<table>
<thead>
<tr>
<th>Title</th>
<th>Autonomous deployment for load balancing k-surface coverage in sensor networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Li, F; Luo, J; Wang, WP; He, Y</td>
</tr>
<tr>
<td>Citation</td>
<td>IEEE Transactions on Wireless Communications, 2015, v. 14 n. 1, p. 279-293</td>
</tr>
<tr>
<td>Issued Date</td>
<td>2015</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10722/206077">http://hdl.handle.net/10722/206077</a></td>
</tr>
<tr>
<td>Rights</td>
<td>This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.; ©2015 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.</td>
</tr>
</tbody>
</table>
Abstract—Although the problem of \(k\)-area coverage has been intensively investigated for dense wireless sensor networks (WSNs), how to arrive at a \(k\)-coverage sensor deployment that optimizes certain objectives in relatively sparse WSNs still faces both theoretical and practical difficulties. Moreover, only a handful of centralized algorithms have been proposed to elevate 2-D area coverage to 3-D surface coverage. In this paper, we present a practical algorithm, i.e., the Autonomous dePLOYment for Load baLancing \(k\)-surface eCoverage (APOLLO), to move sensor nodes toward \(k\)-surface coverage, aiming at minimizing the maximum sensing range required by the nodes. APOLLO enables purely autonomous node deployment as it only entails localized computations. We prove the termination of the algorithm and the (local) optimality of the output. We also show that our optimization objective is closely related to other frequently considered objectives for 2-D area coverage. Therefore, our practical algorithm design also contributes to the theoretical understanding of the 2-D \(k\)-area coverage problem. Finally, we use extensive simulation results to both confirm our theoretical claims and demonstrate the efficacy of APOLLO.

Index Terms—Wireless sensor networks (WSNs), autonomous deployment, \(k\)-area/surface coverage, load balancing.

I. INTRODUCTION

ONE of the major functions of wireless sensor networks (WSNs) is to monitor a certain area in terms of whatever physical quantities demanded by applications [2]. In achieving this goal, a basic requirement imposed onto WSNs is their physical quantities demanded by applications [2]. In achieving deployments (e.g., [33], [45]), there exists an unfailing trend or scheduling sensor activity to retain the coverage of given performance of static sensor deployments (e.g., [5], [25]) where the extent that every point in this area/surface is initially. Our goal is to cover a certain monitored area or surface to the extent that every point in this area/surface is

Due to the vulnerability of sensor nodes, multiple-coverage \((k\)-coverage\) is often applied to enhance the fault tolerance in face of node failures (e.g., [5], [45]). In addition, \(k\)-coverage may yield higher sensing accuracy through data fusion [13]. Existing approaches in achieving \(k\)-coverage rely on either randomized (e.g., [15], [25]) or regular (e.g., [3], [5]) deployments. Whereas randomized deployments require a substantially denser network (e.g., [15], [25]), regular deployments serve only as theoretical guidelines [3], [5] as they often require centralized coordinations and may not accommodate irregular network regions. Also, if the physical phenomena under surveillance change, the cost for re-deployment can be huge. Therefore, autonomous deployments, when movable nodes [10] are available, are good complements to the randomized or regular deployments: they may achieve a density comparable to that of regular deployments, while being more adaptive to irregularity and variations of the network regions.

However, existing techniques for autonomous deployments may only handle 1-coverage, and extending them to \(k\)-coverage is highly nontrivial. First of all, autonomous deployments through (node) motion control require each node to compute its coverage in a localized manner (i.e., relying as much as possible on close-by nodes). Although quite a few localized algorithms have been proposed to perform such computations for 1-coverage (e.g., [7], [34]), no algorithm, to the best of our knowledge, exists for localized \(k\)-coverage computations.

Secondly, even if the \(k\)-coverage computations can be performed locally, there is no guarantee whether a motion control strategy may converge, due to the significant difference between 1-coverage and \(k\)-coverage. Thirdly, existing approaches are almost heuristics that offer no provable guarantee on the quality of the eventual deployment. Finally, as indicated in [43], extending 2-D area coverage (e.g., [28]) to 3-D surface coverage for coverage for WSNs deployed on 3-D terrains may incur new challenges even for 1-coverage. By far, only a handful of proposals employ (partially) centralized algorithms to handle the deployment for 1-surface coverage [18], [43]; no purely autonomous deployment mechanism has ever been proposed. These are exactly the problems we want to tackle in our paper.

In this paper, we consider the problem of moving sensor nodes towards \(k\)-coverage. In particular, we assume that nodes have tunable sensing ranges and are randomly deployed initially. Our goal is to cover a certain monitored area or surface to the extent that every point in this area/surface is
at least monitored by $k$ sensor nodes and that the maximum sensing range used by the nodes is minimized. As a larger sensing range implies a larger energy consumption of a node, our APOLLO (Autonomous dePloymt for Load balanCing $k$-surface cOverage) approach aims at balancing the sensing load (thus prolonging network lifetime) while guaranteeing $k$-coverage, with the help of mobile nodes. The main contributions we are making in this paper are:

- We design the APOLLO algorithm such that it executes in a localized manner, i.e., relying only on information from close-by nodes.
- We prove the termination of APOLLO as well as the (local) optimality of its output.
- We discuss the relation between APOLLO and other commonly used optimization objectives for 2-D area coverage, which provides a better understanding of optimal $k$-coverage deployments whose theoretical characterizations are hard to obtain under general settings.
- We show that, by using geodesic distance [9] to replace the conventional Euclidean metric, APOLLO can be naturally extended to handle autonomous deployments on 3-D surfaces.
- Since motion and communication cost cannot be neglected, our algorithm allows for a tradeoff between the two factors. Through extensive simulations driven by realistic power consumption data, we show that the existing hardware platform can afford the energy consumption of our algorithm in terms of motion and communication.

To the best of our knowledge, we are the first to tackle the problem of $k$-coverage autonomous deployment, for both 2-D areas and 3-D surfaces.

The remaining of our paper is organized as follows. We briefly survey the closely related literature in Section II. We formally define our model and problem in Section III, in which we also review the basic mathematical tools we need in our later algorithm design. In Section IV, we present our APOLLO algorithm for 2-D $k$-area coverage and analyze its performance, we also interpret our solution with respect to other optimization frameworks. We then extend APOLLO to handle 3-D $k$-surface coverage in Section V. The efficacy of APOLLO is further confirmed by extensive simulation results reported in Section VI. We finally conclude our paper in Section VII.

II. RELATED WORK

Before surveying the proposals related to area/surface coverage and mobile assisted autonomous deployment, we first review another interesting topics, point (or target) coverage and area coverage with random deployments, from which we may gain some hints for our proposal. Point coverage problem has been extensively studied in the past decade. Besides providing coverage service, their another concern is the limited energy supply of sensor nodes. Given a random deployment with static sensor nodes, they either divide the nodes into multiple sets and schedule the duties of these set (e.g. [8], [11], [44]), or minimize the number of the active sensor nodes (e.g., [14]), to guarantee the network lifetime. The energy limitation is also taken into account in area coverage with random deployments, e.g. [21], [25], [38], [41], [45]. Inspired by them, maximizing the network lifetime is also one of the main objectives of our proposal but in a quite different way.

The static and deterministic area coverage problem is essentially a geometry problem; the results for 1-coverage with a minimum number of nodes can be directly taken from pure mathematical research [22]. In later research proposals for WSN coverage, the focus is more on minimum node 1-coverage with certain connectivity requirement (e.g., [4], [5], [19]). While it is known that a 1-covered WSN is also connected if the transmission range $R_t$ and the sensing range $R_s$ satisfy $R_t \geq \sqrt{3} R_s$, a strip-based deployment strategy is proposed in [19] for other values of $R_s$, which is proven to be asymptotically optimal in [4]. In fact, more strips allows higher degree of connectivity (or $k$-connectivity).

Compared with $k$-connectivity, the progress on $k$-coverage appears to be relatively slower. A few 3-coverage heuristics that aim at bounding the minimum separation among sensor nodes is proposed in [23]; the paper also shows that bounding the min-separation may lead to lower coverage redundancy and is hence a good approximation to minimum node 3-coverage. To the best of our knowledge, the only optimality result in terms of minimum node $k$-coverage is presented in [5], where $k = 2$. It appears that minimum node $k$-coverage (for $k > 2$) is better to be tackled indirectly due to its hardness. As we will show in Section IV-C, our objective of a $k$-coverage with minimum sensing range may also imply minimum node $k$-coverage. Deploying WSNs for $k$-coverage using mobile nodes is also reported in [3], [36], but their approaches are not autonomous as they all rely on a “blueprint” to guide the node mobility.

Considering that a WSN can be deployed on a 3-D terrain for some application scenario, the surface coverage problem has also been investigated. Zhao et al. [24], [43] show that extending just 1-area coverage to 1-surface coverage already poses great challenges, and they provide approximation algorithms to the resulting NP-complete problem. Recently, Jin et al. [18] applies Centroid Voronoi Tessellation (CVT) [12] and discrete Ricci flow [17] to obtain some form of optimal surface 1-coverage by defining sensing range upon geodesic distance. However, the discrete Ricci flow is a global parametrization process, so the motion control cannot be implemented in a distributed manner using only local information.

Our work is also related to the sensing heterogeneity issue [6], [35]. Unlike the previous proposals that aim to cope with sensing heterogeneity or evaluate its impact, we actively exploit the sensing heterogeneity to construct our algorithm that guides the autonomous deployment.

III. PROBLEM DEFINITION AND MATHEMATICAL BACKGROUND

In this section, we first present the system model and define our optimization problem, and then introduce the relevant mathematical basics. To simplify the exposition, the above discussions are all for 2-D plane with Euclidean metric. We then show that a 2-D plane can be straightforwardly generalized to a 3-D surface by replacing Euclidean metric with geodesic distance,
and we also introduce the mathematical tools to perform this map locally.

A. System Model

We assume a WSN consisting of a set \( \mathcal{N} = \{n_1, \ldots, n_N\} \) of sensor nodes, and \( |\mathcal{N}| = N \). Let \( \mathcal{U} = \{u_1, \ldots, u_N\} \) denote the locations of sensor nodes. The nodes are initially deployed arbitrarily on a 2-D targeted area \( A \). Each node \( n_i \) is equipped with certain mechanisms (e.g., motors plus wheels) that allow it to gradually change its location \( u_i \) [10]. We also suppose that nodes are equipped with bumper sensors to detect and avoid obstacles in the targeted area [30]. All nodes have an identical transmission range \( r \), and we also introduce the mathematical tools to perform this map locally.

We define the omnidirectional sensing model as a disk centered at \( u_i \) with sensing range \( r_i \). We assume the sensing ranges are adjustable according to different application requirements [42], [45]. A point \( v \in A \) is said to be covered by node \( n_i \) iff the Euclidean distance between \( v \) and node location \( u_i \) is no longer than \( r_i \), i.e., \( \|v - u_i\|_2 \leq r_i \). We use \( f(v, u_i, r_i) \) to indicate if \( v \) is covered by node \( n_i \); \( f(v, u_i, r_i) = 1 \) if \( v \) is covered by \( n_i \); otherwise, \( f(v, u_i, r_i) = 0 \). We apply the MDS-based technique [32] that relies on the ranging ability of each node to construct a local coordinate system for motion control, but we do not require global location information as it is immaterial to our algorithm.

In terms of energy cost, we consider only the cost induced by the sensing activities of a node. Because our network deployment strategy aims to achieve a constant (and long-term) coverage by moving sensor nodes in the initial phase, the communication cost becomes negligible as the data transmission activities only take place sporadically, while the energy spent in moving is only a one-time investment. We assume that the energy consumed by a sensor node \( n_i \) is an increasing function \( E(r_i) \) of its sensing range \( r_i \), and this function is identical for all nodes. In other words, with a certain amount of energy \( E(r_i) \), \( n_i \) can only cover the points \( \{v \in A : \|v - u_i\|_2 \leq r_i\} \).

B. Problem Formulation

The node locations and sensing ranges define a network deployment with a certain coverage.

Definition: A network deployment \( \{u_i, r_i\} \) is said to achieve \( k \)-coverage iff for any point \( v \in A \), there exist at least \( k \) sensor nodes covering it, or \( \sum_j f(v, u_i, r_i) \geq k \).

To allow the sensor nodes \( k \)-cover the targeted area, we divide \( A \) into several disjoint areas \( \mathcal{A}_j \), \( j = 1, 2, \ldots \), and at least \( k \) sensor nodes are allocated to take care of each area. In other words, each sensor node \( n_i \) takes care of multiple subareas, and we indicate this covering relation by \( n_i(\mathcal{A}_j) \): it equals 1 if \( n_i \) covers \( \mathcal{A}_j \); otherwise 0. We also denote by \( \mathcal{A}_n \) the area covered by \( n_i \); we have \( \mathcal{A}_n = \bigcup_{n_i(\mathcal{A}_j) = 1} \mathcal{A}_j \). The sensing range \( r_i \) of \( n_i \) is determined by the farthest point in \( \mathcal{A}_n \) from \( u_i \), i.e., \( r_i = \max_{v \in \mathcal{A}_n} \|v - u_i\|_2 \), so that \( \mathcal{A}_n \) can be totally covered by \( n_i \).

Our \( k \)-coverage sensor deployment problem (\( k \)-CSDP) can be formulated as follows.

\[
\text{minimize} \quad \{u_i, \{A_j^k\}, \{n_i(A_j^k)\}\} \\
\text{subject to} \quad \sum_{i=1}^N f(v, u_i, r_i) \geq k, \quad \forall v \in A \\
\|v - u_i\|_2 \leq R, \quad \forall i, v \in A_j^k \\
A_j^k \cap A_j^k = \emptyset, \quad \bigcup_j A_j^k = A. 
\]

Literally, \( k \)-CSDP aims at determining the node locations \( \{u_i\} \), the area partition \( \{A_j^k\} \), and the covering relations \( \{n_i(A_j^k)\} \), such that the targeted area \( A \) is \( k \)-covered, while the maximum sensing range among all nodes is minimized. As energy consumption is an increasing function of sensing range, \( k \)-CSDP is equivalently balancing the energy consumption over a whole WSN and hence maximizing the lifetime of the WSN. As the problem is generally not convex due to its non-convex feasible region, we have to be contented with local minimum (i.e., a locally minimized maximum sensing range).

C. High Order Voronoi Diagram

We hereby briefly introduce the ideas and theories on high order Voronoi diagram [31]. They are key to our autonomous deployment strategy.

In a \( k \)-voronoi diagram, the targeted area \( A \) is segmented into \( \mathcal{N}^k \) disjoint areas \( \{V_j^k\}_{j=1, \ldots, \mathcal{N}^k} \) each of which is associated with \( k \) closest generators (sensor nodes in our case), i.e., a subset \( \mathcal{N}^k \subset \mathcal{N} \) with \( |\mathcal{N}^k| = k \). The \( k \)-order Voronoi cell \( V_j^k \) is defined as

\[
V_j^k = \left\{ v \in A : \|v - u_i\|_2 \leq \|v - u_j\|_2, \quad \forall n_i \in \mathcal{N}^k, \quad n_i \in \mathcal{N}/\mathcal{N}^k \right\}. 
\]

The set \( \mathcal{N}^k \) is called the generator set of \( V_j^k \). It is straightforward to see that each sensor node \( n_i \) is associated with multiple Voronoi cells. Let \( \mathcal{V}_n^k \) denote the union of the Voronoi cells for which \( n_i \) serves as a generator; we term \( \mathcal{V}_n^k \) the dominating region of \( n_i \) (hence \( n_i \) the dominator of \( V_j^k \)). We also have the following proposition.

Proposition 1: A point \( v \in A \) is said to belong to \( V_j^k \) iff there exist at most \( k - 1 \) other generators such that their distance to \( v \) is less than \( \|v - u_j\|_2 \).

Proof: Assume \( v \in \mathcal{N}^k \), but there were another \( k \) nodes \( \{n_j\}_{j \in J, \|u_j - V_j\|_2 < \|u_j - v\|_2} \). Then the point \( v \) would strictly belong to the set of \( k \)-order Voronoi cells generated by \( \{n_j\} \), which does not include \( n_i \) as a generator;

\[2\]In 1-order Voronoi diagram, the number of Voronoi cells equals the number of generators (i.e., \( \mathcal{N}^1 = \mathcal{N} \)), while in generalized \( k \)-order Voronoi diagram (\( k \geq 1 \)), \( \mathcal{N}^k \) is \( \mathcal{O}(k(N-k)) \) [31].
a contradiction. Conversely, if there are at most \( k - 1 \) nodes, \( \{ n_j \}_{j \in I, \text{gcd}(I,J) \leq k-1} \), closer to \( v \) than \( n_i \), we can find the set of \( v \)'s \( k \)-nearest nodes (which obviously contains \( \{ n_j \} \cup \{ n_i \} \)) to generate a set of \( k \)-order Voronoi cells containing \( v \). As \( n_i \) is a generator, \( v \in V_{n_i}^k \).

Based on the above proposition, assuming \( S_{n_i}^k (v) = \{ n_j \in \mathcal{N} \mid \| u_j - v \|_2 < \| u_i - v \|_2, j \neq i \} \), we can re-define the dominating region of \( n_i \) as

\[
V_{n_i}^k = \{ v \in A \mid |S_{n_i}^k (v)| \leq k - 1 \}.
\]

We illustrate \( k \)-order Voronoi partition \( (k = 1, 2, 3, 4) \) generated by 30 nodes in Fig. 1. The cells shown in each figure are \( \{ V_{n_i}^k \} \). Taking 2-order Voronoi partition for example, as shown in Fig. 1(b), the area enclosed by red (resp. green) polygon is actually the dominating region of the red node (resp. green node). The hatched area is the Voronoi cell generated by the two nodes, i.e., the points in this area are closer to the two nodes than any other nodes.

### D. Generalization to 3-D Surfaces

For WSN deployed on a 3-D surface, the conventional Euclidean metric is no longer appropriate. Serving as the generalization of “straight line” in curved space (e.g., 3-D surfaces), geodesic is the shortest path between two given points on the surface [9]. Therefore, geodesic distance metric is a natural choice for measuring distance on a 3-D surface. By replacing Euclidean metric with geodetic distance, we may migrate the aforementioned model and problem definitions directly from 2-D planes to 3-D surfaces.

1) **Problem Description on 3-D Surfaces**: Consider the case where a WSN \( \mathcal{N} \) is deployed on the 3-D surface \( \mathcal{M} \), we use geodesic distance as the distance metric. In particular, we replace all the distant metric \( \| u - v \|_2 \) used for 2-D planes by \( g(u, v) \), the geodesic distance between \( u \) and \( v \). As a result, all the definitions in Section III-A to III-C can be directly migrated to 3-D surface. For example, \( k \)-CSDP and high order Voronoi diagram can be redefined using geodesic metric. The only difference here is that, while \( A \) does not need an explicit characterization, \( \mathcal{M} \) is often represented by a triangular mesh that is given to all nodes during the initialization phase.

In order to compute geodesic distance on \( \mathcal{M} \), we adopt the Improved Chen & Han’s (ICH) algorithm [39]. The ICH algorithm handles the “single source, all destinations” geodesic problem, aiming at computing the geodesic from the source \( u \in \mathcal{M} \) to any destination point \( v \in \mathcal{M} \) within a certain geodesic radius \( r \) from near to far with a time complexity of \( O(n^2 \log n) \) where \( n \) is the number of vertices of the triangular mesh \( \mathcal{M} \) within \( r \). It maintains a priority queue of geodesic windows on the edges of the triangular mesh, and outputs a poly-line geodesic path for each pair of source and destination. The ICH algorithm also can be extended to other types of geodesic problem [40], e.g., “single source, single destination” and “multiple sources, any destination”. Such a package of geodesic computation tools is sufficient for our 3-D extension.

2) **Logarithm and Exponential Maps**: As we need a local coordinate system to compute \( k \)-order Voronoi diagrams in a localized manner for every node, we cannot rely on a parameterization method such as Ricci flow due to its global nature and high computational cost. We instead apply the ICH algorithm to compute an logarithm/exponential map [9] around a certain node, which constructs a (local) geodesic polar coordinate system on the curved surface as follows.

Given a point \( u \in \mathcal{M} \), we designate a local coordinate system on its tangent plane \( T_u \) centered at \( u \). A logarithm map \( \exp_u^{-1} : \mathcal{M} \to T_u \) maps the points on \( \mathcal{M} \) to \( T_u \). In particular, for any other point \( v \in \mathcal{M} \) in a sufficiently small neighborhood of \( u \), there is a unique geodesic \( \Theta_u(v) \) extending from \( u \) to \( v \). As \( \Theta_u'(v) \), the tangent of \( \Theta_u(v) \), is obviously tangent to \( \mathcal{M} \) at \( u \), \( \Theta_u'(v) \in T_u \). Therefore, \( v \) can be mapped to \( T_u \) with geodesic polar coordinates \( \{ g(u, v), \theta_{u,v} \} \) where \( g(u, v) \) is the geodesic distance between \( u \) and \( v \) and \( \theta_{u,v} \) is the polar angle of \( \Theta_u'(v) \) on \( T_u \). Consequently, the geodesic coordinates on \( \mathcal{M} \) around \( u \) can be transformed to normal coordinates under any orthogonal basis \( \{ e_1, e_2 \} \) with origin \( u \). The inverse of logarithm map is called exponential map. We illustrate the log/exp map in Fig. 2.

Here the ICH algorithm [39] is used to trace the geodesic \( \Theta_u(v) \) such that any point \( v \) in the neighborhood of \( u \) on \( \mathcal{M} \) can get its geodesic polar coordinates on \( T_u \). As mentioned above, the ICH algorithm maintains a “wavefront” and propagates outward from \( u \), finally outputting a local geodesic coordinate system within a radius \( r \).

### IV. APOLLO: LOCALIZED \( k \)-COVERAGE NODE DEPLOYMENT ALGORITHM

In this section, we first develop two optimality conditions for \( k \)-CSDP. Then we present the APOLLO algorithm details. The
correctness of APOLLO is then proven, and we finally discuss the relation between \(k\)-CSDP and other optimization problems related to \(k\)-coverage deployment, along with the corresponding properties of APOLLO.

### A. Optimal Conditions

To motivate our algorithm, we develop two optimality conditions for \(k\)-CSDP. Firstly, we show if we fix \(\{u_i\}\), then \(k\)-order Voronoi diagram is the optimal solution to \(k\)-CSDP.

**Proposition 2:** If we fix the sensor locations \(\{u_i\}\), the \(k\)-order Voronoi diagram \(\{V^k_i\}\) generated by \(\{u_i\}\) is an optimal partition of \(A\). Also, \(n_i(V^k_i) = 1\) if \(V^k_i \subseteq V^k_{\bar{n}_i}\); otherwise \(n_i(V^k_i) = 0\).

**Proof:** The proof is by contradiction. Suppose for fixed \(\{u_i\}_{i=1,\ldots,N}\), there exists an optimal solution to \(k\)-CSDP denoted by \(R^*, \{A^*_{k}\}, \{n_i(A^*_{k})\}\). Let \(r^*_{i} = \max_{v \in A^*_{k_i}} \|v - u_i\|_2\) and \(r^*_{i} = \max_{v \in V^k_{A^*_{k_i}}} \|v - u_i\|_2\). Also assume that the optimal value is obtained for \(n_i\); i.e., \(R^* = r^*_i\). If \(r^*_{i} = r^*_{i}\), then it is straightforward to see that \(r^*_{i} = \max_i\{r^*_{i}\}\), otherwise a contradiction to the definition of \(V^k_{A^*_{k_i}}\); some regions are not covered by the \(k\)-closest nodes. Therefore, in this case the \(k\)-order Voronoi diagram is equally optimal. If \(r^*_{i} > r^*_{i}\), it means that \(n_i\) could cede a certain region to have it covered by other nodes while reducing \(\max_i\{r^*_{i}\}\). However, this again contradicts the definition of \(V^k_{A^*_{k_i}}\): as \(n_i\) is already one of the \(k\)-closest nodes that can cover the ceded region, ceding this region to some other nodes would increase \(\max_i\{r^*_{i}\}\).

Before stating the second optimality condition, we need the following definition.

**Definition 2:** Given an arbitrary set \(S\) in Euclidean space, the Chebyshev center \(u_c\) is defined as \(u_c = \arg\min_{u} (\max_{a \in S} \|u - a\|_2)\).

Given an area partition \(\{A^*_{k}\}\) and its dominator allocation (or covering relations) \(\{n_i(A^*_{k})\}\), the optimal locations of \(\{n_i\}\) can be obtained according to the following proposition.

**Proposition 3:** If we fix the partition \(\{A^*_{k}\}\) and its dominator allocation \(\{n_i(A^*_{k})\}\), the optimal sensor location \(u^*_i\) for \(k\)-CSDP is given by the Chebyshev center of \(A^*_{k_i}\).

**Proof:** As \(n_i\) needs to cover \(A^*_{k_i}\) and the objective of \(k\)-CSDP is to minimize the maximum sensing range among all sensors, the optimal solution under a fixed partition is achieved if each sensor individually minimizes its own sensing range. This exactly coincides with the property of Chebyshev center, hence the proposition follows.

### B. The Algorithm

Given the two optimality conditions stated in Section IV-A, we immediately have an iterative algorithm to solve \(k\)-CSDP. The pseudo-codes of our APOLLO algorithm are presented in Algorithm 1. The algorithm proceeds in rounds. At the beginning of each round, the \(k\)-order Voronoi diagram is computed for the whole WSN, resulting in \(\{V^k_{1_N}, \ldots, V^k_{N_k}\}\) along with \(\{V^k_{n_1}, \ldots, V^k_{n_N}\}\) (Line 2). Then each node computes the Chebyshev center of its dominating region (Line 3), and moves to that location to end this round (Line 4–6). The algorithm terminates if each node is indeed located at the Chebyshev center of its dominating region. As a perfect matching is impossible in face of numerical errors, we use a small value \(\varepsilon\) as the stopping tolerance: the algorithm terminates if the distance from the node’s current location to the Chebyshev center of its dominating region is smaller than \(\varepsilon\). Also, in order to avoid oscillation, a step size \(\alpha < 1\) is chosen to confine the motion of the nodes. At the termination, each node tunes its sensing range to be the minimum value (the circumradius of the dominating region) that covers its dominating region. As a dominating region is a polygon, we apply Welzl’s algorithm [37] to compute the Chebyshev center by taking the vertices of the region as the input.

![Algorithm 1: APOLLO](image)

Similar algorithms have been applied in [6], [34], but they were used to indirectly optimize a different objective (see our discussions in Section IV-C). Consequently, their approaches do not abide by the optimality conditions and employ a very different termination condition. Therefore, their termination proofs do not apply to our case even for \(k = 1\). Most importantly, as our algorithm deals with a more general \(k\)-coverage, we are facing the following new challenges in understanding our algorithm: 1) How to compute \(V^k_{n_i}\) in a localized manner without involving all nodes? 2) Does the algorithm terminate for \(k \geq 1\)? 3) What is the complexity of computations? We tackle these challenges in the following.

1) **Localized Algorithm for Computing \(V^k_{n_i}\):** Unlike 1-order Voronoi diagram that can be computed (mostly) by only interacting with one-hop neighbors \(N(n_i)\) of a given node \(n_i\), \(N(n_i)\) may not be sufficient to obtain \(k\)-order Voronoi cells, especially when \(k\) is large. The reason is simple: at least \(k + 1\)
nodes should be involved to compute a dominating region of \( n_i \) \[26\]. Therefore, we propose Algorithm 2 to locally compute \( \mathcal{V}_n^k \) in an expanding ring manner.

**Algorithm 2: Localized \( \mathcal{V}_n^k \) Computation**

| Input: For each \( n_i \in \mathcal{N} \), initial ring radius \( \rho = 0 \)          |
| Output: \( \mathcal{V}_n^k \)                                                                 |
| 1 repeat                                                                                   |
| 2 \( \rho \leftarrow \rho + \gamma; \) out \leftarrow true                                 |
| 3 \( \mathcal{N}(n_i, \rho) \leftarrow \{ n_j \mid \| u_j - u_i \|_2 < \rho \} \)         |
| 4 Construct a local coordinate system using \( \mathcal{N}(n_i, \rho) \)                   |
| 5 foreach \( v \in \mathcal{A} \) s.t. \( \| v - n_i \|_2 = \rho/2 \) do                |
| 6 \( S^k_{n_i}(v) \leftarrow \{ n_j \in \mathcal{N}(n_i, \rho) \mid \| u_j - v \|_2 < \) |
| 7 if \( \| S^k_{n_i}(v) \| < k \) then out \leftarrow false; break                        |
| 8 end                                                                                        |
| 9 until out = true                                                                         |
| 10 Compute \( \mathcal{V}_n^k \) based on \( \mathcal{N}(n_i, \rho) \)                    |

Basically, we expand the search ring \( \rho \) with a granularity of the transmission range \( \gamma \) (line 2). As expanding \( \rho \) beyond \( \gamma \) will need multi-hop communication and the hop number is always an integer, it makes no sense to apply a smaller granularity. We use the embedding algorithm proposed in \[32\] to construct a local coordinate system (line 4). If the location information is available, this step is not necessary. Under the constructed coordinate system, we check whether the circle centered at \( u_i \) with a radius \( \rho/2 \) is not dominated by \( n_i \) anymore (lines 5 to 8, based on **Proposition 1**). Because the Voronoi edges computed given \( \mathcal{N}(n_i, \rho) \) divide the circle with radius \( \rho/2 \) into a finite number of arcs, each of which either fully dominated by \( n_i \) or not at all, we only need to check an arbitrary point on each arc in the actual implementation. The ring expending terminates if the answer becomes true. Finally, we compute \( \mathcal{V}_n^k \) using only nodes falling into the current search ring (line 10).

We need another lemma to show that \( \mathcal{V}_n^k \) computed by Algorithm 2 is indeed the one that would be computed in a centralized manner using global information.

**Lemma 1:** If the dominating region of \( n_i \) is enclosed by a circle centered at \( u_i \) with a radius of \( \rho/2 \), then it is fully determined by all the nodes located within another circle centered at \( u_i \) with a radius of \( \rho \).

**Proof:** For a disk \( \odot(u_i, \rho/2) \) centered at \( u_i \) with radius \( \rho/2 \), if \( \mathcal{V}_n^k \subset \odot(u_i, \rho/2) \), the boundary of \( \mathcal{V}_n^k \) also belongs to \( \odot(u_i, \rho/2) \). According to the definition of Voronoi cells, the cell boundary consists of bisectors, each of which is determined by two generators. For \( \mathcal{V}_n^k \), one generator is \( n_i \), and all other generators can be obtained by going through each segment (or bisectors) on the boundary of \( \mathcal{V}_n^k \) and identifying another generator that determines this bisector along with \( n_i \). Since the boundary of \( \mathcal{V}_n^k \) belongs to \( \odot(u_i, \rho/2) \), all generators of \( \mathcal{V}_n^k \) belong to \( \odot(u_i, \rho) \).

The correctness of our algorithm is immediate from this lemma: as the algorithm terminates when \( n_i \) is not dominating the circle centered at \( u_i \) with a radius \( \rho/2 \) anymore, the nodes falling into \( \odot(u_i, \rho) \) are sufficient to compute \( \mathcal{V}_n^k \). In Fig. 4, we demonstrate this sufficiency using \( k \)-order dominating region \( (k = 1 \text{ to } 12) \) in a regularly deployed WSN. The regular deployment is chosen to facilitate exposition, our algorithm works for any arbitrary deployments.

For a node \( n_i \) on the boundary, the search ring will never stop expanding, as the arc that is out of the network coverage will always need the domination of \( n_i \). To cope with this issue, we first refer to our previous work \[29\] for an on-boundary detection service. Since each sensor node can identify if it is on the network boundary only relying on local positions of it one-hop neighbors, this boundary detection is highly efficient and thus is quite suitable for our autonomous deployment. Based on the boundary awareness, the boundary node executes the circle checking procedure (lines 5 to 8) only for the arc that lies within the network coverage area, as shown in Fig. 3. Finally, the boundary node calculates its dominating region, using \( \mathcal{N}(n_i, \rho) \) as well as the searching ring to determine the boundary of the dominating region. For an initial random deployment in which nodes only occupy a small fraction of \( \mathcal{A} \), this procedure has the effect of “pushing” boundary nodes outwards, hence expanding the network coverage to the whole \( \mathcal{A} \). In fact, such a constrained checking procedure should always be used by nodes on the boundary of \( \mathcal{A} \), the difference is that \( \mathcal{A} \)’s boundary, known in advance to the nodes, serves as a natural boundary for a dominating region.

2) **Termination Analysis:** Showing the termination of Algorithm 1 appears to be highly non-trivial, as many \( k \)-order Voronoi cells are concerning a certain node, and the dominating region of a node is mostly probably a non-convex region. Fortunately, the termination can be shown by focusing on the boundary of a dominating region.

**Proposition 4:** Algorithm 1 terminates for \( \alpha \in (0, 1] \).

**Proof:** As shown in Fig. 5, \( u_i^l \) and \( \mathcal{V}_n^{k,l} \) are the location and dominating region of node \( n_i \) at the beginning of the \( l \)-th round, respectively. We also denote by \( c_i^l \) and \( R_i^l \) the Chebyshev center and the circumradius of \( \mathcal{V}_n^{k,l} \) computed by \( n_i \) during the \( l \)-round. Let \( \hat{R}_i^l = \max_{u \in \mathcal{V}_n^{k,l}} \| u - u_i^l \|_2 \) be the farthest distance from \( u_i^l \) in \( \mathcal{V}_n^{k,l} \). Finally, we define \( R_i^l = \max_i \{ R_i^l \} \) and \( \hat{R}_i^l = \max_i \{ \hat{R}_i^l \} \).

We first prove the termination for \( \alpha = 1 \) by contradiction. For each \( n_i \), we put a disk \( \odot(c_i^l, R_i^l) \) centered at \( c_i^l \) with radius \( R_i^l \). Obviously, \( \bigcup_{i=1}^{N} \odot(c_i^l, R_i^l) \) form a \( k \)-coverage over the targeted area \( \mathcal{A} \). The termination is naturally justified if we can prove that \( \mathcal{V}_n^{k,l+1} \) is inside \( \odot(c_i^l, R_i^l) \) after \( u_i^l \) is updated to \( c_i^l \).
calculated in an expanding ring manner (see Algorithm 2). The dominating regions are the regions and Chebyshev centers. The dominating regions are involved in Algorithm 2. Additionally, we illustrate multi-hop transmission range using red circles in (a). While the cases for \( k = 1 \) can be handled by involving only the 6 closest nodes (1-hop neighbors) to the central node, computing the 2-, 3-, and 4-order dominating regions requires 2-hop neighbors. When \( k > 4 \), all sensor nodes within 3 hops are involved. (a) 1-order; (b) 2-order; (c) 3-order; (d) 4-order; (e) 5-order; (f) 6-order; (g) 7-order; (h) 8-order; (i) 9-order; (j) 10-order; (k) 11-order; (l) 12-order.

(i.e., \( u_i^{l+1} = c_i^l \)). For each point \( q \) on the boundary of \( V^{k,l+1}_i \), it is straightforward to see that \( c_i^l \) is the location of the \( k \)-th nearest nodes. Assume that \( q \) is outside \( \circ(c_i^l, R_i^l) \), there would be only \( k - 1 \) disks covering \( q \), which contradicts the fact that \( \bigcup_{l=1}^{N_i} \circ(c_i^l, R_i^l) \) constitute a \( k \)-coverage over \( A \).

We then prove the termination for \( 0 < \alpha < 1 \). According to line 5 of Algorithm 1, during the \( l \)-th round, \( u_i^{l+1} = u_i^l + \alpha(c_i^l - u_i^l) \). We put disks \( \circ(u_i^l, R_i^l) \) and \( \circ(c_i^l, R_i^l) \) centered at \( u_i^l \) and \( c_i^l \), respectively. Obviously, \( V^{k,l}_{n_i,l} \) is inside \( \circ(u_i^l, R_i^l) \cap \circ(c_i^l, R_i^l) \), which implies \( V^{k,l}_{n_i,l} \subseteq \circ(u_i^{l+1}, R_i^{l+1}) \). Hence, \( \bigcup_{i=1}^{N} \circ(u_i^{l+1}, R_i^{l+1}) \) constitute a \( k \)-coverage over \( A \). Following a similar argument as for \( \alpha = 1 \), we have \( V^{k,l}_{n_i,l} \) is involved in \( \circ(u_i^{l+1}, R_i^{l+1}) \), which completes the proof.

In summary, our APOLLO algorithm terminates for any \( \alpha \in (0, 1) \). Usually, smaller \( \alpha \) leads to slower convergence but smoother moving locus. As a byproduct of the proof, we also conclude that \( R \) is non-increasing iteratively and finally equivalent to \( R \). Especially, when \( \alpha = 1 \), \( R \) is also non-increasing in the iterative process. Therefore,

**Corollary 1:** Algorithm 1 terminates at a local minimum of \( k \)-CSDP.

It is important to note that \( R_i^l \) and \( \tilde{R}_i^l \) are introduced only for our proof. During the algorithm execution, each node \( n_i \) can only compute its own \( R_i^l \) and \( \tilde{R}_i^l \). According to our earlier discussion in Section IV-B1 (see Fig. 3 also), the evolution of the node positions often takes two phases: an expanding phase and a converging phase. The expanding phase exists if the initial node distribution is non-uniform, our APOLLO algorithm will force the node to spread out during this phase, as discussed at the end of Section IV-B1. During this phase, both \( R^l \) and \( \tilde{R}^l \) are most probably achieved by boundary nodes. The expanding phase ends when all the boundary nodes are at the boundary of \( A \), this is when the converging phase starts.

3) **Computational Complexity Analysis:** Each iteration of APOLLO consists of two major steps: computing dominating regions and Chebyshev centers. The dominating regions are calculated in an expanding ring manner (see Algorithm 2).

We suppose \( N_i(\rho) = |N_i(\rho)| \) is the number of the neighboring nodes of \( n_i \) within the searching ring \( \rho \). According to [26], each sensor node computes local \( k \)-order Voronoi edges (as well as vertices) with a complexity of \( O(k^2 N_i(\rho) \log N_i(\rho)) \). Recall that \( N_i(\rho) \) leads to \( O(k N_i(\rho - k)) \) Voronoi edges [31], the following checking operation has a complexity of \( O(k N_i(\rho) (N_i(\rho) - k)) \) in the worst case. The underlying boundary detection service requires only one-hop neighbors, thus merely results in a negligible cost of \( O(N_i(\gamma) \log N_i(\gamma)) \) [29]. Assuming up to \( H \)-hop neighbors are required (i.e., \( \rho \) expands from \( \gamma \) to \( H \gamma \) with a granularity of \( \gamma \), Algorithm 2 has a complexity of \( O(k^2 H N_i(H \gamma) \log N_i(H \gamma) + k H N_i(H \gamma)(N_i(H \gamma) - k)) \) for certain coverage order \( k \), the overall complexity of Algorithm 2 is highly limited by the number of required communication hops \( H \). According to our experiments shown in Fig. 4, even the transmission range is restricted (only available for computing 1-order Voronoi dominating region), nodes within two or three hops are sufficient in most cases. Finally, given \( O(k N_i(H \gamma) - k)) \) Voronoi vertices outputted by Algorithm 2, we spend \( O(k N_i(H \gamma) - k) \) on computing their Chebyshev center [37].

The total number of iterations of APOLLO depends on individual cases, and thus cannot be derived by exiting analysis techniques. Similar with [6], [18], we will employ extensive simulations to reveal APOLLO’s convergence rate as well as the induced time cost in Section VI.
C. Discussions

In this section, we show the relations between the output of our APOLLO algorithm and other optimization objectives often considered for area coverage problems in WSNs.

Min-Node k-Coverage: One type of problem that is commonly tackled in the research community is to achieve k-coverage using a minimum number of nodes (e.g., [3], [5], [43]). As this problem often assumes that all nodes have a fixed and identical sensing range \( r_s \), it appears that APOLLO may not suggest a direct solution to it. However, we can transform our algorithm to deliver a good approximation to this min-node k-coverage problem as follows. APOLLO is called iteratively\(^3\) and \( R^* \) is compared with \( r_s \) at the end of each iteration. Nodes are added (resp. reduced) if \( R^* > r_s \) (resp. \( R^* < r_s \)), until \( R^* \leq r_s \) but adding one more node would make \( R^* > r_s \). Although the solution may not be optimal, it yields very good approximation to the optimal solution, as we will demonstrate in Section VI. If fact, as the up-to-date algorithms are all approximation to the optimal solution, as we will demonstrate.

Maximum k-Coverage: Another type of problem aims at maximizing coverage under fixed sensing ranges, but existing proposals only focus on 1-coverage [6], [18], [34]. A natural definition of the general maximum k coverage problem is to maximize the area that is k-covered under a fixed sensing range. The major difference between \( k = 1 \) and \( k > 1 \) is that the former achieves maximum coverage if nodes are far apart from each other whereas the same principle does not apply to the latter. An obvious example is the following: assume only 3 nodes are used to 3-cover an area, the maximum coverage is achieved only if all three nodes are put at the same location. Consequently, the heuristic of bounding the minimum separation among nodes [23] fails. Intuitively, APOLLO may deliver a good approximation to the maximum k-coverage problem, e.g., APOLLO terminates at the optimal solution for the aforementioned 3-node case.

Connectivity: Although maintaining network connectivity is not our concern in designing APOLLO, it appears to be a natural outcome of achieving k-coverage for \( k \geq 2 \). Under k-coverage, it is easy to see that there should be at least \( k \) nodes in the sensing range \( r_i \) of a node \( n_i \) (including \( n_i \)), otherwise \( u_i \) is not k-covered. In reality, as shown in Fig. 4, there are at least 7 nodes in a certain sensing range for \( k \geq 2 \). Given the common assumption in the literature that \( \gamma \geq r_i \) (e.g., [6], [34], [44]), each node in a WSN has at least a degree of 6, which is sufficient to guarantee connectivity in the WSN.

Min-Max Fair: While our k-CSDP only requires that the maximum sensing range is minimized and hence does not concern nodes with non-maximum sensing ranges, the min-max fair utility (a Pareto optimal point) requires that a node \( n_i \) cannot further reduce its sensing ranges \( r_i \) without increasing the sensing range \( r_j \) \( (r_j \geq r_i) \) of another node \( n_j \). According to the property of k-order Voronoi diagram, the output of APOLLO is at least locally optimal with respect to the min-max fair utility, i.e., if we reduce \( r_i \), another node \( n_j \) that shares dominating region boundary with \( n_i \) should increase \( r_j \) to maintain k-coverage, but we know \( r_j \geq r_i \) before \( r_i \) gets reduced. In fact, our simulation results in Section VI show that, after APOLLO terminates, the maximum and minimum sensing ranges are almost the same for \( k > 2 \).

V. APOLLO ON 3D SURFACES

Though replacing Euclidean metric by geodesic distance yields a straightforward extension of our problem from 2-D planes to 3-D surfaces (as we discussed in Section III-D), our APOLLO algorithm has to be slightly tuned to adapt to the local coordinate maps (i.e., the log/exp maps). As APOLLO (Algorithm 1) involves two main computations: dominating region and Chebyshev center, we present the APOLLO 3-D extension with respect to these two separately.

A. Computing Dominating Regions

By redefining the k-order Voronoi diagram based on geodesic distance. Algorithm 2 could be extended to handle computations on 3-D surfaces, while still guaranteeing the locality of the computations. After constructing a local coordinate system on the 3-D surface (see Section III-D2), each node \( n_i \) expands its searching ring \( \rho \) with a granularity of transmission range \( \gamma \), until the geodesic disk \( \mathcal{O}_i = \{ v \in \mathcal{M} | g(u_i, v) \leq \rho/2 \} \) is not dominated by \( n_i \) anymore. \( n_i \) needs only to communicate with \( \mathcal{N}(n_i, \rho) = \{ n_j | g(u_i, u_j) \leq \rho \} \) if its dominating region lies in \( \mathcal{O}_i \), according to Lemma 1. This extension is pretty straightforward; the only difference is that, while we compute \( \| u_i - u_j \|_2 \) on 2-D planes, we use the ICH algorithm to obtain \( g(u_i, u_j) \) on 3-D surfaces.

A seeming discrepancy here is that, while the sensing range is mostly determined by Euclidean metric, APOLLO operates on geodesic distance. Fortunately, based on [16], it can be derived that \( 1 \leq g(u_i, v)/\|u_i - v\|_2^2 \leq \beta \), where \( \beta \) is a constant determined by the geometric properties of a 3-D surface (i.e., its maximum Gaussian curvature). Consequently, the sensor nodes simply assign the geodesic distance \( g(u_i, v) \) (the upper bound of the Euclidean distance) to their Euclidean sensing ranges (i.e., \( r_i = g(u_i, v) \)). This leads to a feasible solution that does not compromise much of the optimality. For brevity, we omit this step in the later presentations.

B. Computing Chebyshev Centers

After determining the dominating region \( V^k_{n_i} \), the next step is for \( n_i \) to compute the Chebyshev center of its dominating region. The problem is reduced to that, given a set of points (i.e., the vertices of \( V^k_{n_i} \) in our case) on a surface, how to compute their Chebyshev center. Unfortunately, compared with its 2-D counterpart, computing Chebyshev centers on 3-D surface (i.e., under geodesic distance) appears to be highly non-trivial; it has not been addressed in the literature to the best of our knowledge. We thus propose an iterative algorithm to calculate an approximate Chebyshev center \( c_i \in \mathcal{M} \) of \( V^k_{n_i} \) using log/exp

---

\(^3\)If an application only requires a one-time (rather than autonomous) deployment, we may use APOLLO in a centralized fashion.
map (see the pseudo-codes in Algorithm 3): the basic idea is to first compute the mass center of the dominating region by iteratively applying log/exp map (lines 1–6), and then determine the Chebyshev center (lines 7–8).

Algorithm 3: Approximating a Chebyshev Center on a 3D Surface

\begin{verbatim}
Input: For each \( n_i \in X \), a local geodesic coord. system on the surface, the dominating region \( V_{n_i}^k \), stopping tolerance \( \varepsilon \)
Output: \( c_i' \)
1 Initialize the mass center of \( V_{n_i}^k \) as \( \omega_i \leftarrow u_i \)
2 repeat
3 Compute the logarithm map, \( \exp_{\omega_i}^{-1}(V_{n_i}^k) \in T_{\omega_i} \), of \( V_{n_i}^k \) at \( \omega_i \)
4 Compute the mass center \( \hat{\omega}_i \in T_{\omega_i} \) of \( \exp_{\omega_i}^{-1}(V_{n_i}^k) \)
5 if \( \|\hat{\omega}_i\|_2 > \varepsilon \) then \( \Delta \omega_i \leftarrow \delta (\exp_{\omega_i}(\hat{\omega}_i) - \omega_i) \)
6 until \( \|\hat{\omega}_i\|_2 \leq \varepsilon \)
7 \( \hat{V}_{n_i}^k \leftarrow \exp_{\omega_i}(V_{n_i}^k) \); compute the Chebyshev center, \( c_i' \), of \( \hat{V}_{n_i}^k \) in the 2D tangent plane \( T_{\omega_i} \)
8 \( c_i' \leftarrow \exp_{\omega_i}(c_i') \)
\end{verbatim}

As the difficulty in computing the Chebyshev center is the local shape distortion resulting from any 3-D-to-2-D map, we want to find a log/exp map that yields the smallest distortion within \( V_{n_i}^k \). Intuitively, the mass center of \( V_{n_i}^k \), \( \omega_i = \arg \min_u \int g^2(u, v) dv \), may yield a log/exp map that has the smallest shape distortion. Therefore, we take \( V_{n_i}^k \) to the tangent plane \( T_{\omega_i} \) by applying the logarithm map to \( V_{n_i}^k \) at \( \omega_i \), and we compute the Chebyshev center of \( \exp_{\omega_i}^{-1}(V_{n_i}^k) \) (line 7), and finally determine the Chebyshev center of \( V_{n_i}^k \) using the exponential map (line 8). As the shape distortion has been suppressed as far as possible, we believe that \( c_i' \) is a good approximation to the real Chebyshev center of \( V_{n_i}^k \).

Proposition 5: Algorithm 1 terminates with \( \{c_i'\} \) if the step size \( \alpha \) is sufficiently small.

Proof: Let \( V \) on \( M \) be contained in a geodesic ball \( B(y, r) \) centered at \( y \in M \) with radius \( r < (\pi/2 \max\{\kappa, \eta\}) \), where \( \kappa \) is the maximal Gaussian curvature of points inside \( B(y, r) \). It is proven in [20] that the function \( \Phi(\omega) = \int g^2(\omega, v) dv \) for \( \omega \in V \) is convex and achieves a unique minimum \( \omega^* \in B(y, r) \). A simple computation shows that \( \nabla \Phi(\omega^*) = 2 \int g^2(\omega^*, v) dv \). In other words, the mass center of \( V \) is uniquely defined and independent of the initial value, hence the iterative computation (lines 1–6) converges to the mass center. With the exponential map at the mass center \( \omega^* \), a tangent plane \( T_{\omega^*} \) is constructed and the Chebyshev center \( c \) is computed on that plane. This ends one round for Algorithm 1. Suppose we take a sufficiently small step size \( \alpha \), the next round for Algorithm 1 will be done on almost the same tangent plane \( T_{\omega^*} \). Therefore, from a node \( n_i \)'s point of view, the computations involved in two consecutive rounds \( l \) and \( l + 1 \) are done in 2-D. So we can apply the proving method for Proposition 4 to show \( V_{n_i}^{k+1} \subset \odot(c_i', R_i) \). As the log/exp map is a bijection within \( B(y, r) \), we have \( V_{n_i}^{k+1} \subset \odot(c_i', R_i) \), which completes the proof.

The computations of Algorithm 3 are done by individual nodes with neither communications nor motions. Since \( \Phi(\omega) \) has \( C^2 \) smoothness, the algorithm has a quadratic convergence rate, causing a negligible computational cost. \( \alpha < 1 \) almost always guarantees the overall convergence.

VI. SIMULATIONS

In this section, we report our simulation results. We first present the convergence of APOLLO. Studying the energy consumptions during and after the autonomous deployments, we also evaluate the performance of APOLLO in Min-Node \( k \)-Coverage and Maximum \( k \)-Coverage, followed by the adaptability to network irregularities. Finally, we validate the effectiveness of APOLLO 3-D extension.

A. Convergence

As convergence results we obtain from our extensive experiments are all similar, we present only two cases to demonstrate the convergence of our algorithm. We consider a targeted area of 1 km\(^2\), and initially deploy 100 sensor nodes either at the bottom-left corner (see Fig. 6(a)), or separated into two disjoint groups located at the bottom-left and upper-right corners (see Fig. 6(b)). According to the following four subfigures for both cases, our algorithm apparently leads to an “even” node distribution in the sense of multiple coverage. Specifically, in the multiple coverage cases with \( k = 2, 3, 4 \), nodes tend to cluster in groups of size \( k \), in contrast to the pure even distribution for \( k = 1 \). This is not a surprise as such an “even clustering” distribution yields more overlaps of the dominating regions among every cluster, which in turn reduces the required sensing range. Interestingly, this appears to also meet the needs of maximum \( k \)-coverage. As we discussed in Section IV-C, APOLLO leads to a co-location deployment for the extreme example of using three nodes to achieve 3-coverage. The second case also shows that two disconnected clusters will eventually merge into a concocted network. Our extensive simulations show that the initial deployment does not have significant impact on the algorithm output.

We show the convergence process of APOLLO in Fig. 7. Since a sensor node finally reaches the Chebyshev center of its dominating region and the sensing range is equivalent to the circumsphere of the dominating region, we illustrate the relationship between execution rounds (of length \( \tau \) ms each) and the maximum/minimum circumradius. As the nodes are deployed at the corner of the targeted area initially, the maximum circumcircle usually appears on the network boundary, which is mainly determined by the searching ring (Fig. 3). Consequently, the maximum circumball for \( k = 1, 2, 3, 4 \) are almost the same at the beginning. Corresponding to our proof of termination, the maximum circumradius is monotonically decreasing with the execution rounds of APOLLO, while the minimum radius is increasing in general. In the end, the maximum and minimum radii are very close to each other, especially for larger \( k \). While the monotonic decreasing in maximum circumradius shows the termination of APOLLO, the fact that the minimum circumradius coincides with the maximum one further confirms the balanced sensing load.
Fig. 6. Initial deployments and k-coverage \((k = 1, 2, 3, 4)\) deployments as the output of APOLLO. (a) Initial deployment I (b) 1-coverage I (c) 2-coverage I (d) 3-coverage I (e) 4-coverage I (f) Initial deployment II (g) 1-coverage II (h) 2-coverage II (i) 3-coverage II (j) 4-coverage II.

Fig. 7. The convergence of APOLLO.

### B. Energy Consumption During Deployments

In this section, we use TOSSIM [27] simulations driven by realistic power consumption data to evaluate the energy consumption of the whole deployment process. We assume that a mobile sensor node is equipped with a Micromo coreless DC motor (www.micromo.com/coreless-dc-motors-datasheets.aspx). It moves a MicaZ Mote in a speed of 0.2 m/s with an energy consumption of 120 mW. We get the communication cost data from the specification of the popular CC2420 radio [1]: transmit power 52.5 mW and receiving (or idle-listening) power 56.4 mW. We assume that, during the \(i\)-th round, the radios are disabled when nodes move, and the communication (and computation) session starts only when nodes have moved to their new locations \(\{u_i^+\}\) (see Algorithm 1).

Based on the same scenario studied in Section VI-A (i.e., 100 nodes on 1 km\(^2\) area), Fig. 8 demonstrates the actual energy consumption of six autonomous deployments. It is evident that a smaller step size \(\alpha\) results in more rounds but shorter total moving distances; this is shown by a decreasing communication consumption in Fig. 8(a) and an increasing motion consumption in Fig. 8(b) as functions of \(\alpha\). Therefore, given certain power consumption specifications for motion and communication, we may tune \(\alpha\) to obtain an energy efficient deployment. For our current settings, the best step size shown by Fig. 8(c) is around 0.3 to 0.7. We also pick up nodes that consume the highest energy in each WSN to illustrate the energy consumed by individual nodes in Fig. 8(d). To put these consumptions into perspective, a 2450 mAh Energizer (www.energizer.com) AA battery contains 13kJ, so the (maximum) individual node consumption of 200J only accounts for a small part of the node’s energy storage.

We also report the time cost of APOLLO in Fig. 9. As the time cost stems from both communication and motion, the general trend is similar to the total combined consumption shown by Fig. 8(c): the time cost is minimized around \(\alpha \in [0.3, 0.7]\) as well, for which APOLLO terminates within around 25 minutes and is reasonable for practical applications.

### C. Energy Consumption After Deployments

In this section, we show the sensing energy consumption after APOLLO completes the deployments. We again consider a targeted area of 1 km\(^2\), while scaling the network size from 20 to 180. As data processing and memory accessing consume most of power in sensing and their frequency depends on the covered area, we model the energy consumption in sensing to be proportional to the area of the sensing disk centered at the sensor node with a radius \(r_i\). In particular, we define the energy consumption function as \(E(r_i) = \pi r_i^2\): an increasing function of \(r_i\).

We illustrate the sensing energy consumption in terms of maximum load \(\max \{E(r_i)\}_{i=1,\ldots,N}\) and total load \(\sum_{i=1}^{N} E(r_i)\) in Fig. 10. As we deploy more sensor nodes to a given targeted area, each node takes care of less area when achieving a certain coverage. The maximum sensing load is
Fig. 8. Energy consumption during the autonomous deployments. (a) Total communication consumption (b) total motion consumption (c) total combined consumption (d) node combined consumption.

Fig. 9. Total time cost.

thus decreasing with the increasing number of nodes. Given a certain number of nodes, to achieve higher coverage degree, each sensor node is supposed to cover larger area thereby enhancing the maximum sensing load. We also observe that for \( k_1 \)-coverage and \( k_2 \)-coverage, the ratio of maximum loads between them is roughly \( k_1/k_2 \), which can be explained as follows. Since APOLLO makes the minimum sensing range very close to the maximum one, each sensor node roughly covers the same area \( k|\mathcal{A}|/N \), i.e., \( E(r_i) = k|\mathcal{A}|/N \) where \( |\mathcal{A}| \) is the area of the targeted region. Nevertheless, increasing the number of nodes does decrease the total sensing load of the WSN, shown by Fig. 10(b). Since using a less number of nodes implies a larger sensing disk for each node, this in turn yields more overlap between sensing ranges (i.e., a larger sensing redundancy), thus a higher total load.

D. Comparisons With Min-Node \( k \)-Coverage

As mentioned in Section IV-C, our APOLLO algorithm results in a good approximation to min-node \( k \)-coverage problem (where all nodes have the same sensing range and the objective is to minimize the number of nodes used to achieve \( k \)-coverage). In this section, we compare our algorithm with the deployment strategies proposed in [5] and [3], in terms of the required number of nodes guaranteeing \( k \)-coverage (\( k \geq 2 \)). As we can increase the minimum sensing range to the maximum one in the output of APOLLO without compromising coverage, we assign an identical sensing range to every node as the maximum sensing range \( R^* \) in our case.

Bai et al. have proven in [5] that, without considering boundary effect and with an identical sensing range \( r \), the optimal congruent deployment density\(^4\) for 2-coverage is \( 4\pi/3\sqrt{3} \). Given a targeted area \( \mathcal{A} \), we thus compute the minimum number of sensor nodes \( N^*_{k=2} \) for 2-coverage as:

\[
N^*_{k=2} = \frac{|\mathcal{A}|}{4\pi/3\sqrt{3}R^*} = \frac{|\mathcal{A}|}{3\sqrt{3}R^*^2},
\]

where \( |\mathcal{A}| \) to replace the area of Voronoi polygons generated by sensor nodes, which leads to an under-estimation of \( N^*_{k=2} \) due to the boundary effect. We simulate large-scale WSNs with size ranging from 1000 to 1600 in a 1 km\(^2\) targeted area. The result is shown in Table I. In general, the number of nodes deployed by APOLLO is about 15% higher than the minimum value, and it is obvious that the boundary effect is the main reason for this difference. As the boundary effect is not considered in [5], extra nodes are needed to cover the vacancies on the boundary due to the mismatch between the congruent deployment and an arbitrarily shaped targeted area. Therefore, we conclude that APOLLO actually leads to a very good approximation of the min-node 2-coverage problem.

In [3], Ammari et al. propose to decompose a targeted area into adjacent Reuleaux triangles, and nodes are deployed in the intersection areas between these triangles (so-called lens in [3]). According to their derivation, \( 6k|\mathcal{A}|/(4\pi - 3\sqrt{3})r^2 \) nodes are required to \( k \)-cover \( \mathcal{A} \) where \( k \geq 3 \) and \( r \) is the sensing range.

\(^4\)Deployment density is defined as a ratio of the area of sensing disks to the area of Voronoi polygons generated by sensor nodes [5].
Here we compare this feasible deployment with APOLLO. We deploy 180 nodes in a 1 km$^2$ area and let all nodes have the same sensing range $R^*_k$. We also compute the number of nodes deployed according to the strategy proposed in [3] as $N^*_k = \frac{6k}{(4\pi - 3\sqrt{3})} R^*_k^2$. From the results shown in Table II, it is very clear that APOLLO can use much less nodes to achieve the same level of coverage compared with [3].

### E. Performance in Maximum $k$-Coverage

In addition to the analysis in Section IV-C, we evaluate the performance of APOLLO in solving maximum $k$-coverage problem. Due to space limitation, we only illustrate the results of 4-coverage. We assume sensor nodes have fixed sensing ranges of 135 m and are deployed in a square targeted region of 1 km$^2$. We also vary the network scale from 80 to 110 with a step of 10. The 4-coverage ratio (i.e., the ratio between the 4-covered area and the whole targeted region) in each round is demonstrated in Fig. 11. It is shown the area 4-covered by the sensor nodes is increased with the execution of APOLLO, and reaches maximum when APOLLO converges. Since the sensing ranges of the sensor nodes are fixed, a network with small scale may not be able to fully 4-cover the targeted region, e.g., 80, 90 or 100 nodes in our case. By gradually deploying more sensor nodes (e.g., 110 nodes), we can have the whole targeted area 4-covered. Recall that $N = 110$ sensor nodes with a fixed sensing range of $r = 135$ m can 4-cover (hence $k = 4$) an area of up to $N(4\pi - (\sqrt{3})) r^2 / 6k = 0.9$ km$^2$. Therefore, we believe that, APOLLO provides a good approximation to the maximum $k$-coverage problem.

### F. Adaptability to Obstacles

We demonstrate the autonomous adaptability of APOLLO to arbitrarily shaped targeted area (with obstacles inside) in

---

**Table I**

<table>
<thead>
<tr>
<th>$N$</th>
<th>1000</th>
<th>1200</th>
<th>1400</th>
<th>1600</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^*_k$ (m)</td>
<td>30.35</td>
<td>27.12</td>
<td>25.23</td>
<td>23.57</td>
</tr>
<tr>
<td>$N^*_k$</td>
<td>836</td>
<td>1047</td>
<td>1210</td>
<td>1386</td>
</tr>
</tbody>
</table>

**Table II**

<table>
<thead>
<tr>
<th>$k$</th>
<th>$R^*_k$ (m)</th>
<th>$N^*_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>87.6</td>
<td>318</td>
</tr>
<tr>
<td>4</td>
<td>102.1</td>
<td>313</td>
</tr>
<tr>
<td>5</td>
<td>112.4</td>
<td>323</td>
</tr>
<tr>
<td>6</td>
<td>123.6</td>
<td>320</td>
</tr>
<tr>
<td>7</td>
<td>133.9</td>
<td>318</td>
</tr>
<tr>
<td>8</td>
<td>143.2</td>
<td>318</td>
</tr>
</tbody>
</table>

---

**Fig. 11.** 4-coverage ratio.

**Fig. 12.** The “holes” within the network region represent obstacles that mobile sensor nodes cannot move upon. Obviously, APOLLO adapts well to these irregularities and again achieves the even clustering distribution as if the area were regular.

### G. Extension to 3-D Surfaces

We apply the 3-D APOLLO extension discussed in Section V for WSN deployments on various 3-D terrain surface models. The three terrain models are approximated by 5k, 20k and 130k triangles, respectively. In Fig. 13, each row shows one terrain where we deploy the sensor nodes. Since a large-scale sensor network are usually air-dropped in the applications of terrain monitoring, we initially deploy 400, 400 and 800 mobile sensor nodes for these three terrains respectively in a random manner. Fig. 13 shows the outputs also reflect clustering distributions in the multiple coverage cases with $k = 2, 3, 4$. The similarity between Fig. 13 and Fig. 6 clearly demonstrates that our deployment algorithm designed for 2-D deployments has been successfully extended to handle 3-D surface deployments. Considering space limit, we hereby omit the illustration of the convergence process of APOLLO in 3-D deployment, as it is very similar to its 2-D counterpart.

---

**Table III**

<table>
<thead>
<tr>
<th>$k$</th>
<th>Maximum $R_k^*$ (m)</th>
<th>Minimum $R_k^*$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>110</td>
<td>94</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>160</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
<td>240</td>
</tr>
</tbody>
</table>

In order to make the numbers comparable to each other, we normalize the three surfaces such that their 2-D projections are all on a 1 km$^2$ area. A direction interpretation, by comparing Table III with Fig. 7, the differences between the maximum sensing ranges and the minimum sensing ranges in 3-D deployments are a little larger than the ones delivered by 2-D deployments. In another word, quality of APOLLO’s output
Fig. 12. Adaptability of APOLLO to arbitrarily shaped areas and obstacles as well. (a) Initial deployment I (b) 2-coverage I (c) 4-coverage I (d) 6-coverage I (e) 8-coverage I (f) Initial deployment II (g) 2-coverage II (h) 4-coverage II (i) 6-coverage II (j) 8-coverage II.

Fig. 13. $k$-coverage deployments on 3-D surfaces, $k = 2, 3, 4$. (a) Initial deployment (b) 2-coverage deployment (c) 3-coverage deployment (d) 4-coverage deployment.

Table III: The maximum and minimum sensing ranges for three surface deployments

<table>
<thead>
<tr>
<th>$k$</th>
<th>Surface 1</th>
<th></th>
<th>Surface 2</th>
<th></th>
<th>Surface 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Max range</td>
<td>56.77</td>
<td>69.35</td>
<td>81.80</td>
<td>44.94</td>
<td>54.53</td>
</tr>
<tr>
<td>Min range</td>
<td>47.06</td>
<td>62.21</td>
<td>76.97</td>
<td>37.55</td>
<td>46.93</td>
</tr>
</tbody>
</table>

In terms of load balancing is worse in 3-D than in 2-D. This is expectable as the problem becomes significantly harder to handle on 3-D surfaces. However, the results in Table III do not indicate a worse performance of APOLLO in terms of solving the $k$-coverage optimization problem, because it has never been proven that an optimal $k$-coverage solution (for $k > 2$) for either 2-D planes or 3-D surfaces can be or have to be achieved by disks with an identical radius. It is highly possible that, on 3-D surfaces, an optimal $k$-coverage solution indeed accommodates variable radii. Another reason is that, we employ an approximated Chebyshev centers in 3-D APOLLO which may lead to compromise in terms of load balancing.
VII. CONCLUSION

In this paper, we have focused on minimizing the maximum sensing range to achieve load balancing \( k \)-coverage through autonomous deployments (i.e., relying on mobile sensors nodes and the wireless communications among them). We have innovated in applying the \( k \)-order Voronoi diagram in a localized manner, and proposed APOLLO to solve the optimization problem through a distributed and localized procedure. Our approach is the first to tackle the problem of \( k \)-coverage autonomous deployment, for WSNs on both 2-D planes and 3-D surfaces. We have proven the termination of APOLLO as well as the (local) optimality of its output. We have also explained the close relations between the output of APOLLO and other commonly used optimization objectives, which provides a better understanding of optimal \( k \)-coverage deployments whose theoretical characterizations are hard to obtain under general settings. Finally, our simulation results strongly confirm our theoretical claims, as well as the adaptability of APOLLO to the irregularities of the targeted sensing regions and the effectiveness of its 3-D surface extension.

This paper currently takes into account only omnidirectional sensing model, while some types of real sensors may have certain directional features (e.g., radar or acoustic sensors). We are on the way of extending APOLLO to deal with directional sensing model.

REFERENCES

Feng Li received the B.S. and M.S. degrees in computer science from Shandong Normal University, Jinan, China, in 2007 and 2010, respectively. He is currently working toward the Ph.D. degree at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include computational geometry and its application in wireless sensor networks.

Jun Luo received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland. From 2006 to 2008, he has worked as a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. Since 2008, he has been with the School of Computer Engineering, Nanyang Technological University, Singapore, where he is currently an Assistant Professor. His research interests include wireless networking, mobile and pervasive computing, applied operations research, and network security.

Wenping Wang is currently a Professor of computer science with The University of Hong Kong, Pokfulam, Hong Kong. His main research interests include computer graphics, visualization, and geometric computing. His current research interests include mesh generation and surface modeling for architectural design. Mr. Wang serves as the Journal Associate Editor for Computer Aided Geometric Design, Computers and Graphics, IEEE Transactions on Visualization and Computer Graphics (2008–2012), Computer Graphics Forum, and IEEE Computer Graphics and Applications. He has been the Program Chair of several international conferences, including Pacific Graphics 2003, ACM Symposium on Physical and Solid Modeling (SPM 2006), and International Conference on Shape Modeling (SMI 2009), and the Conference Chair of Pacific Graphics 2012, SIAM Conference on Geometric and Physical Modeling 2013 (GD/SPM’13), and SIGGRAPH Asia 2013.

Ying He received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from the State University of New York, Stony Brook, NY, USA. He is currently an Associate Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include visual computing, with a focus on the problems that require geometric computation and analysis.