = [q_1, q_2]^T \). The position of the sensor \( S_i \) is denoted by \( x_i \), with the corresponding sensing disk of \( D(x_i) \).

The Voronoi polygon (or cell) of the sensor \( S_i \) is defined as \( P_i = \{ q \in \mathbb{R}^2 : d(q,S_i) \leq d(q,S_j), \forall j \in \{1, \ldots, n\}, j \neq i \} \) where \( d(\cdot, \cdot) \) denotes the euclidean distance of two points. The set of \( n \) Voronoi polygons \( \{P_i \mid i = 1, \ldots, n\} \) which is named the Voronoi diagram of the metric space partition the target field in a systematic manner with many nice geometric properties.

The concept of Voronoi diagram is widely used in computational geometry and many applied science problems [17]. Voronoi-based decomposition of the target field in sensor networks area, is a way to assign to each sensor a part of the field for service delivery. The important point about Voronoi polygons is that each sensor, having its own position and the position of its neighbors, can build-up its own Voronoi polygon. This means that any deployment strategy based on Voronoi diagram can be implemented in a distributed manner by agents with limited information about the whole network.

Distributed Voronoi-based coverage optimization approaches follow similar steps which can be described in an abstract simplified form as follows:

**General Voronoi-based Coverage Optimization Algorithm**

1. Based on the received information from neighboring sensors, each sensor \( S_i \) builds up its own Voronoi polygon \( P_i \).
2. Each sensor calculated the covered area by itself in its own Voronoi polygon.
3. Based on a point selection strategy, each sensor finds a candidate point inside its own polygon as its next position.
4. Each sensor calculates its covered area in case it moves to the new position. If this expected value of covered area would be more than the current covered area by a specified factor (like 1 %), the sensor moves to the new point. If not, it will stay at its current position.
5. If any of the sensors moves to a new point, the algorithm will be repeated from step 1. If not, the sensors have already reach a stable points and the movement will be terminated.

It has been shown that the total coverage of mobile sensors converges to a final value assuming the general locally increasing behavior of mobile sensors [9]. Note that we assumed that the movement of sensors in step 4 is performed in a synchronized manner, which means that the next round of movement is distinguishable for all sensors. This can be done using one of the existing time synchronization strategies for sensor networks ([18], [19], [20]).

Different coverage optimization methods in the above-mentioned framework differ mainly in their point selection strategy. The overall performance of coverage in the sensor network is affected by the strategy utilized by individual sensors. The main contribution of this paper is to propose a point selection strategy in which each sensor maximizes its own coverage in its Voronoi polygon. The next section describes how the optimum point can be found inside each polygon.

III. THE OPTIMUM SENSOR LOCATION (OSL) PROBLEM

As the main element of Max-area (MaxS) algorithm, the problem of optimum sensor location is discussed in this section. Consider a sensor \( S \) located at \( x_s \) has disk-shaped sensing pattern \( D(x_s) \) with radius \( R_s \). Assume that a density function \( \phi(q) \) is defined over a convex polygon \( P \) as the target region. The density function shows the relative importance of the points inside \( P \) and is considered as a weight function over \( P \).

The OSL problem looks for an optimum point inside \( P \) with the property that if the sensor moves to that point the weighted covered area of it inside \( P \) would be maximized. This problem can be defined as following optimization problem:

\[
x^*_s = \arg \max_{x_s \in P} F(x_s) = \arg \max_{x_s \in P} \int_{P \cap D(x_s)} \phi(q)dq
\]

The relevant geometry of the sensor location problem is shown in Fig. 1 where the goal is to maximize the hatched region by optimally locating the center of the disk inside the polygon.

This problem is a nonlinear optimization problem with linear constraints. Note that it is very hard to express the objective function as an explicit function of the elements of \( x_s \) as decision variables. Even the computation of the objective function for a specific value of \( x_s \) requires double integration over a convex region which is simple but time-consuming. Also implementation of the proposed algorithm by the distributed mobile sensing agents requires many considerations. The reason is that these agents have usually low computational capabilities and movement strategy for them has to be made as simple as possible. Detailed mathematical approach presented here is designed carefully by consideration of the requirement for the mobile computing agents.
A similar facility location problem is of great interest in the context of locational optimization theory where a geographical area is described by a polygon and service range of a facility by a circle. The density function might represent the population density inside the area and the problem is to find a place inside the region to locate a facility with maximum achievable benefits. A famous example is finding the optimum point for a shop to sell most [21]. Although many articles are written on this problem, to the best of authors’ knowledge, the exact solution has not yet been reported. A nonlinear optimization approach to solve this problem is provided in this work.

It can be observed that the solution to the sensor location problem is obvious for some extreme cases. Let the center and radius of the smallest enclosing disk of a polygon be denoted by $x_{enc}$ and $R_{enc}$ respectively. As an example, consider the polygon in Fig. 2, with the corresponding smallest enclosing circle. For the case when the sensing range is greater than or equal to $R_{enc}$, the solution to the OSL problem is simply the point $x_{enc}$. As it can be seen from this figure, in this particular case the sensor can cover the whole polygon from this point.

The center of the largest inscribed disk inside a polygon is known as the Chebychev center of the polygon and can be computed efficiently, e.g., by solving a simple linear programming problem [22]. Let the center and radius of this circle (also shown in Fig. 2), be denoted by $x_{cheb}$ and $R_{cheb}$, respectively. Obviously, $x_{cheb}$ is the solution to the OSL problem if the density function is uniform over the polygon and the sensing range is less than or equal to $R_{cheb}$. However, for the cases where $R_{cheb} < R_s < R_{enc}$, or where the density function is not uniform, the solution is not straightforward, and a method is proposed in the next section to tackle this problem.

IV. NONLINEAR OPTIMIZATION APPROACH

The approach presented in this section is a gradient-based optimization algorithm which finds a suitable movement direction $p_k$ in an iterative fashion to increase the objective function in the OSL problem, based on its gradient with respect to the sensor position $x_s$. Given the direction $p_k$, a procedure called line search determines the optimal step size for maximizing the objective function in that direction. The optimization variable is updated in each iteration as follows:

$$x_{k+1} = x_k + \alpha_k p_k$$

The value of $\alpha_k$ is obtained from the line search procedure as noted above. It is discussed later that for the most efficient solution, the movement of the sensor should not be exactly in the direction of the gradient of the objective function; instead, it should be a scaled version of it. Different elements of this optimization strategy are described in detail in the following subsections.
A. Computation of the Gradient Vector

The gradient of the objective function is computed in the sequel using the results of [23]. Consider a region defined by the intersection of \( k \) inequalities \( h_i(x_s, q) \leq 0 \) for \( i = 1, \ldots, k \). By concatenation of the boundary functions \( h_i(x_s, q) \) as \( h(x_s, q) = [h_1(x_s, q), \ldots, h_k(x_s, q)]^T \), this region can be represented by \( h(x_s, q) \leq 0 \). Define also \( \mu(x_s) = \{ q \in \mathbb{R}^n : h(x_s, q) \leq 0 \} \), and denote the boundary of this set by \( \partial \mu(x_s) \). The boundary of the region corresponds to the equalities in the above formulation. Note that this boundary has \( k \) segments, where each segment can be expressed as:

\[
\partial \mu(x_s) = \mu(x_s) \cap \{ q \in \mathbb{R}^n : h_i(x_s, q) = 0 \}
\]

A generic integral function over the region is considered as follows:

\[
F(x_s) = \int_{h(x_s, q) \leq 0} p(x_s, q) dq
\]

The gradient of \( F(x_s) \) with respect to \( x_s \) can be computed as:

\[
\nabla_{x_s} F(x_s) = \int_{\mu(x_s)} \nabla_{x_s} p(x_s, q) dq - \sum_{i=1}^{k} \int_{\partial \mu_i(x_s)} \frac{p(x_s, q)}{\nabla h_i(x_s, q)} \nabla_{x_s} h_i(x_s, q) dL
\]

For the sensor location problem, it is desired to find the gradient with the integrand \( p(x_s, q) = \phi(q) \). For a polygon with \( m \) facets, the corresponding region is described by a set of \( m \) linear inequalities as \( Hq - K \leq 0 \), where \( H_{m \times 2} \) and \( K_{m \times 1} \) are matrices with real entries. The sensing disk centered at \( x_s \) can also be expressed as \( ||q - x_s||^2 - R^2_s \leq 0 \), or equivalently as \( (q - x_s)^T(q - x_s) - R^2_s \leq 0 \).

With this formulation, one can set \( h_i(x_s, q) = H_i q - K_i \) for \( i = 1, \ldots, m \) (where the \( H_i \) and \( K_i \) denote the \( i \)-th row of the matrices \( H \) and \( K \) respectively) and \( h_{m+1}(x_s, q) = (q - x_s)^T(q - x_s) - R^2_s \).

Now, one can find the gradient of the objective function from equation (5). Since the density function is independent of \( x_s \), thus \( \nabla_{x_s} \phi(q) = 0 \), and hence the first term in the right side of (5) vanishes. Also, the first \( m \) functions defining the region \( (h_i(x_s, q) \text{ for } i = 1, \ldots, m) \) do not depend on \( x_s \), which means that \( \nabla_{x_s} h_i(x_s, q) = 0 \) for \( i = 1, \ldots, m \). The gradients of \( h_{m+1}(x_s, q) \) is computed as follows:

\[
\nabla_{x_s} h_{m+1}(x_s, q) = -2(q - x_s)
\]

\[
\nabla q h_{m+1}(x_s, q) = 2(q - x_s)
\]

Therefore, the gradient of \( F(x_s) \) with respect to the position of the sensor is given by:

\[
\nabla_{x_s} F(x_s) = \int_{\partial_{m+1} \mu(x_s)} \frac{q - x_s}{||q - x_s||} \phi(q) dL
\]

where \( \partial_{m+1} \mu(x_s) \) is the portion of the perimeter of the sensing disk which is inside the polygon \( P \). Note that the (8) is, in fact, the integral of a vector normal to the perimeter, pointing out of the sensing disk, and that the result of the integration is also a vector (the normal vector is denoted by \( n(q) \) in Fig. 1).

The gradient function (8) provides an ascent direction for the area of the disk inside the polygon, which implies that moving the disk in this direction will increase the covered area within the polygon. This is intuitively expected from the geometry shown in Fig. 1. For a uniform density function, when the perimeter of the sensing disk is located completely inside the polygon or completely outside it, the above integral would be zero. Obviously, in these two special cases the overlapped area is maximized and can not be further increased by moving the sensing disk.

The points on the perimeter of the sensing disk can be characterized as:

\[
q = x_s + R_s \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}
\]

where \( \theta \in [0, 2\pi) \). As a result \( ||q - x_s|| = R_s, \forall q \in \partial_{m+1} \mu(x_s) \). On the other hand, the path \( \partial_{m+1} \mu(x_s) \) can also be described by a number of arcs defined by a set of intervals \( \Theta = \{ [\theta_1, \theta_2], [\theta_3, \theta_4], \ldots \} \). Thus the integral in the right side of (8) can be equivalently calculated as follows:

\[
\nabla_{x_s} F(x_s) = R_s \int_{\theta=0}^{\theta} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \phi(q) d\theta
\]

where \( q \) given in by (9). Note that \( dL \) in (8) is replaced by \( R_s d\theta \) in (10). This relation is important in that it presents a simple technique for finding the gradient by each individual mobile computing agent. More precisely, by choosing a sufficiently
large number of points on the perimeter and using a proper numerical integration method, one can find a sufficiently close approximation of (10).

In order to reformulate the underlying optimization as a more conventional minimization problem, define \( f(x_i) = -F(x_i) \). As a result, the OSL problem to a minimization form as follows:

\[
x_i^* = \arg\min_{x_i \in \mathcal{P}} f(x_i) = \arg\min_{x_i \in \mathcal{P}} \left[ - \int_{P \cap D(x_i)} \phi(q) dq \right]
\]

Also for the sake of simplicity we denote the decision variables by \( x \) instead of \( x_i \). Obviously the gradient is computed as follows:

\[
\nabla f(x) = -R_s \int_{\theta \in \Theta} n(q) \phi(q) d\theta
\]

Before starting to present the elements of the proposed nonlinear optimization approach, some analytical results about the OSL problem are expressed in the next subsection.

**B. Analytical Results about the Solution**

Using the gradient vector found in the previous subsection, some interesting analytical points are presented in this subsection. These results are obtained by investigation of the *Karush-Kuhn-Tucker (KKT) optimality conditions* for the solution of a constrained nonlinear optimization problem ([24], p. 342).

The next theorem expresses helpful analytical results about the optimum point of the defined objective function.

**Theorem 1:** Considering the optimization problem (11), one of the following facts holds at the optimum point \( x_i^* \):

(a) \( \nabla f(x_i^*) = 0 \)

(b) \( x_i^* \in \partial \mathcal{P} \)

Additionally for positive-valued functions of \( \phi(q) \), the first holds at the optimum and \( x_i^* \notin \partial \mathcal{P} \).

**Proof:** The KKT optimality conditions state that for \( x_i^* \) as a relative minimum point of the following general nonlinear optimization problem:

\[
\min f(x) \\
\text{subject to : } e(x) = 0, g(x) \leq 0
\]

there are vectors \( \lambda \) and \( \mu \) with \( \mu \geq 0 \) such that the following set of relations holds:

\[
\nabla f(x_i^*) + \lambda^T e(x_i^*) + \mu^T g(x_i^*) = 0 \\
\mu^T g(x_i^*) = 0
\]

There is no equality constraint for the OSL problem and the inequality constraints describing the Voronoi cell are considered as \( g(x) = Hx - K \) with the gradient as \( \nabla g(x) = H \). The optimum point either belongs to the polygon border (where (b) holds) or not. If \( x_i^* \) is an interior point of the polygon, the constraints hold in a strict sense \( (g(x) < 0) \) and this follows that \( \mu = 0 \). This in turn implies the (a) case above.

For the last part of the theorem, note that for each facet of the polygon described by \( g_j(x) = 0 \), the \( \nabla g_j(x) \) is a vector perpendicular to the facet and pointing outside the polygon. Now we prove the last part by contradiction. Assume that the optimum point is located on \( \partial \mathcal{P} \). The point should be a vertex or simply a point on a facet like the \( i \)-th one. For the latter case, it follows that \( g_j(x_i^*) \neq 0, \forall j \in \{1, \ldots, m\}, j \neq i \) and from the KKT conditions \( \mu_j = 0, \forall j \in \{1, \ldots, m\}, j \neq i \). Again using the KKT conditions, it follows that:

\[
\mu_i \nabla g_i(x_i^*) = -\nabla f(x_i^*) = R_s \int_{\theta \in \Theta} n(q) \phi(q) d\theta
\]

Note that \( \mu_i \geq 0 \). Multiplication of the both side by vector \( \nabla g_i(x_i^*)^T \) will result an inner product as follows:

\[
\mu_i \|\nabla g_i(x_i^*)\|^2 = R_s \int_{\theta \in \Theta} \nabla g_i(x_i^*)^T n(q) \phi(q) d\theta
\]
In case where a vertex might be the optimum point, two constraints would be active at $x^*_i$. Denoting the indices of the active constraint as $i$ and $k$, the similar observation about the KKT conditions yields:

$$
\mu_i \nabla g_i(x^*_i) + \mu_k \nabla g_k(x^*_i) = -\nabla f(x^*_i) = R_x \int_{\theta \in \Theta} n(q) \phi(q) d\theta
$$

(18)

Multiplication of the both side by the transpose of left-hand side vector results:

$$
\|\mu_i \nabla g_i(x^*_i) + \mu_k \nabla g_k(x^*_i)\|^2 = R_x \mu_i \int_{\theta \in \Theta} [\nabla g_i(x^*_i)T n(q)] \phi(q) d\theta 
+ R_x \mu_k \int_{\theta \in \Theta} [\nabla g_k(x^*_i)T n(q)] \phi(q) d\theta
$$

(19)

Again the left-hand side is positive but both terms of the right-hand side are negative. This completes the proof by contradiction.

This analytical result can be used to define a stopping criterion for the iterative nonlinear optimization algorithm. When moving in a descent direction, the optimum point will be reached at the border of the Voronoi polygon or when the gradient goes to zero. Specially for positive-valued density functions the stationary point of the generated sequence can be distinguished by the gradient measure only.

C. Line Search

Having a descent direction at hand, it is of great importance to find the specific amount of movement in the selected direction. This is usually performed using one of the several existing line search methods.

In a category of line search algorithms we are interested in finding optimum value of $\alpha$ as the solution to the following optimization problem:

$$
\alpha_k = \arg \min_{\alpha} f(x_k + \alpha p_k)
$$

(20)

For explicitly defined function of $f(x)$ this is an optimization problem in scaler $\alpha$ and can be easily performed using simple optimization methods. However for the OSL problem the objective function is known implicitly and the above-mentioned approach is not usable. Alternatively one can run a direct line search in which the optimum value of $\alpha$ is found by iteratively evaluating the objective function on the selected line. This requires the computation of the weighted intersected area of the polygon and the sensing disk while the center of the disk is moving on the selected line.

This is a time-consuming task and to avoid it we propose to do the line search based on the information obtained from the gradient vector.

It can be easily observed the fact that differentiation of the function $f(x_k + \alpha p_k)$ with respect to the parameter $\alpha$ will result

$$
p_kT \nabla f(x_k + \alpha p_k) = 0
$$

This means that at the optimum point, the gradient is perpendicular to the search direction $p_k$. So by defining the function:

$$
M(\alpha) = p_kT \nabla f(x_k + \alpha p_k)
$$

(21)

we can find the zero of $M(\alpha)$ by iteratively evaluating its value to do a bisection or Newton-Raphson search. Obviously this is much less computationally demanding than the direct line search approach.

The zero of $M(\alpha)$ could also be found using a Newton-Raphson search method which has faster convergence but needs also computing the value of $dM(\alpha)/d\alpha$ which might be hard to find.

Note that in general using the optimum value of $\alpha$ at some iterations might lead to a point $x_{k+1}$ which is outside the Voronoi polygon. This is not a problem since we will run a projection procedure in such cases as is detailed in the next subsection.

D. Projection onto the Voronoi Polygon

As an element of the proposed nonlinear optimization approach, in case the generated points fall outside the convex Voronoi polygon, they will be projected onto the polygon. For a typical point $x_0 \notin P$, the projected point in $P$ is denoted by $[x_0]_P$ and is defined as follows:

$$
[x_0]_P = \arg \min_{x \in P} \|x - x_0\|^2
$$

(22)

The problem simply seeks for the nearest point to $x_0$ inside $P$ and in general requires solving a quadratic programming problem. However the convexity of the Voronoi polygons and simplicity of the geometry make the projection an easy task to do. Following lemma states that in this case the projected point is either a vertex of the Voronoi polygon or a perpendicular foot on its facets.

Lemma: Consider the Voronoi polygon $P$ with $m$ facets and denote the perpendicular foots of $x_0$ onto the facets by $(x_0)_i^\perp$ for $i = 1, \ldots, m$. Note that in general $(x_0)_i^\perp$ might or might not belong to $\partial P$. Define the set $I$ as:

$$
I = \{i \in \{1, \ldots, m\} : (x_0)_i^\perp \in \partial P\}
$$

(23)
and the corresponding points as:
\[(x_0)^T = \{(x_0)^T : i \in I\}\] (24)

Then:
\[x_0|_p = \arg \min_{x \in A} \|x - x_0\|^2\] (25)

where:
\[A = \begin{cases} (x_0)^T, & I \neq \phi \\ \{v_1, \ldots, v_m\}, & I = \phi \end{cases}\] (26)

**Proof:** The lemma simply states that the search for finding the projected point can be limited to the set of perpendicular feet on the border of the polygon (if any) and the vertices of the polygon.

The proof of the lemma results by considering the KKT optimality conditions for the quadratic programming problem (22) as is omitted here for the sake of brevity.

Based on the above lemma we can verify the candidate points to find the projected point. This an easy task for the coverage optimization problem since there are almost always just a few points to be considered.

### E. Scaled Gradient Projection Algorithm

The proposed nonlinear optimization algorithm is a gradient projection method which is equipped with non-diagonal scaling of the descent direction. The descent-based methods of optimization might show zigzagging behavior while approaching the optimum point. It is generally known that utilization of the Hessian matrix of the objective function \(\nabla^2 f(x)\) results in better convergence of descent-based algorithms. The simplest form of such algorithms is based on finding the descent direction from following equation:
\[\nabla^2 f(x_k)p_k = -\nabla f(x_k)\] (27)
or equivalently:
\[p_k = -[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)\] (28)

In many situations (like the OSL problem) computation of the Hessian matrix and its inverse is a computationally hard task to do. To resolve the problem some approaches are developed which try to find and maintain in each iteration a category which is known as Broyden class. The approximate matrix to the inverse of the Hessian matrix (like the unity matrix) and updated in each iteration in such a way to preserve the positive-definiteness property ([25], p. 470). Defining the values of \(s_k\) and \(y_k\) as follows:
\[s_k = x_{k+1} - x_k\] (29)
\[y_k = \nabla f(x_{k+1}) - \nabla f(x_k)\] (30)

and denoting the approximation to the inverse of the Hessian matrix by \(H_k\), the Sherman-Morrison updating formula is as follows:
\[H_{k+1} = \left[ I - \frac{s_ky_k^T}{y_k^Ts_k} \right] H_k \left[ I - \frac{y_ky_k^T}{y_k^Ts_k} \right] + \frac{s_ky_k^T}{y_k^Ts_k}\] (31)

Now the descent direction is found similar to the equation (28) as:
\[p_k = -H_k\nabla f(x_k)\] (32)

The existing experience in the nonlinear optimization theory confirms the improvement of the gradient projection algorithms when the descent direction is scaled by a matrix as above ([26], p. 233)

Regarding the termination of the iterative algorithm, we consider the cases where the optimum point is located inside the polygon or where it’s over the border. For the former case the value of \(\|\nabla f(x)\|\) is a reasonable value to decide about proximity to the optimum point. Since the updating formula for \(H_k\) also uses the difference of the gradient in the two last steps of the algorithms, it’s good for numerical stability of algorithm implementation to check the value of \(\|y_k\|\) also.

In case where \(x^*_k \in \partial P\), it can be shown that for the optimum point \(x^*_k = [x^*_k]^T\). This means that \(\|s_k\|\) represents the proximity to the optimum point. Based on these facts a small positive number \(\varepsilon\) is selected before running the algorithm and three mentioned values \(\|\nabla f(x)\|, \|y_k\|\) and \(\|s_k\|\) are evaluated at each round. The algorithm will be terminated whenever either of these values are smaller than the selected threshold \(\varepsilon\). We refer to this procedure in the following as evaluating the termination condition.

Using the important elements described so far, the nonlinear optimization approach for solving the OSL problem is stated as below. Note that as stated before, the solution to the OSL problem is known for large values of \(R_i\) and with uniform
density function for small values of $R_s$. The following algorithm needs to be run when the sensing range $R_s$ is such that the solution to the OSL problem can not be obviously found. This will further clarified in section V.

**Scaled Gradient Projection Algorithm**

1. Initialize the optimum value of the sensor position by the current position of the sensor, i.e. $k = 0$ and $x_k = x_s$. Also initialize the $H_k$ matrix as $H_0 = I$.
2. Compute the value of $\nabla f(x_k)$ from (12) and the descent direction $p_k$ from (32).
3. Perform the line search procedure as described above to find the value of $\alpha_k$.
4. Update the optimum value by setting $x_k+1 = x_k + \alpha_k p_k$.
5. If the value of $x_k+1$ falls outside the Voronoi polygon $P$, use the projection procedure and set $x_k+1 = [x_k+1]_P$.
6. Compute the value of $\nabla f(x_k+1)$.
7. Test for optimality of the solution. If passed, set $x^* = x_k+1$ and terminate the algorithm. If not go to the next step.
8. Find $s_k$ and $y_k$ from (29) and (30) respectively and the $H_{k+1}$ matrix using the updating formula (31).
9. set $k = k + 1$ and go to step 3.

To clarify the proposed algorithm, consider the OSL problem for a typical polygon depicted in Fig. 3 where the sensing radius is assumed to be as $R_s = 1.5m$ and the target field has non-uniform density function as follows:

$$\phi(q) = \exp[-2(q_1 - 2.8)^2 - 2(q_2 - 1)^2]$$

To verify the correctness of the algorithm the level sets of the objective function are also depicted in Fig. 3. Note that the defined objective function for the OSL problem (relation 11), has implicitly considered the constraints and so the level sets are not centered around the maximum point of $\phi(q)$ which is $q = [2.8, 1]^T$. In fact placing a disk at $q = [2.8, 1]^T$ would result in a far-from-optimum value for the objective function since the intersected area of the disk and the polygon would be small.

Starting from the point $x_0 = [2.5, 3.7]^T$, the generated points for the first two steps of the algorithm fall outside the polygon and the projection procedure has to be used. The generated path toward the optimum point is also shown in Fig. 3.

To better clarify the projection behavior, a zoomed view of the points near the border of the polygon is also depicted in Fig. 4.

As is seen, the algorithm converges in 5 steps to the optimum point $x^*_s = [1.80, 2.05]^T$. Some remarks are mentioned here about the proposed algorithm.

**Remark 1** Note that the optimization does not need new data to be measured during the optimization, so the sensor does not need to move before termination of the iterative optimization procedure. The sensor is virtually moved toward the optimum points and the real movement just takes place after finding the optimum point inside the polygon.

**Remark 2** For uniform density function, having the solution for large and small values of $R_s$ (as described before), suggests a subtle point. When $R_{cheb} < R_s < R_{enc}$ and the optimization algorithm has to be run, it can be wisely initialized based on the proximity of $R_s$ to $R_{enc}$ and $R_{cheb}$. A good initial point in this case can be defined as follows:

$$x_0 = x_{cheb} + \frac{R_s - R_{cheb}}{R_{enc} - R_{cheb}}(x_{enc} - x_{cheb})$$

(33)
The defined initial points here is on the line joining $x_{cheb}$ and $x_{enc}$ and based on the value of $R_s$ it would be closer to one of the two ends of the line. Simulation shows that the iterative optimization converges much faster using this initial point.

**Remark 3** The sensor location problem is discussed in a pioneering work by Cortés et al. [27] as area problem. The gradient relation is derived in an alternative way and a gradient-based algorithm is briefly presented which seems to be a steepest descent algorithm equipped with some kind of line search. However the approach presented here considers several aspects of the problem like the constraints and the improvement by scaling of the steepest descent direction.

V. MULTI-SENSOR COVERAGE OPTIMIZATION

The same strategy for optimum sensor location is used by all sensors in multi-sensor framework of coverage optimization. Some relevant issues about the general behavior of the Max-area (MaxS) algorithm is discussed here.

The convergence of the proposed approach in multi-sensor situation is assured by discrete-time version of well-known LaSalle’s invariance principle [28]. The procedure of Max-area (MaxS) can be viewed as a discrete-time map which at each time instance updates the position of the sensors based on the solution to the OSL problem. As stated before, it’s a known fact that using any coverage control strategy which aims at increasing the local coverage of each sensor inside its polygon, the total area would be non-decreasing [9]. Considering the negated value of the total covered area as the Lyapunov function, the convergence to a set of critical points of the Lyapunov function is guaranteed by the Lasalle’s invariance principle. Under finiteness of the critical points, this results in the convergence of movement of sensors.

A steady state topology of the network is described by a situation in which applying the point selection and movement strategy does not result in change of topology. This means that the sensors are either in the optimum point of their Voronoi polygon or can not increase their local coverage by moving to such a point. In some articles this situation of the network is called a critical point of topology [27]. A typical coverage optimization problem has many critical points and so the final coverage pattern depends on initial point and the selected strategy to optimize local coverage of the sensors. Achieving the optimal coverage performance by distributed decision making strategies is still an open problem. For this reason the different coverage optimization algorithms are usually compared to each other by Monte Carlo simulations where the performance criteria are assessed by simulating the algorithms with many initial random values. We use the same strategy here to show the effectiveness of the proposed algorithm compared to Minimax approach as a well-known efficient coverage optimization approach [9].

Speed of convergence, final coverage pattern and traveled distance (as a measure of energy consumption) are common assessment criteria for a multi-sensor coverage optimization algorithm.

To present the behavior of Max-area (MaxS) in multi-agent framework it’s been simulated for a typical situation in uniform 50m by 50m square target field with 24 moving sensors with $R_s = 6m$ and $R_{com} = 20m$. Initial topology of the sensors and corresponding Voronoi diagram is depicted in Fig. 5. The movement paths of the sensors under Max-area (MaxS) are also shown on the same figure.

Starting from total coverage of 63% of the whole field, the proposed algorithm achieves the final coverage of 88% of the whole field in 16 rounds. Note that the whole movement paths of the sensors for 16 rounds are drawn and ended by the yellow points in Fig. 5. The final topology is shown in Fig. 6. Considering the short movement paths and 25% improvement in total coverage, Max-area (MaxS) shows reasonable performance.

To compare the performances of Max-area (MaxS) and Minimax, we used 50 different initial topologies of 30 similar sensors and for each situation both algorithms were run. Then we take the average of total coverage percentage for each
algorithm. The result is depicted in Fig. 7. The figure shows that Max-area (MaxS) outperforms the Minimax in terms of total coverage as well as speed of convergence.

Note that for each algorithm there are 50 different stopping round numbers and to do the averaging we prolongate the convergent experiment up to the latest round in all 50 experiments. The above figure shows that the latest stopping round is 15 for Max-area (MaxS) and 19 for Minimax. The average value of traveled distance while the agents are commanded by Max-area (MaxS) is 118.7 meters and for Minimax this value is equal to 119.1 meters. The difference is negligible but the Max-area (MaxS) is not outperformed by Minimax. Note that the price for better performance of Max-area (MaxS) is its higher computational complexity although its reasonable from implementation point of view.

VI. CONCLUSION

To optimize the total coverage of a group of mobile sensing agents, the Max-area (MaxS) approach utilizes the distributed information available to each sensor. The Voronoi diagram used to optimally partition the target field and the proposed scaled gradient projection algorithm optimally locates each sensor inside its corresponding Voronoi polygon. Comparative performance assessment of the Max-area (MaxS) algorithm by Monte Carlo simulations shows that it outperforms well-known coverage optimization approaches in terms of total coverage and speed of convergence. While the algorithm is supported by strong mathematical bases, its implementation into each sensing device is simple and suitable for operational situations.
Fig. 7: Percentage of total covered area in consecutive rounds of Max-area (MaxS) and Minimax algorithms

REFERENCES


