Autonomous Deployment for Load Balancing k-Surface Coverage in Sensor Networks*

Feng Li, Jun Luo, Wenping Wang, and Ying He

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 2D area coverage to 3D surface coverage. In this paper, we present a practical algorithm APOLLO (Autonomous dePlOyment for Load baLancing k-surface cOverage) 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, as well as the (local) optimality of the output. We also show that our optimization objective is closely related to other frequently considered objectives for 2D area coverage. Therefore, our practical algorithm design also contributes to the theoretical understanding of the 2D 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, 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 *area/surface coverage*:[1] it indicates the monitoring quality of WSNs. Whereas many research proposals focus on either analyzing the performance of static sensor deployments (e.g., [5], [25]) or scheduling sensor activity to retain the coverage of given deployments (e.g., [33], [45]), there exists an unfailing trend in seeking autonomous deployments assisted by mobile sensor nodes to arrive at certain predefined objectives (e.g., [6], [34]). Our proposal in this paper falls into this later trend.

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 2D area coverage to 3D surface coverage for WSNs deployed on 3D 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 pure 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 dePlOyment for Load baLancing k-surface cOverage) approach aims at balancing the sensing load (thus prolonging network lifetime) while guaranteeing

The preliminary conference version of this paper has been published by IEEE ICDCS 2012 [28], where our focus was only on 2D k-area coverage. In this paper, we generalize [28] to 3D k-surface coverage. This work is supported in part by AcRF Tier 2 Grant ARC15/11.

Feng Li, Jun Luo, and Ying He are with School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: {fli3, junluo, yhe}@ntu.edu.sg

Wenping Wang is with Department of Computer Science, The University of Hong Kong, China. E-mail: wenping@cs.hku.hk

[1] For WSNs deployed on 2D planes, we only focus on approaches concerning area coverage (e.g., [5], [25]), as opposed to the *point coverage* (e.g., [8], [14], [44]). Surface coverage is an elevation of area coverage from 2D planes to 3D surfaces (terrains), making the deployment more useful in practice (e.g., for monitoring volcanos).

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 2D 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 3D 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 2D areas and 3D surfaces.

The remaining of our paper is organized as follows. We briefly survey the closely related literature in Sec. II. We formally define our model and problem in Sec. III, in which we also review the basic mathematical tools we need in our later algorithm design. In Sec. IV, we present our APOLLO algorithm for 2D k-area coverage and analyze its performance, we also interpret our solution with respect to other optimization frameworks. We then extend APOLLO to handle 3D k-surface coverage in Sec. V. The efficacy of APOLLO is further confirmed by extensive simulation results reported in Sec. VI. We finally conclude our paper in Sec. 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_t, 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 Sec. IV-C, our objective of a k-coverage with minimax 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 3D 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 2D plane with Euclidean metric. We then show that a 2D plane can be straightforwardly generalized to a 3D 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 $\mathcal{N} = \{n_1, \cdots, n_N\}$ of sensor nodes, and $|\mathcal{N}| = N$. Let $\mathcal{U} = \{u_1, \cdots, u_N\}$ denote the locations of sensor nodes. The nodes are initially deployed arbitrarily on a 2D *targeted area* $\mathcal{A}$. 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 γ, and we denote by $\mathcal{N}(n_i) = \{n_j | \|u_i - u_j\|_2 \le \gamma, i \ne j\}$ the one-hop neighbors of n_i.

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 $v \in \mathcal{A}$ 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., $\|v - u_i\|_2 \le r_i$. 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 [30] 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 $\{v \in \mathcal{A} | \|v - u_i\|_2 \le r_i\}$.

B. Problem Formulation

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

Definition 1. *A network deployment $\{u_i, r_i\}$ is said to achieve k-coverage iff for any point $v \in \mathcal{A}$, there exist* ***at least*** *k sensor nodes covering it, or $\sum_i f(v, u_i, r_i) \ge k$.*

To allow the sensor nodes k-cover the targeted area, we divide $\mathcal{A}$ into several disjoint areas $\{\mathcal{A}_j^k\}_{j=1,2,\ldots}$, 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 $n_i(\mathcal{A}_j^k)$: it equals 1 if n_i covers $\mathcal{A}_j^k$; otherwise 0. We also denote by $\mathcal{A}_{n_i}^k$ the area covered by n_i: we have $\mathcal{A}_{n_i}^k = \bigcup_{n_i(\mathcal{A}_j^k)=1} \mathcal{A}_j^k$. The sensing range r_i of n_i is determined by the farthest point in $\mathcal{A}_{n_i}^k$ from u_i, i.e., $r_i = \max_{v \in \mathcal{A}_{n_i}^k} \|v - u_i\|_2$, so that $\mathcal{A}_{n_i}^k$ can be totally covered by n_i.

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

$$\underset{\{u_i\},\{\mathcal{A}_j^k\},\{n_i(\mathcal{A}_j^k)\}}{\text{minimize}} \quad R \tag{1}$$

$$\text{subject to} \quad \textstyle\sum_{i=1}^{N} f(v, u_i, r_i) \ge k, \ \forall v \in \mathcal{A} \tag{2}$$

$$\|v - u_i\|_2 \le R, \qquad \forall i, v \in \mathcal{A}_{n_i}^k \tag{3}$$

$$\mathcal{A}_{j_1}^k \bigcap \mathcal{A}_{j_2}^k = \emptyset, \ \ \textstyle\bigcup_j \mathcal{A}_j^k = \mathcal{A} \tag{4}$$

Literally, k-CSDP aims at determining the node locations $\{u_i\}$, the area partition $\{\mathcal{A}_j^k\}$, and the covering relations $\{n_i(\mathcal{A}_j^k)\}$, such that the targeted area $\mathcal{A}$ 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-order Voronoi diagram, the targeted area $\mathcal{A}$ is segmented into $\hat{N}^k$ disjoint areas $\{\mathcal{V}_j^k\}_{j=1,\ldots,\hat{N}^k}$,[2] each of which is associated with k closest generators (sensor nodes in our case), i.e., a subset $\mathcal{N}_j^k \subseteq \mathcal{N}$ with $|\mathcal{N}_j^k| = k$. The k-order Voronoi cell $\mathcal{V}_j^k$ is defined as

$$\mathcal{V}_j^k = \left\{ v \in \mathcal{A} \;\middle|\; \begin{array}{l} \|v - u_i\|_2 \le \|v - u_{i'}\|_2, \\ \forall\, n_i \in \mathcal{N}_j^k, \ \ n_{i'} \in \mathcal{N}/\mathcal{N}_j^k \end{array} \right\} \tag{5}$$

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

Proposition 1. *A point $v \in \mathcal{A}$ is said to belong to $\mathcal{V}_{n_i}^k$ iff there exist* ***at most*** *$k-1$ other generators such that their distance to v is less than $\|v - u_i\|_2$.*

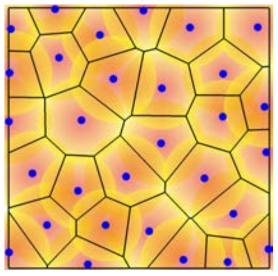
Proof: Assume $v \in \mathcal{V}_{n_i}^k$ but there were another k nodes $\{n_j\}_{j \in J, i \notin J, |J|=k}$ such that $\|u_j - v\|_2 < \|u_i - v\|_2$. Then the point v would **strictly** belong to the set of k-order Voronoi cells generated by $\{n_j\}$, which does not include n_i as a generator; a contradiction. Conversely, if there are at most $k-1$ nodes, $\{n_j\}_{j \in J, i \notin J, |J| \le k-1}$, closer to v than n_i, we can find the set of v's k-nearest nodes (which obviously contains $\{n_j\} \bigcup \{n_i\}$) to generate a set of k-order Voronoi cells containing v. As n_i is a generator, $v \in \mathcal{V}_{n_i}^k$. ■

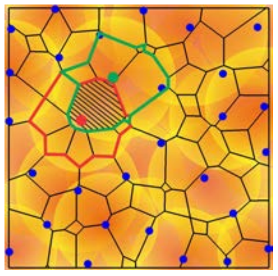
Based on the above proposition, assuming $\mathcal{S}_{n_i}^k(v) = \{n_j \in \mathcal{N} | \|u_j - v\|_2 < \|u_i - v\|_2, j \ne i\}$, we can re-define the dominating region of n_i as

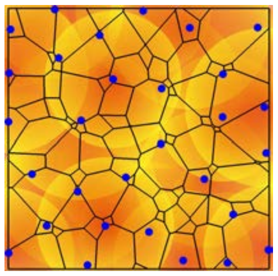
$$\mathcal{V}_{n_i}^k = \{v \in \mathcal{A} \mid |\mathcal{S}_{n_i}^k(v)| \le k-1\} \tag{6}$$

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 $\{\mathcal{V}_j^k\}$. 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.

[2]In 1-order Voronoi diagram, the number of Voronoi cells equals the number of generators (i.e., $\hat{N}^1 = N$), while in generalized k-order Voronoi diagram ($k \ge 1$), $\hat{N}^k$ is $\mathcal{O}(k(N-k))$ [31].


(a) 1-order


(b) 2-order


(c) 3-order

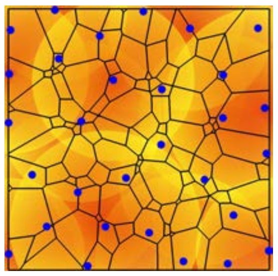

(d) 4-order

Fig. 1. k-order Voronoi partition for $k = 1, 2, 3, 4$. The disks at the backdrop represent the (overlapping) sensing ranges of individual sensor nodes.

D. Generalization to 3D Surfaces

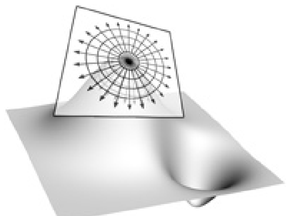
For WSN deployed on a 3D surface, the conventional Euclidean metric is no longer appropriate. Serving as the generalization of "straight line" in curved space (e.g., 3D surfaces), *geodesic* is the shortest path between two given points on the surface [9]. Therefore, geodesic distance metric is a natural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

1) Problem Description on 3D Surfaces: Consider the case where a WSN $\mathcal{N}$ is deployed on the 3D surface $\mathcal{M}$, we use geodesic distance as the distance metric. In particular, we replace all the distant metric $\|u - v\|_2$ used for 2D planes by $g(u, v)$, the geodesic distance between u and v. As a result, all the definitions in Sec. III-A to III-C can be directly migrated to 3D surface. For example, k-CSDP and high order Voronoi diagram can be redefined using geodesic metric. The only difference here is that, while $\mathcal{A}$ 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 $\mathcal{O}(n^2 \log n)$ 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 3D 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 parametrization 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 $\mathcal{T}_u$ centered at u. A *logarithm map* $\exp_u^{-1} : \mathcal{M} \to \mathcal{T}_u$ maps the points on $\mathcal{M}$ to $\mathcal{T}_u$. In particular, for any other point $v \in \mathcal{M}$ in a sufficiently small neighborhood of u, there is a unique geodesic $\mathfrak{G}_u(v)$ extending from u to v. As $\mathfrak{G}'_u(v)$, the tangent of $\mathfrak{G}_u(v)$, is obviously tangent to $\mathcal{M}$ at u, $\mathfrak{G}'_u(v) \in \mathcal{T}_u$. Therefore, v can be mapped to $\mathcal{T}_u$ 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 $\mathfrak{G}'_u(v)$ on $\mathcal{T}_u$. Consequently, the geodesic coordinates on $\mathcal{M}$ around u can be transformed to normal coordinates under any orthogonal basis $\{\mathbf{e_1}, \mathbf{e_2}\}$ 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


(a) Coordinate system in $\mathcal{T}_u$

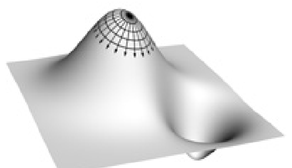

(b) Coordinate system in $\mathcal{M}$

Fig. 2. The log/exp maps at point u introduces a mapping between a vector originating at u on $\mathcal{T}_u$ and a geodesic beginning at u on $\mathcal{M}$. The circles on $\mathcal{T}_u$ are mapped to the contours of the geodesic function on $\mathcal{M}$.

to trace the geodesic $\mathfrak{G}_u(v)$ such that any point v in the neighborhood of u on $\mathcal{M}$ can get its geodesic polar coordinates on $\mathcal{T}_u$. 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 $\{\mathcal{V}_j^k\}$ generated by $\{u_i\}$ is an optimal* *partition of $\mathcal{A}$. Also, $n_i(\mathcal{V}_j^k) = 1$ if $\mathcal{V}_j^k \subseteq \mathcal{V}_{n_i}^k$; otherwise $n_i(\mathcal{V}_j^k) = 0$.*

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

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

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

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

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

Proof: As n_i needs to cover $\mathcal{A}_{n_i}^k$ 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 Sec. 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

Algorithm 1: APOLLO

```
Input: For each $n_i \in \mathcal{N}$, initial location $u_i^{(0)}$, stopping
    tolerance $\varepsilon$
Output: $\{u_i^*\}$ and $\{r_i^*\}$
1 For every node $n_i \in \mathcal{N}$ periodically (every $\tau$ ms):
2 Compute its dominating region $\mathcal{V}_{n_i}^k$
3 Compute the Chebyshev center $c_i$ of $\mathcal{V}_{n_i}^k$
4 if $\|u_i - c_i\|_2 > \epsilon$ then
5     $u_i^+ \leftarrow u_i + \alpha(c_i - u_i)$        /*$\alpha$ is the step size*/
6 end
7 $u_i^* \leftarrow c_i$,   $r_i^* \leftarrow \max_{u\in\mathcal{V}_{n_i}^k} \|u-u_i\|_2$
```

beginning of each round, the k-order Voronoi diagram is computed for the whole WSN, resulting in $\{\mathcal{V}_1^k, \cdots, \mathcal{V}_{\hat{N}^k}^k\}$ along with $\{\mathcal{V}_{n_1}^k, \cdots, \mathcal{V}_{n_N}^k\}$ (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 ε 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 ε. 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 its dominating region) that covers its dominating region. As a dominating region is a polygon, we apply Welzl's algorithm [37] to compute the Chebysehev center by taking the vertices of the region as the input.

Similar algorithms have been applied in [6], [34], but they were used to indirectly optimize a different objective (see our discussions in Sec. 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 $\mathcal{V}_{n_i}^k$ 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 $\mathcal{V}_{n_i}^k$: Unlike 1-order Voronoi diagram that can be computed (mostly) by only interacting with one-hop neighbors $\mathcal{N}(n_i)$ of a given node n_i, $\mathcal{N}(n_i)$ 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 calculate $\mathcal{V}_{n_i}^k$ in an expanding ring manner.

Algorithm 2: Localized $\mathcal{V}_{n_i}^k$ Computation

```
Input: For each $n_i \in \mathcal{N}$, initial ring radius $\rho = 0$
Output: $\mathcal{V}_{n_i}^k$
1  repeat
2      $\rho \leftarrow \rho + \gamma$;   $out \leftarrow$ true
3      $\mathcal{N}(n_i, \rho) \leftarrow \{n_j | \|u_j - u_i\|_2 < \rho\}$
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          $\hat{\mathcal{S}}_{n_i}^k(v) \leftarrow \{n_j \in \mathcal{N}(n_i,\rho) | \|u_j - v\|_2 <$
           $\|u_i - v\|_2, j \neq i\}$
7          if $|\hat{\mathcal{S}}_{n_i}^k(v)| < k$ then  $out \leftarrow$ false;  break
8      end
9  until $out =$ true;
10 Compute $\mathcal{V}_{n_i}^k$ based on $\mathcal{N}(n_i, \rho)$
```

Basically, we expand the search ring ρ with a granularity of the transmission range γ (line 2). As expending ρ beyond γ 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 $\mathcal{V}^k_{n_i}$ using only nodes falling into the current search ring (line 10).

We need another lemma to show that $\mathcal{V}^k_{n_i}$ 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 ρ.*

Proof: For a disk $\odot(u_i,\rho/2)$ centered at u_i with radius $\rho/2$, if $\mathcal{V}^k_{n_i} \subset \odot(u_i,\rho/2)$, the boundary of $\mathcal{V}^k_{n_i}$ 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 $\mathcal{V}^k_{n_i}$, one generator is n_i, and all other generators can be obtained by going through each segment (or bisectors) on the boundary of $\mathcal{V}^k_{n_i}$ and identifying another generator that determines this bisector along with n_i. Since the boundary of $\mathcal{V}^k_{n_i}$ belongs to $\odot(u_i,\rho/2)$, all generators of $\mathcal{V}^k_{n_i}$ belong to $\odot(u_i,\rho)$. ■

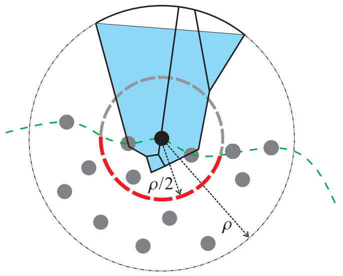


Fig. 3. Recognizing the network boundary (the green dashed curve), a boundary node (dark) determines a searching ring (the outer circle) and check the half-radius arc within the network coverage area (the inner red arc). It then calculates its dominating region (the blue area) with $\mathcal{N}(n_i,\rho)$, while the searching ring helps to determine part of the boundary.

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 $\mathcal{V}^k_{n_i}$. In Fig. 4, we demonstrate this sufficiency using k-order dominating region ($k=1$ 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-line boundary detection service. Sine 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^l_i and $\mathcal{V}^{k,l}_{n_i}$ are the location and dominating region of node n_i at the beginning of the l-th round, respectively. We also denote by c^l_i and R^l_i the Chebyshev center and the circumradius of $\mathcal{V}^{k,l}_{n_i}$ computed

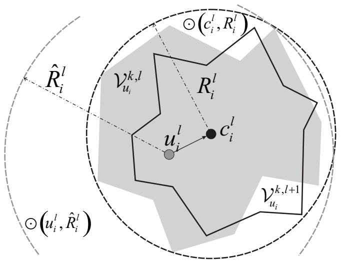
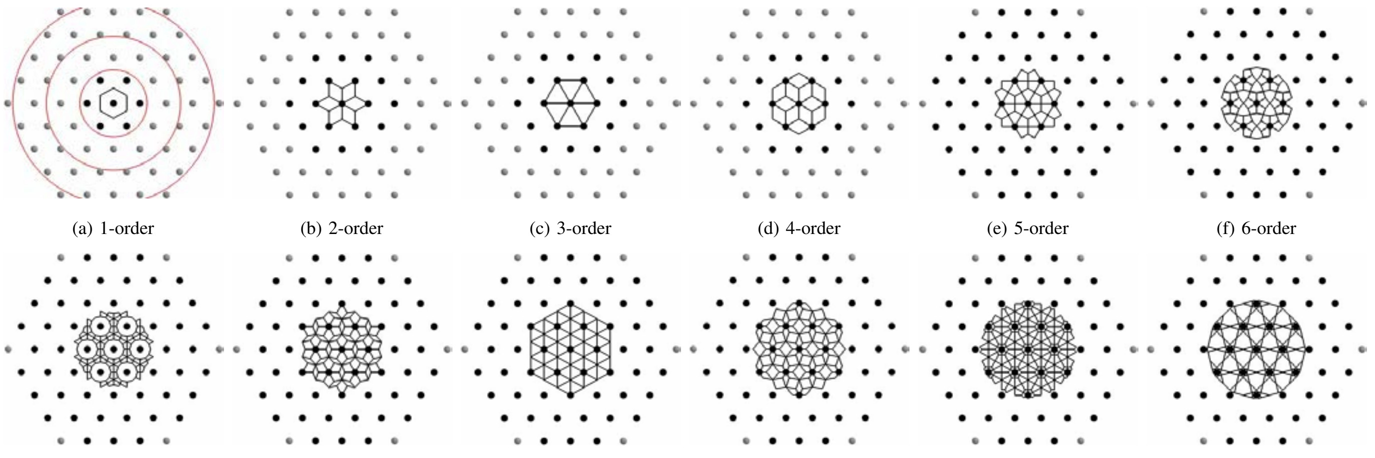

$\odot(c_i^l, R_i^l)$
$\hat{R}_i^l$
$\mathcal{V}_{u_i}^{k,l}$
R_i^l
u_i^l
c_i^l
$\mathcal{V}_{u_i}^{k,l+1}$
$\odot(u_i^l, \hat{R}_i^l)$

Fig. 5. Notations in the proof for Proposition 4.

by n_i during the l-round. Let $\hat{R}^l_i = \max_{u\in\mathcal{V}^{k,l}_{n_i}} \|u-u^l_i\|_2$ be the farthest distance from u^l_i in $\mathcal{V}^{k,l}_{n_i}$. Finally, we define $R^l = \max_i\{R^l_i\}$ and $\hat{R}^l = \max_i\{\hat{R}^l_i\}$.

We first prove the termination for $\alpha = 1$ by contradiction. For each n_i, we put a disk $\odot(c^l_i, R^l)$ centered at c^l_i with radius R^l. Obviously, $\bigcup_{i=1}^N \odot(c^l_i, R^l)$ form a k-coverage over the targeted area $\mathcal{A}$. The termination is naturally justified if we can prove that $\mathcal{V}^{k,l+1}_{n_i}$ is inside $\odot(c^l_i, R^l)$ after u^l_i is updated to c^l_i (i.e., $u^{l+1}_i = c^l_i$). For each point q on the boundary of $\mathcal{V}^{k,l+1}_{n_i}$, it is straightforward to see that c^l_i is the location of the k-th nearest nodes. Assume that q is outside $\odot(c^l_i, R^l)$, there would be only $k-1$ disks covering q, which contradicts the fact that $\bigcup_{i=1}^N \odot(c^l_i, R^l)$ constitute a k-coverage over $\mathcal{A}$.

We then prove the termination for $0 < \alpha < 1$. According to line 5 of **Algorithm 1**, during the l-th round, $u^{l+1}_i = u^l_i + \alpha(c^l_i - u^l_i)$. We put disks $\odot(u^l_i, \hat{R}^l)$ and $\odot(c^l_i, \hat{R}^l)$ centered at u^l_i and c^l_i, respectively. Obviously, $\mathcal{V}^{k,l}_{n_i}$ is inside $\odot(u^l_i, \hat{R}^l) \bigcap \odot(c^l_i, \hat{R}^l)$, which implies $\mathcal{V}^{k,l}_{n_i} \subset \odot(u^{l+1}_i, \hat{R}^l)$. Hence, $\bigcup_{i=1}^N \odot(u^{l+1}_i, \hat{R}^l)$ constitute a k-coverage over $\mathcal{A}$.



(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

Fig. 4. The dominating region of the central node in k-order Voronoi diagram $k = 1, \cdots, 12$. The central node needs to collect location (or range) information from its neighboring nodes (the dark nodes) via multi-hop communication according to **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.

Following a similar argument as for $\alpha = 1$, we have $\mathcal{V}_{n_i}^{k,l+1}$ is involved in $\odot(u_i^{l+1}, \hat{R}^l)$, which completes the proof. ■

In summary, our APOLLO algorithm terminates for any $\alpha \in (0, 1]$. Usually, smaller α leads to slower convergence but smoother moving locus. As a byproduct of the proof, we also conclude that $\hat{R}$ 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^l and $\hat{R}^l$ are introduced only for our proof. During the algorithm execution, each node n_i can only compute its own R_i^l and $\hat{R}_i^l$. According to our earlier discussion in Sec. 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 Sec. IV-B1. During this phase, both R^l and $\hat{R}^l$ are most probably achieved by boundary nodes. The expanding phase ends when all the boundary nodes are at the boundary of $\mathcal{A}$, 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 $N_i(\rho) = |\mathcal{N}(n_i, \rho)|$ is the number of the neighboring nodes of n_i within the searching ring ρ. According to [26], each sensor node computes local k-order Voronoi edges (as well as vertices) with a complexity of $\mathcal{O}(k^2 N_i(\rho) \log N_i(\rho))$. Recall that $N_i(\rho)$ leads to $\mathcal{O}(k(N_i(\rho) - k))$ Voronoi edges [31], the following checking operation has a complexity of $\mathcal{O}(kN_i(\rho)(N_i(\rho) - k))$ in the worst case. The underlying boundary detection service requires only one-hop neighbors, thus merely results in a negligible cost of $\mathcal{O}(N_i(\gamma) \log N_i(\gamma))$ [29]. Assuming up to H-hop neighbors are required (i.e., ρ expands from γ to $H\gamma$ with a granularity of γ), **Algorithm 2** has a complexity of $\mathcal{O}(k^2 H N_i(H\gamma) \log N_i(H\gamma) + kHN_i(H\gamma)(N_i(H\gamma) - k))$. 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 $\mathcal{O}(k(N_i(H\gamma) - k))$ Voronoi vertices outputted by **Algorithm 2**, we spend $\mathcal{O}(k(N_i(H\gamma) - k))$ 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 Sec. 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 Sec. VI. If fact, as the

[3]If an application only requires a one-time (rather than autonomous) deployment, we may use APOLLO in a centralized fashion.

up-to-date algorithms are all approximations for $k > 2$ and they are not autonomous (e.g., [3]), our algorithm is also the first autonomous deployment for approximating min-node k-coverage with an arbitrary k.

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 Sec. 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 2D planes to 3D surfaces (as we discussed in Sec. 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 3D 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 3D surfaces, while still guaranteeing the locality of the computations. After constructing a local coordinate system on the 3D surface (see Sec. III-D2), each node n_i expands its searching ring ρ with a granularity of transmission range γ, until the geodesic disk $\odot_i = \{v \in \mathcal{M} | g(u_i, v) \leq \rho/2\}$ 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 $\odot_i$, according to **Lemma 1**. This extension is pretty straightforward; the only difference is that, while we compute $\|u_i - u_j\|_2$ on 2D planes, we use the ICH algorithm to obtain $g(u_i, u_j)$ on 3D 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 $1 \leq \frac{g(u_i,v)}{\|u_i - v\|_2} \leq \beta$, where β is a constant determined by the geometric properties of a 3D 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 $\mathcal{V}^k_{n_i}$, 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 $\mathcal{V}^k_{n_i}$ in our case) on a surface, how to compute their Chebyshev center. Unfortunately, compared with its 2D counterpart, computing Chebyshev centers on 3D surface (i.e., under geodesic destance) 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 $\mathcal{V}^k_{n_i}$ using log/exp 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

```
  Input: For each $n_i \in \mathcal{N}$, a local geodesic coord. system
         on the surface, the dominating region $\mathcal{V}^k_{n_i}$,
         stopping tolerance $\varepsilon$
  Output: $c'_i$
1 Initialize the mass center of $\mathcal{V}^k_{n_i}$ as $\omega_i \leftarrow u_i$
2 repeat
3     Compute the logarithm map, $\exp^{-1}_{\omega_i}(\mathcal{V}^k_{n_i}) \in \mathcal{T}_{\omega_i}$, of
      $\mathcal{V}^k_{n_i}$ at $\omega_i$
4     Compute the mass center $\tilde{\omega}_i \in \mathcal{T}_{\omega_i}$ of $\exp^{-1}_{\omega_i}(\mathcal{V}^k_{n_i})$
5     if $\|\tilde{\omega}_i\|_2 > \epsilon$ then $\Delta\omega_i \leftarrow \delta \left[\exp_{\omega_i}(\tilde{\omega}_i) - \omega_i\right]$
6 until $\|\tilde{\omega}_i\|_2 \leq \epsilon$;
7 $\tilde{\mathcal{V}}^k_{n_i} \leftarrow \exp^{-1}_{\omega_i}(\mathcal{V}^k_{n_i})$; compute the Chebyshev center, $\tilde{c}_i$,
  of $\tilde{\mathcal{V}}^k_{n_i}$ in the 2D tangent plane $\mathcal{T}_{\omega_i}$
8 $c'_i \leftarrow \exp_{\omega_i}(\tilde{c}_i)$
```

As the difficulty in computing the Chebyshev center is the local shape distortion resulting from any 3D-to-2D map, we want to find a log/exp map that yields the smallest distortion within $\mathcal{V}^k_{n_i}$. Intuitively, the mass center of $\mathcal{V}^k_{n_i}$, $\omega_i = \arg\min_u \int_{\mathcal{V}^k_{n_i}} g^2(u,v)dv$, may yield a log/exp map that has the smallest shape distortion. Therefore, we take $\mathcal{V}^k_{n_i}$ to the tangent plane $\mathcal{T}_{\omega_i}$ by applying the logarithm map to $\mathcal{V}^k_{n_i}$ at ω_i, and we compute the Chebyshev center of $\exp^{-1}_{\omega_i}(\mathcal{V}^k_{n_i})$ (line 7), and we finally determine the Chebyshev center of $\mathcal{V}^k_{n_i}$ 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 $\mathcal{V}^k_{n_i}$. Here $\delta\left[\exp_{\omega_i}(\tilde{\omega}_i) - \omega_i\right]$ in line 5 is computed with respect to a geodesic from $\exp_{\omega_i}(\tilde{\omega}_i)$ to ω_i on the 3D surface.

The question is whether replacing $\{c_i\}$ by $\{c'_i\}$ in **Algorithm 1** still allows terminate. Fortunately, we have

Proposition 5. ***Algorithm 1*** *terminates with* $\{c'_i\}$ *if the step size* α *is sufficiently small.*

Proof: Let $\mathcal{V}$ on $\mathcal{M}$ be contained in a geodesic ball $B(y,r)$ centered at $y \in \mathcal{M}$ with radius $r < \frac{\pi}{2\max\{0,\kappa\}}$, where κ is the maximal Gaussian curvature of points inside $B(y,r)$. It is proven in [20] that the function $\Phi(\omega) = \int_{\mathcal{V}} g^2(\omega,v)dv$ for $\omega \in \mathcal{V}$ is convex and achieves a unique minimum $\omega^* \in B(y,r)$. A simple computation shows that $0 = \nabla\Phi(\omega^*) = 2\int_{\mathcal{V}} \exp^{-1}_{\omega^*}(v)dv$. In other words, the mass center of $\mathcal{V}$ 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 ω^*, a tangent plane $\mathcal{T}_{\omega^*}$ is constructed and the Chebyshev center $\tilde{c}$ is computed on that plane. This ends one round for **Algorithm 1**. Suppose we take a sufficiently small step size α, the next round for **Algorithm 1** will be done on almost the same tangent plane $\mathcal{T}_{\omega^*}$. Therefore, from a node n_i's point of view, the computations involved in two consecutive rounds l and $l+1$ are done in 2D. So we can apply the proving method for **Proposition 4** to show $\tilde{\mathcal{V}}^{k,l+1}_{n_i} \subset \tilde{\odot}(c^l_i, R^l)$. As the log/exp map is a bijection within $B(y,r)$, we have $\mathcal{V}^{k,l+1}_{n_i} \subset \odot(c^l_i, R^l)$, 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 3D extension.

A. Convergence

As convergence results we obtain from our extensive experiments are all similar, we present only two case 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(f)). According to the following four sub-figures 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 Sec. 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

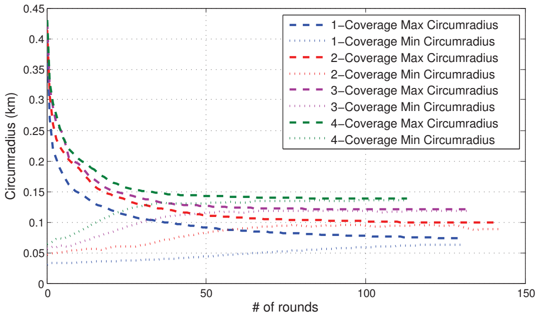
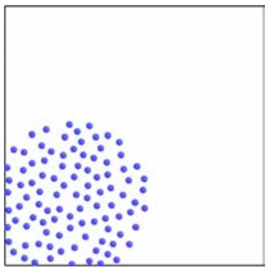


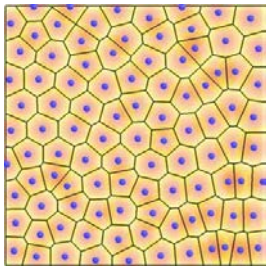
Fig. 7. The convergence of APOLLO.

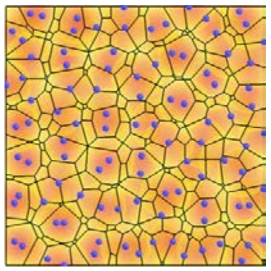
its dominating region and the sensing range is equivalent to the circumradius of the dominating region, we illustrate the relationship between execution rounds (of length τ ms each) and the maximum/minimum circumradii. As the nodes are deployed at the corner of the targeted area initially, the maximum circumcicle usually appears on the network boundary, which is mainly determined by the searching ring (Fig. 3). Consequently, the maximum circumradii 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.

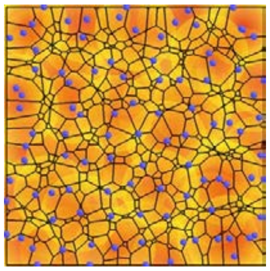
B. Energy Consumption during Deployments

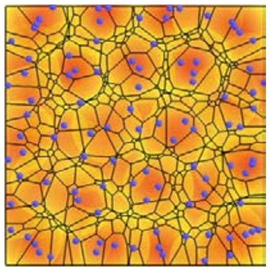
In this section, we use TOSSIM [27] simulations driven by realistic power consumption data to evaluate the energy consumption of the whole deployment process. We assume that a mobile sensor node is equipped with a Micromo coreless D-C motor (www.micromo.com/coreless-dc-motors-data-sheets.aspx). It moves 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)

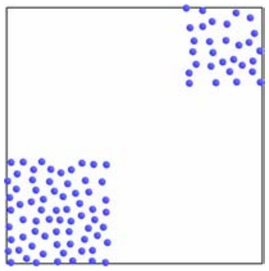

(a) Initial deployment I

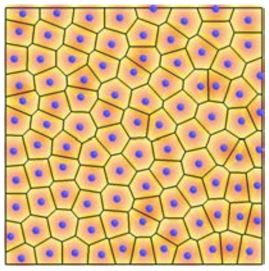

(b) 1-coverage I

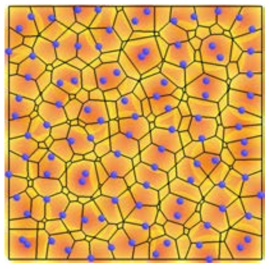

(c) 2-coverage I

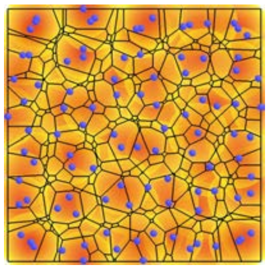

(d) 3-coverage I


(e) 4-coverage I


(f) Initial deployment II


(g) 1-coverage II


(h) 2-coverage II


(i) 3-coverage II

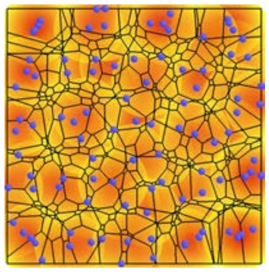
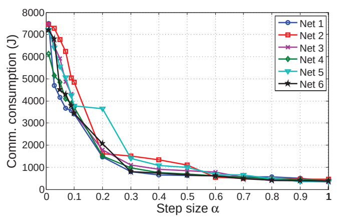
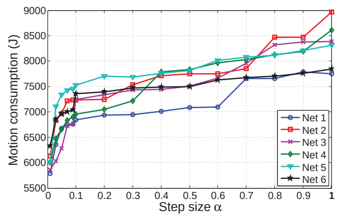

(j) 4-coverage II

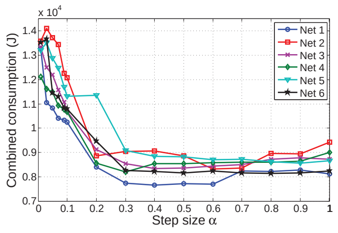
Fig. 6. Initial deployments and k-coverage ($k = 1, 2, 3, 4$) deployments as the output of APOLLO.



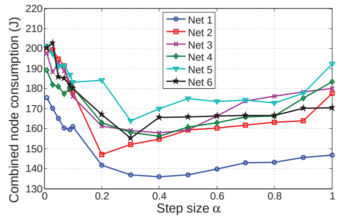
(a) Total communication consumption



(b) Total motion consumption



(c) Total combined consumption



(d) Node combined consumption

Fig. 8. Energy consumption during the autonomous deployments

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 Sec. 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 α 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 α. Therefore, given certain power consumption specifications for motion and communication, we may tune α 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 2450mAh Energizer (www.energizer.com) AA battery contains 13kJ, so the (maximum) individual node consumption of 200J only accounts from 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.

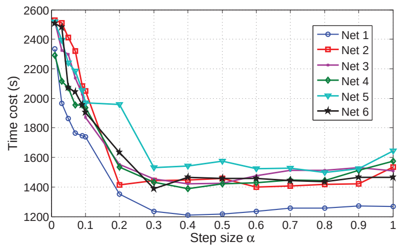
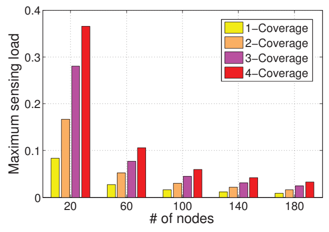


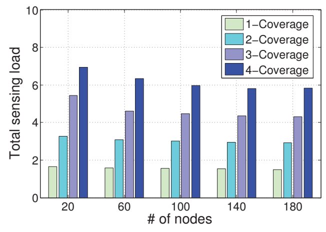
Fig. 9. Total time cost.

C. Energy Consumption after Deployments

In this section, we show the sensing energy consumption after APOLLO completes the deployments. We again consider



(a) Maximum sensing load



(b) Total sensing load

Fig. 10. Energy consumption in the final deployments of 100 nodes

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 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 redundance), thus a higher total load.

D. *Comparisons with Min-Node k-Coverage*

As mentioned in Sec. 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: $N^*_{k=2} = \frac{|\mathcal{A}|\frac{4\pi}{3\sqrt{3}}}{\pi r^2} = \frac{4|\mathcal{A}|}{3\sqrt{3}R^{*2}}$, here we use $|\mathcal{A}|$ 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.

[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].

TABLE I
THE MINIMUM NUMBER OF SENSOR NODES TO ACHIEVE 2-COVERAGE

N	1000	1200	1400	1600
R^* (m)	30.35	27.12	25.23	23.57
$N^*_{k=2}$	836	1047	1210	1386

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, $\frac{6k|\mathcal{A}|}{(4\pi-3\sqrt{3})r^2}$ nodes are required to k-cover $\mathcal{A}$ where $k \geq 3$ and r is the sensing range. 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 compte the number of nodes that deployed according to the strategy proposed in [3] as $N^*_k = \frac{6k}{(4\pi-3\sqrt{3})R^{*2}_k}$. 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].

TABLE II
THE NUMBER OF SENSOR NODES TO ACHIEVE k-COVERAGE WITH THE STRATEGY PROPOSED IN [3] FOR $k = 3, 4, ..., 8$

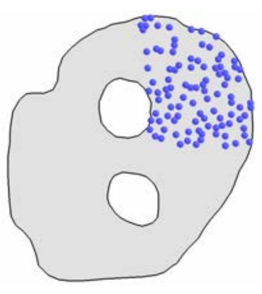
k	3	4	5	6	7	8
R^*_k (m)	87.6	102.1	112.4	123.6	133.9	143.2
N^*_k	318	313	323	320	318	318

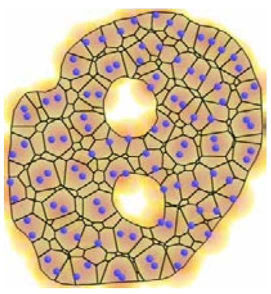
E. *Performance in Maximum k-Coverage*

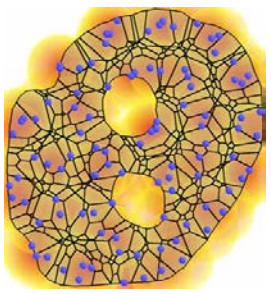
In addition to the analysis in Sec. IV-C, we evaluate the performance of APOLLO in solving maximum k-coverage problem. Due to space limitation, we only illustrate the results of 4-coverage. We assume sensor nodes have fixed sensing ranges of 135m and are deployed in a square targeted region of 1 km^2. 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 $\frac{N(4\pi-\sqrt{(3)})r^2}{6k} = 0.9$km^2. Therefore, we believe that, APOLLO provides a good approximation to the maximum k-coverage problem.

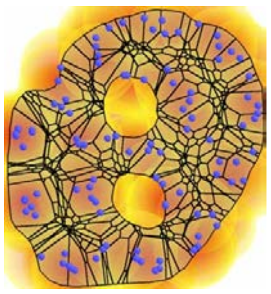
F. *Adaptability to Obstacles*

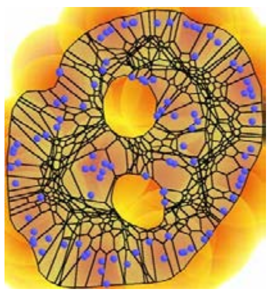
We demonstrate the autonomous adaptability of APOLLO to arbitrarily shaped targeted area (with obstacles inside) in

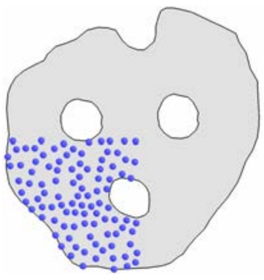

(a) Initial deployment I

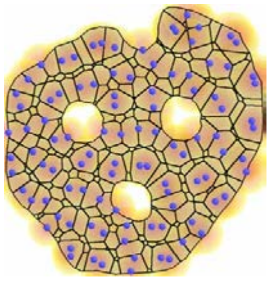

(b) 2-coverage I

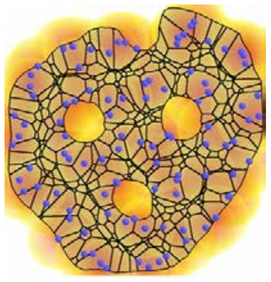

(c) 4-coverage I

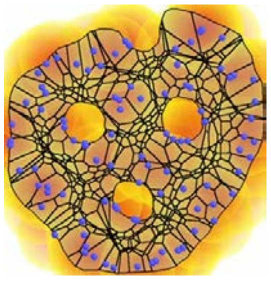

(d) 6-coverage I


(e) 8-coverage I


(f) Initial deployment II


(g) 2-coverage II


(h) 4-coverage II


(i) 6-coverage II

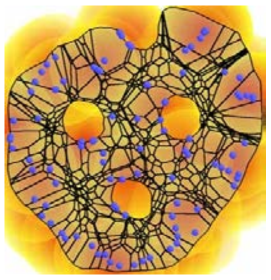

(j) 8-coverage II

Fig. 12. Adaptability of APOLLO to arbitrarily shaped areas and obstacles as well.

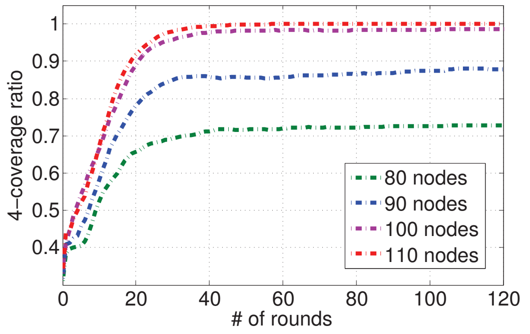


Fig. 11. 4-coverage ratio.

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 3D Surfaces

We apply the 3D APOLLO extension discussed in Sec. V for WSN deployments on various 3D terrain surface models. The three terrain models are approximated by 5k, 20k and 130k triangles, respectively. In Fig. 13, each row shows one terrian 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 2D deployments has been successfully extended to handle 3D surface deployments. Considering space limit, we hereby omit the illustration of the convergence process of APOLLO in 3D deployment, as it is very similar to its 2D counterpart.

In Table III, we use the maximum and minimum (Euclidean) sensing ranges resulted from the autonomous deployments to show the quality (in terms of load balancing) of the coverage. In order to make the numbers comparable to each other, we normalize the three surfaces such that their 2D projections are all on a 1 km^2 area. A direction interpretation,

TABLE III
THE MAXIMUM AND MINIMUM SENSING RANGES FOR THREE SURFACE DEPLOYMENTS

	Surface 1			Surface 2			Surface 3		
k	2	3	4	2	3	4	2	3	4
Max range	56.77	69.35	81.80	44.94	54.53	55.65	35.56	38.09	47.46
Min range	47.06	62.21	76.97	37.55	46.93	53.32	29.49	31.69	41.45

by comparing Table III with Fig. 7, the differences between the maximum sensing ranges and the minimum sensing ranges in 3D deployments are a little larger than the ones delivered by 2D deployments. In another word, quality of APOLLOs output in terms of load balancing is worse in 3D than in 2D. This is expectable as the problem becomes significantly harder to handle on 3D 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 2D planes or 3D surfaces can be or have to be achieved by disks with an identical radius. It is highly possible that, on 3D surfaces, an optimal k-coverage solution indeed accommodates variable radii. Another reason is that, we employ an approximated Chebyshev centers in 3D 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 n-odes and the wireless communications among them). We have

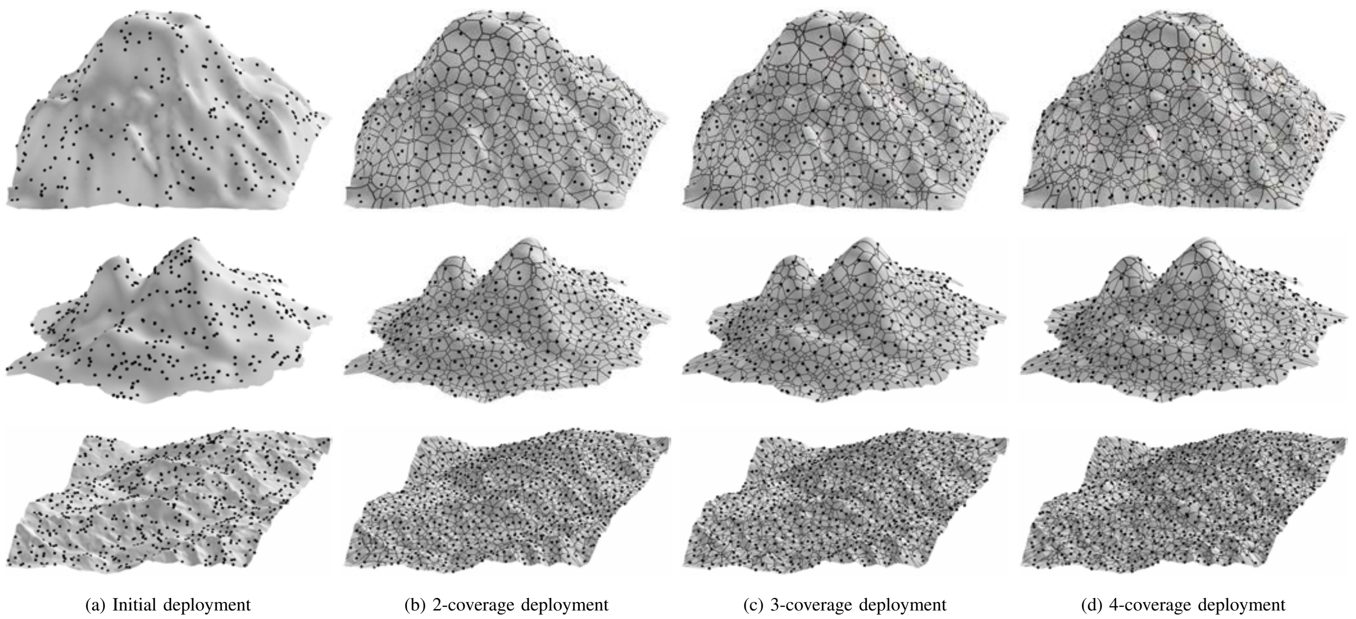

(a) Initial deployment (b) 2-coverage deployment (c) 3-coverage deployment (d) 4-coverage deployment

Fig. 13. k-coverage deployments on 3D surfaces, $k = 2, 3, 4$.

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 2D planes and 3D 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 3D 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

[1] Chipcon's CC2420 2.4G IEEE 802.15.4/ZigBee-ready RF Transceiver.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communication Mag.*, 40(8):104–112, 2002.

[3] H.M. Ammari and S.K. Das. Centralized and Clustered k-Coverage Protocols for Wireless Sensor Networks. *IEEE Trans. on Computers*, 6(1):118–133, 2012.

[4] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai. Deploying Wireless Sensors to Achieve Both Coverage and Connecitvity. In *Proc. of the 7th ACM MobiHoc*, pages 131–142, 2006.

[5] X. Bai, Z. Yun, D. Xuan, B. Chen, and W. Zhao. Optimal Multiple-Coverage of Sensor Networks. In *Proc. of the 30th IEEE INFOCOM*, pages 2498–2506, 2011.

[6] N. Bartolini, T. Calamoneri, T.F. La Porta, and S. Silvestri. Autonomous Deployment of Heterogeneous Mobile Sensors. *IEEE Trans. on Mobile Computing*, 10(6):753–766, 2011.

[7] B.A. Bash and Pe.J. Desnoyers. Exact Distributed Voronoi Cell Computation in Sensor Networks. In *Proc. of the 6th ACM/IEEE IPSN*, pages 236–243, 2007.

[8] M. Cardei, M.T. Thai, Y. Li, and W. Wu. Energy-Efficient Target Coverage in Wireless Sensor Networks. In *Prof. of the 24th IEEE INFOCOM*, pages 1976–1984, 2005.

[9] M.P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[10] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S. Sukhatme. Robomote: Enabling Mobility in Sensor Networks. In *Proc. of the 4th ACM/IEEE IPSN*, 404-409.

[11] L. Ding, W. Wu, J. Willson, L. Wu, Z. Lu, and W. Lee. Constant-Approximation for Target Coverage Problem in Wireless Sensor Networks. In *Proc. of the 31st IEEE INFOCOM*, pages 1584–1592, 2012.

[12] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithm. *SIAM Review*, 41(4):637–676, 1999.

[13] D. L. Hall and J. Llinas. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[14] K. Han, L. Xiang, J. Luo, and Y. Liu. Minimum-Energy Connected Coverage in Wireless Sensor Networks with Omni-Directional and Directional Features. In *Proc. of the 13rd ACM MobiHoc*, pages 1–10, 2012.

[15] M. Hefeeda and M. Bagheri. Randomized k-Coverage Algorithms For Dense Sensor Networks. In *Proc of the 26th IEEE INFOCOM*, pages 2376–2380, 2007.

[16] K. Hildebrandt, K. Polthier, and M. Wardetzky. On the convergence of metric and geometric properties of polyhedral surfaces. *Geometriae Dedicata*, 123(1):89–112, 2006.

[17] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete Surface Ricci Flow. *IEEE Trans. on Visualization and Computer Graphics*, 15(5):1030–1043, 2008.

[18] M. Jin, G. Rong, H. Wu, L. Shuai, and X. Guo. Optimal Surface Deployment Problem in Wireless Sensor Networks. In *Proc. of the 31st IEEE INFOCOM*, pages 2345–2353, 2012.

[19] K. Kar and S. Banerjee. Node Placement for Connected Coverage in Sensor Networks. In *Proc. of the 1st IEEE/ACM WiOpt*, 2003.

[20] H. Karcher. Riemannian Center of Mass and Mollifier Smoothing. *Communications on Pure and Applied Mathematics.*, 30(5):509–541, 1977.

[21] G. Kasbekar, Y. Bejerano, and S. Sarkar. Lifetime and Coverage Guarantees Through Distributed Coordinate-Free Sensor Activity. *IEEE/ACM Trans. on Networking*, 19(22):470–483, 2011.

[22] R. Kershner. The Number of Circles Covering A Set. *American J. Math.*, 61(3):665–671, 1939.

[23] J.-E. Kim, J. Han, and C.-G. Lee. Optimal 3-Coverage with Minimum Separation Requirements for Ubiquitous Computing Environments. *Springer Mobile Netw. & Appl.*, 14(5):556–570, 2008.

[24] L. Kong, M. Zhao, X. Liu, J. Lu, Y. Liu, M. Wu, and W. Shu. Surface Coverage in Sensor Networks. *IEEE Trans. on Parallel and Distributed Systems*, 25(1):234–243, 2014.

[25] S. Kumar, T.H. La, and J. Balogh. On k-coverage in a Mostly Sleeping Sensor Network. In *Proc. of the 10th ACM MobiCom*, pages 144–158, 2004.

[26] D.T. Lee. On k-Nearest Neighbor Voronoi Diagrams in a Plane. *IEEE Trans. on Computer*, 31(6):478–487, 1982.

[27] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. of the 1st ACM SenSys*, pages 126–137, 2003.

[28] F. Li, J. Luo, S. Xin, W. Wang, and Y. He. LAACAD: Load bAlancing k-Area Coverage through Autonomous Deployment in Wireless Sensor Networks. In *Proc. of the 32nd IEEE ICDCS*, pages 1–10, 2012.

[29] F. Li, C. Zhang, J. Luo, S. Xin, and Y. He. LBDP: Localized Boundary Detection and Parametrization for 3D Sensor Networks. *IEEE/ACM Trans. on Networking*, 22(2):567–579, 2014.

[30] J. McLurkin and E. Demaine. A Distributed Boundary Detection Algorithm for Multi-robot Systems. In *Proc. of IEEE/RSJ IROS*, pages 4791–4798, 2009.

[31] M.I. Shamos and D. Hoey. Closest-Point Problems. In *Proc. of the 16th IEEE FOCS*, pages 151–162, 1975.

[32] Y. Shang and W. Ruml. Improved MDS-Based Localization. In *Proc. of the 23rd IEEE INFOCOM*, pages 2640–2651, 2004.

[33] S. Tang, X. Li, X. Shen, J. Zhang, G. Dai, and S.K. Das. Cool: On Coverage with Solar-Powered Sensors. In *Proc. of the 31st IEEE ICDCS*, pages 488–496, 2011.

[34] G. Wang, G. Cao, and T.F. La Porta. Movement-Assisted Sensor Deployment. *IEEE Trans. on Mobile Computing*, 5(6):640–652, 2006.

[35] X. Wang, X. Wang, and J. Zhao. Impact of Mobility and Heterogeneity on Coverage and Energy Consumption in Wireless Sensor Networks. In *Proc. of the 31st IEEE ICDCS*, pages 477–487, 2011.

[36] Y.-C. Wang and Y.-C. Tseng. Distributed Deployment Schemes for Mobile Wireless Sensor Networks to Ensure Multilevel Coverage. *IEEE Trans. on Parallel and Distributed Systems*, 19(9):1280–1294, 2008.

[37] E. Welzl. Smallest Enclosing Disks (Balls and Ellipsoids). *Results and New Trends in Computer Science (LNCS 555)*, pages 359–370, 1991.

[38] L. Wu, H. Du, W. Wu, D. Li, J. Lv, and W. Lee. Approximations for Minimum Connected Sensor Cover. In *Proc. of the 32nd IEEE INFOCOM*, pages 1187–1194, 2013.

[39] S. Xin and G. Wang. Improving Chen & Han's Algorithm on the Discrete Geodesic Problem. *ACM Trans. on Graphics.*, 28(4):1–8, 2009.

[40] S. Xin and G. Wang. Applying the Improved Chen and Hans Algorithm to Different Versions of Shortest Path Problems on a Polyhedral Surface. *Computer-Aided Design*, 42(10):942–951, 2010.

[41] Q. Yang, S. He, J. Li, J. Chen, and Y. Sun. Energy-Efficient Probabilistic Area Coverage in Wireless Sensor Networks. *IEEE Trans. on Vehicular Technology*, 2014.

[42] M. Younis, S. Ramasubramanian, and M. Krunz. Location-Unaware Sensing Range Assignment in Sensor Networks. In *Proc of the 6th IFIP Networking*, pages 120–131, 2007.

[43] M. Zhao, J. Lei, M. Wu, Y. Liu, and W. Shu. Surface Coverage in Wireless Sensor Networks. In *Proc of the 28th IEEE INFOCOM*, pages 109–117, 2009.

[44] Q. Zhao and M. Gurusamy. Lifetime Maximization for Connected Target Coverage in Wireless Sensor Networks. *IEEE/ACM Trans. on Networking*, 16(6):1378–1391, 2008.

[45] Z. Zhou, S.R. Das, and H. Gupta. Variable Radii Connected Sensor Cover in Sensor Networks. *ACM Trans. Senor Networks*, 5(1):8:1–8:36, 2009.

Feng Li received his BS degree in Computer Science from Shandong Normal University, China, in 2007, and the MS degree in Computer Science from Shandong University, China, in 2010. He is currently a PhD student at School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are computational geometry and its application in wireless sensor networks.

Jun Luo received his BS and MS degrees in Electrical Engineering from Tsinghua University, China, and the PhD degree in Computer Science from EPFL (Swiss Federal Institute of Technology in Lausanne), Lausanne, Switzerland. From 2006 to 2008, he has worked as a post-doctoral research fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. In 2008, he joined the faculty of the School of Computer Engineering, Nanyang Technological University in Singapore, where he is currently an assistant professor. His research interests include wireless networking, mobile and pervasive computing, applied operations research, as well as network security. More information can be found at http://www3.ntu.edu.sg/home/junluo.

Wenping Wang is a professor of computer science at the University of Hong Kong. His research covers computer graphics, visualization, and geometric computing. He has recently focused on mesh generation and surface modeling for architectural design. He is journal associate editor of Computer Aided Geometric Design (CAGD), Computers and Graphics (CAG), IEEE Transactions on Visualization and Computer Graphics (TVCG, 2008-2012), Computer Graphics Forum (CGF), and IEEE Computer Graphics and Applications (CG&A). He has been the program chair of several international conferences, including Pacific Graphics 2003, ACM Symposium on Physical and Solid Modeling (SPM 2006), International Conference on Shape Modeling (SMI 2009), and the conference chair of Pacific Graphics 2012, SIAM Conference on Geometric and Physical Modeling 2013 (GD/SPM13), and SIGGRAPH Asia 2013

Ying He received the BS and MS degrees in Electrical Engineering from Tsinghua University, China, and the PhD degree in Computer Science from the State University of New York (SUNY), Stony Brook, USA. He is currently an associate professor at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are in the broad areas of visual computing, with a focus on the problems that require geometric computation and analysis. More information can be found at http://www.ntu.edu.sg/home/yhe.