Approximate Convex Decomposition Based Localization in Wireless Sensor Networks

Wenping Liu[1] Dan Wang[2] Hongbo Jiang[1] Wenyu Liu[1] Chonggang Wang[3]

[1]Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[3]InterDigital Communications, U.S.A.
[1]{wenpingliu2009,hongbojiang2004}@gmail.com, liuwy@mail.hust.edu.cn, [2]csdwang@comp.polyu.edu.hk, [3]cgwang@ieee.org

***Abstract*—Accurate localization in wireless sensor networks is the foundation for many applications, such as geographic routing and position-aware data processing. An important research direction for localization is to develop schemes using connectivity information only. These schemes primary apply hop counts to distance estimation. Not surprisingly, they work well only when the network topology has a convex shape. In this paper, we develop a new Localization protocol based on Approximate Convex Decomposition (ACDL). It can calculate the node virtual locations for a large-scale sensor network with arbitrary shapes. The basic idea is to decompose the network into convex subregions. It is not straight-forward, however. We first examine the influential factors on the localization accuracy when the network is concave such as the sharpness of concave angle and the depth of the concave valley. We show that after decomposition, the depth of the concave valley becomes irrelevant. We thus define *concavity* according to the angle at a concave point, which can reflect the localization error. We then propose ACDL protocol for network localization. It consists of four main steps. First, convex and concave nodes are recognized and network boundaries are segmented. As the sensor network is discrete, we show that it is acceptable to approximately identify the concave nodes to control the localization error. Second, an approximate convex decomposition is conducted. Our convex decomposition requires only local information and we show that it has low message overhead. Third, for each convex subsection of the network, an improved Multi-Dimensional Scaling (MDS) algorithm is proposed to compute a relative location map. Fourth, a fast and low complexity merging algorithm is developed to construct the global location map. Our simulation on several representative networks demonstrated that ACDL has localization error that is 60%-90% smaller as compared with the typical MDS-MAP algorithm and 20%-30% smaller as compared to a recent state-of-the-art localization algorithm CATL.**

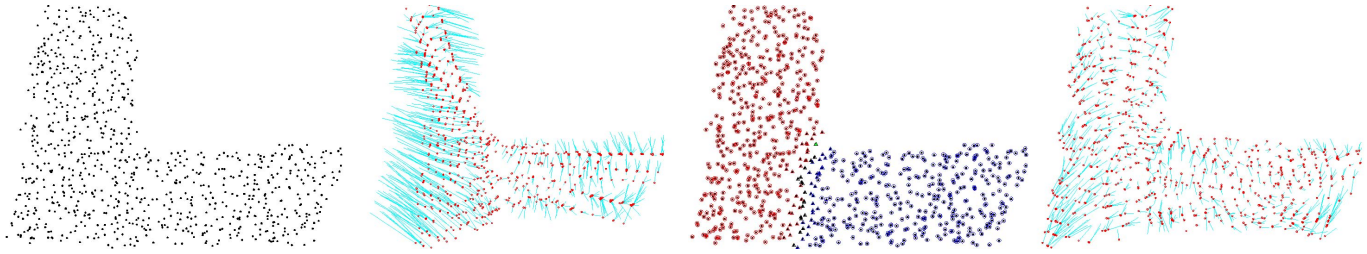
I. Introduction

Location-based service in wireless sensor networks is a key technology for many applications; and localization has attracted academic interest for a long time. The most straightforward method is to use the global positioning system (GPS). Nevertheless, having each node GPS-equipped is extremely expensive for wireless sensor networks. Many algorithms have been proposed to estimate the sensor locations using local information only; a survey on location, localization and localizability can be found in [12].

Recently there has been growing interest in localization protocols that use the connectivity information only. This aims to produce a relative coordinate system for a network without reliance on extra hardware supplements. These schemes can accurately recover the original network topology, up to scaling and rotation. Among the many studies, Multi-Dimensional Scaling (MDS) based localization techniques have been proved to compute locations of high accuracy and request low node density. A state-of-the-art MDS based algorithm, MDS-MAP [15], takes an inter-node hop distance matrix as input, and generates a set of relative coordinates for each node. Nevertheless, the accuracy of MDS-MAP heavily depends on the assumption that the hop-count distance between two nodes correlates well with their Euclidean distance. Such assumption is valid only when the network is in a convex field. In real world, however, this is hardly true. In *anisotropic* networks with concave regions, the shortest path may be significantly bent [11]. As a result, the hop-count distance between nodes would deviate from the Euclidean distance.

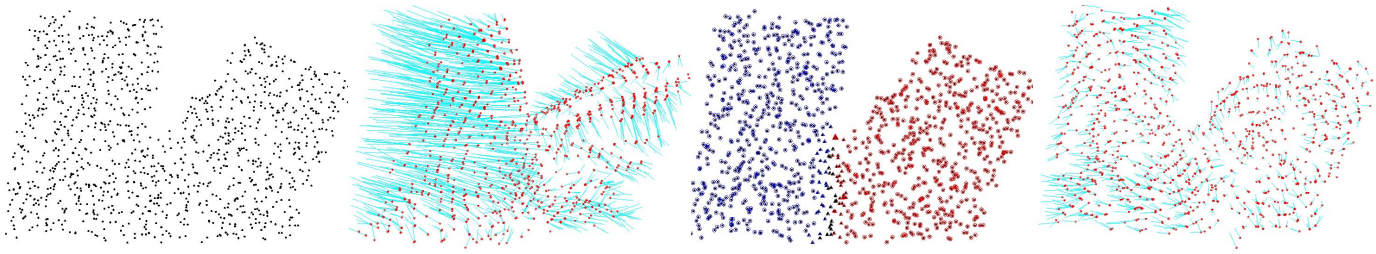
To avoid using hop-count distance between far-away nodes (or to avoid mistakenly using the deviated shortest path), some studies [2], [9], [10], [19] first locate a landmark network. The landmark network is composed of nodes that are uniformly sampled from the original network and the density of the landmark network is usually set by a system parameter. In [9], [10], [19], a triangular mesh structure (the landmark network) is constructed. Each non-landmark node then trilaterates its own location according to the distances to its closest three landmarks. CATL [17] is a recent state-of-the-art localization algorithm. The key idea of CATL is to identify notch nodes where the hop-count of the shortest path between the nodes deviates the true Euclidean distance. CATL then uses an iterative notch-avoiding multilateration scheme to localize the network. The performance of CATL heavily depends on proper deployment of some beacon nodes. In addition, due to the iterative procedure, CATL suffers from error propagation.

In this paper we develop a new localization protocol based on approximate convex decomposition (ACDL). ACDL decomposes the network into several convex subsections. In each subsection, the hop-count distance between nodes can provide a good approximation of the Euclidean distance. ACDL finally unifies the locations of all subsections. ACDL works well not only for arbitrary network shape but also for low density networks since it does not rely on the quality of the extracted



(a) (b) (c) (d)

Fig. 1. Localization of L-shape network with 851 nodes, average degree 11.29. (a) Original map; (b) MDS-MAP; (c) ACD; (d) ACDL.



(a) (b) (c) (d)

Fig. 2. Localization of sharper L-shape network with 1191 nodes, average degree 12.47. (a) Original map; (b) MDS-MAP; (c) ACD; (d) ACDL.

triangular mesh like [9], [10], [19]. It avoids localization error propagation introduced by iterative procedure like [17].

While the idea of convex decomposition is simple, it is impractical to manually identify convex/concave regions during deployment or extract a graph of the network. As such, many difficulties need to be addressed. *First*, there is a lack of understanding of localization error and concavity. *Second*, since the sensor network is discrete, due to boundary noise, it is not easy to clearly specify the nodes that are concave of the network. *Third*, though there are available convex decomposition schemes from the computer graphics community, they target at continuous shapes and use centralized solutions. The network is discrete and the sensor nodes can only obtain local information by message exchange. A low communication complexity scheme is necessary considering the limited resource of each sensor node.

In this paper, we provide a systematic study on the aforementioned problems. Our key contributions are:

- We illustrate the intrinsic problems of localization algorithm on concave-shaped networks. We show that the location accuracy is closely related to the angle at the concave points and it is also related to the depth of the concave "valley". After convex decomposition, the depth of the valley becomes irrelevant, however. With these observations, we make a formal definition of network *concavity* according to the angle at the concave point.
- We develop a distributed approximate convex decomposition algorithm, based on the boundary branches. Our algorithm has low communication complexity.
- An improved MDS is applied in each convex subsection to compute the relative location map. The computation of MDS is $O(N^3)$ but our improved MDS proposed in this paper has a complexity of $O(N^2)$.

The rest of this paper is organized as follows. Section II presents the motivation of our work and the background of our convex decomposition scheme. We give the definition of concavity. Section III is devoted to ACDL, our distributed approximate convex decomposition based localization algorithm. We evaluate the performance of ACDL in Section IV. Finally, Section V concludes the paper.

II. MDS Localization and Network Concavity

Our localization protocol relies upon the convex decomposition. To this end, one essential step is to identify concave node(s). Our idea is quite simple: a boundary node identifies itself a concave node when its curvature is greater than a given threshold (we refer the reader to [3] for the definition of boundary). We will define the node curvature later. First we illustrate the intrinsic problems of traditional algorithm on concave-shape networks.

Let N be the number of sensors in the network. Multi-dimensional scaling [1] first computes an $N \times N$ distance matrix D, which represents the pairwise distance between two nodes, thereby calculating the inner product matrix B of the pairwise distance matrix D. MDS then applies spectrum decomposition on matrix B to extract all eigenvalues and their corresponding eigenvectors of matrix B. Finally, MDS computes locations based on the first 2 largest eigenvalues and eigenvectors. MDS-MAP [15] is an MDS-based localization algorithm. It uses the hop-count of the shortest path between two nodes as the pairwise distance in the matrix. MDS-MAP works well for the sensor field with a simple shape such as a square or a disk. This is because the hop-count of the shortest path between two nodes is a good approximation of the Euclidean distance. For complex networks, this condition could be no longer valid.

As a concrete example, in Fig. 1, we apply MDS-MAP directly to an L-shape network (Fig. 1 (a)). In Fig. 1 (b), the line associated with each node is the deviation between the real location and computed location by MDS-MAP. Not surprisingly, the accuracy of MDS-MAP is low. Especially, the nodes at the end of the two arms of the L-shape network generally have greater errors. To further understand the impact of concavity, we evaluate another L-shape network in Fig. 2. Note that the angle at the concave point in Fig. 2(a) is sharper as compared to that in Fig. 1 (a). From Fig. 2(b), we can see the location accuracy is even worse. Based on these experiments, it is easy to observe 1) the angle of the concave point is of crucial importance; the sharper the angle is, the worse the performance of MDS-MAP; and 2) the sensors at the two arms of the L-shape network suffer greater errors. This indicates that the depth of a concave valley can also be an important factor. A natural idea is to decompose the network into convex regions. To evaluate the effectiveness of such idea, we manually decompose the L-shape network into two convex subsections, see Fig. 1 (c). We then conduct the MDS localization algorithm, and the result is shown in Fig. 1 (d). We see that the localization errors are greatly reduced. In addition, the depth of the concave valley becomes irrelevant.

With these observations, next we turn to the concave node definition. We define the k *neighborhood* of node p, represented by $N_k(p)$, as the set of the nodes at most k hops away to p. Let k*-hop neighborhood* of p, denoted as $\partial N_k(p)$, be the nodes exactly k hops away from p. Intuitively, $\partial N_k(p)$ can be treated as a (or a part of) circle centered at node p. Given two nodes $p_1, p_2 \in \partial N_k(p)$, a *perimeter* from p_1 to p_2, denoted by $D_k^p(p_1, p_2)$, is the set of nodes which are on the shortest path from p_1 to p_2(including p_1 and p_2) using the nodes in $\partial N_k(p)$. We thus define the perimeter distance, $|D_k^p(p_1, p_2)|$, from p_1 to p_2, as the number of sensors in $D_k^p(p_1, p_2)$ minus one. We illustrate our definitions by an example in Fig. 3 and define the *concavity* by:

Definition 1. *Assume $p_1, p_2 \in \partial N_k(p)$ are two boundary nodes which are on the same boundary with p. The* k-hop concavity *(or so-called curvature) of p, $c_k(p)$, is given by:*

$$c_k(p) = \frac{|D_k^p(p_1, p_2)|}{\pi \times k} \tag{1}$$

Intuitively, if $c_k(p)$ is equal to 1, p is not a concave/convex point. Due to the discrete nature of wireless sensor networks, the presence of boundary noise will incur many boundary nodes have a curvature which is larger than 1. As such, to control such boundary noise, we introduce two thresholds and our definitions on concave/convex nodes are as follows.

Definition 2. *Given $\delta_1 > 0$ and $\delta_2 > 0$, a boundary node p is a* concave node *if $c_k(p) > 1 + \delta_1$, or a* convex node *if $c_k(p) < 1 - \delta_2$.*

The larger the curvature of the concave node, the larger the deviation is. Note that for a sensor network, there can be many concave nodes. The concave node q with the maximum concavity has the most significant influence on the localization

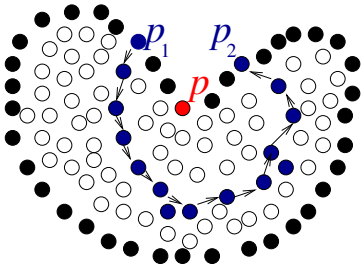


Fig. 3. Boundary nodes are filled with black. p is filled with red. The nodes in $\partial N_3(p)$ are filled with blue. p_1 and p_2 are both boundary nodes and 3-hop neighborhood nodes of p. The perimeter distance $|D_3^p(p_1, p_2)| = 13 - 1 = 12$, and the hop-count is indicated by the arrows. $c_3(p) = \frac{12}{3\pi} = 1.27$. If $\delta_1 < 0.27$, p will be a concave node.

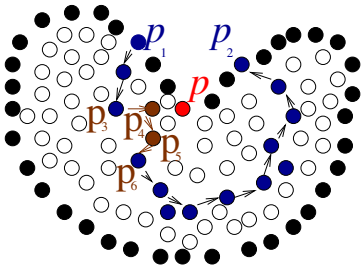


Fig. 4. Boundary nodes are filled with black. p is filled with red. The nodes in $\partial N_3(p)$ are filled with blue. p_1 and p_2 are both boundary nodes and 3-hop neighborhood nodes of p. The auxiliary nodes to make $\partial N_3(p)$ a connected component are filled with brown, $\delta = 1$. The perimeter distance $|D_3^p(p_1, p_2)| = (14 - 1) - 1 = 12$, and the hop-count is indicated by the arrows. $c_3(p) = \frac{12}{3\pi} = 1.27$. If $\delta_1 < 0.27$, p will be a concave node.

accuracy of the network. The *network concavity*, represented by $C_k(V)$, is defined by the maximum concavity of all concave nodes, $C_k(V) = max_{s \in V_c} c_k(s)$, where V_c is the set of the concave nodes.

Based upon the definition of the concave/convex nodes, we will design a distributed algorithm in next section, which can find the concave/convex nodes, decompose the network and compute the locations with high accuracy. It should be noted that the definitions of concavity and concave/convex node depend on parameters k, which corresponds to the depth of the valley, as well as δ_1 and δ_2, which correspond to the sharpness of the angle. Intuitively, a smaller value of k (or δ_1) implies that more concave nodes will be identified, and the network will be partitioned into more subsections.

III. Approximate Convex Decomposition-based Localization (ACDL)

A. An Overview of ACDL

In our paper, we assume that the network is fully connected. Note that since the sensor network is discrete, it is impractical to decompose the wireless sensor network into strictly convex subsections due to the boundary noise. MDS-based localization tolerates localization error gracefully due to its over-determined nature [14]. As such, we only have to decompose the network into approximate convex subsections. For each approximate convex subsection, the maximum concavity is less than the given threshold value $1 + \delta_1$.

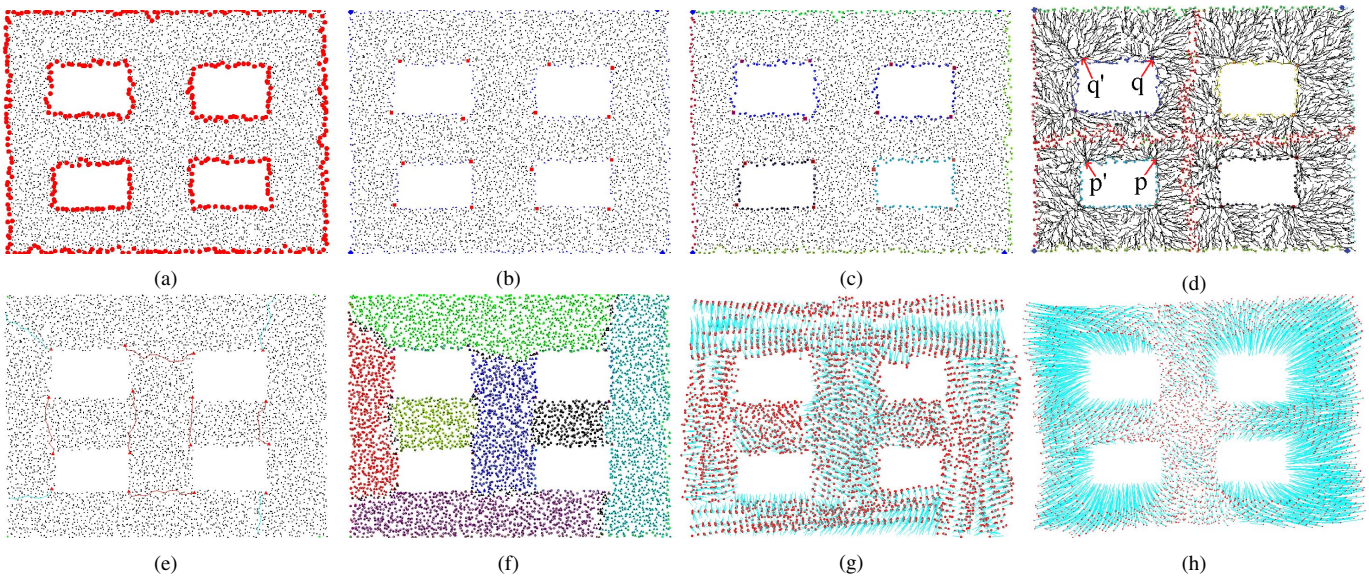


Fig. 5. Localization for window-shape network with 5184 nodes and average degree 12.77. (a) Original map; (b) Concave/convex nodes, $k = 5, \delta_1 = 0.5, \delta_2 = 0.2$. The red rectangular-shaped nodes are concave nodes and the blue diamond-shaped nodes are convex nodes; (c) Boundary branches. Boundary nodes on different branches are marked different colors; (d) Cut trees. Cut nodes are marked red; (e) Segment lines; (f) Approximate convex decomposition; (g) Localization result by our method. Average localization error is 0.43; (h) Localization result by MDS-MAP. Average localization error is 2.18.

Our Approximate Convex Decomposition-based Localization algorithm consists of the following four steps:

(1) Concave/convex Node Recognition and Boundary Segmentation: Assume we have the boundaries of the network (boundary identification is out of the scope of our paper and there are many existing works [3], [4], [13], [18] and we use [18] in our simulation), we first identify the concave and convex nodes. We then segment the boundaries into several boundary branches using these convex nodes. These branches will serve as the basis for our decomposition.

(2) Approximate Convex Decomposition (ACD): The key problem is to find the "lines" that can decompose the network into convex subsections. These lines may end at concave nodes (and boundary nodes). We develop a distributed algorithm to identify these lines.

(3) Local Relative Map Establishment: We proposed an improved MDS technique to build the relative map for each convex subsection. Our algorithm has a computational complexity of $O(N^2)$ while the conventional MDS is $O(N^3)$,

(4) Global Map Establishment: Finally, we will combine the coordinates of all the subsections into a global map. Note that the combination process will need to inform each sensor in the subsection of the new coordinates. If the combination process is conducted one subsection at a time, such combination will be slow. Thus, we develop an algorithm which can balance the time of combination process and the message overhead.

B. Concave/convex Node Recognition and Boundary Segmentation

Given the definition for concave/convex node, we present how each node identifies itself in a distributed manner.

As discussed, each node first uses the technique in [18] to identify whether it is a boundary node. For each boundary node p, it first floods the network for k hops to obtain its k-hop neighborhood $\partial N_k(p)$. Two nodes who belong to $\partial N_k(p)$ and are also boundary nodes identify themselves. Note that at each side of the boundary node p, there may be several boundary nodes belonging to $\partial N_k(p)$; and these boundary nodes naturally form a connected component. For this case, we randomly select one (e.g., the boundary node with smallest node ID) boundary node at each side of the boundary node p. Let these two nodes be p_1, p_2. p_1 and p_2 then flood in $\partial N_k(p)$ to derive the perimeter distances $|D_k^p(p_1, p_2)|$ and $|D_k^p(p_2, p_1)|$; and send this information to p (See Fig. 3). Obviously, $|D_k^p(p_1, p_2)| = |D_k^p(p_2, p_1)|$. We show this process in Algorithm 1.

It is noted that $\partial N_k(p)$ might be disconnected. In this case, we use some auxiliary nodes to estimate the perimeter distance $|D_k^p(p_1, p_2)|$ in a greedy manner, see Fig. 4. Node p_3 can not send the message received from p_1 to node p_6 directly; p_3 then sends this message to its neighbor p_4, which is $(k-1)$ hops from p. Node p_4 will send the message to p_5 since p_4 has no neighbor belonging to $\partial N_k(p)$; and then p_5 sends the message to p_6 until the message from p_1 is received by p_2; and thus the estimated perimeter $D_k^p(p_1, p_2)$ is obtained. Here p_4 and p_5 are auxiliary nodes. The message from p_2 to p_1 can travel in the same way. The perimeter distance is thus estimated by $(|D_k^p(p_1, p_2)| - \delta)$, where δ is the difference between k and the hop count distance (k-1 in Fig. 4) of the closest auxiliary node to the boundary node p.

With the perimeter distance $|D_k^p(p_1, p_2)|$ estimated, node p computes its curvature and identifies whether it is a concave/convex node. We show this process in Fig. 5(b). Using the convex nodes, the boundaries are automatically segmented into several *boundary branches*, i.e., a sub-boundary including all nodes between two adjacent convex nodes; see Fig. 5(c)

for an example. Each convex node floods in the boundary nodes to segment the boundary into several branches. When a boundary node p receives a packet from a convex node, it sends this packet to its neighboring boundary node if p is not a convex node, otherwise it discards this packet. As such, each boundary nodes will keep track of two convex nodes which determine a unique boundary branch. Note that it is possible that there are no convex nodes identified for some boundaries (e.g., the boundaries of the inner holes in Fig. 5). To deal with this, any node on each of these boundaries floods within the same boundary; and the node with smallest node ID will determine a unique boundary branch. Based on these boundary branches, we can find segment lines and decompose the network into convex subsections.

As there can be several concave nodes that are on the same boundary branch and within a small distance (e.g., k hops), resulting in a heavily fragmented network, we choose the concave node with the largest concavity and disregard the others. Notice that after this concave/convex recognition, every node s in $\partial N_k(p)$ obtains its distances to p_1 and p_2. We denote these as $h_k(s,p_1), h_k(s,p_2)$, and let $H_k(s) = \max\{h_k(s,p_1), h_k(s,p_2)\}$. We will see that $H_k(s)$ is very important for convex decomposition in Section III-C.

Algorithm 1 Concave/convex Node Recognition

1: p obtains its k-hop neighborhood $\partial N_k(p)$, two of which are boundary nodes, denoted by p_1, p_2.
2: p derives the perimeter distance $|D_k^p(p_1,p_2)|$.
3: p computes its concavity $c_k(p)$ using Equ. (1).
4: **if** $c_k(p) \geq 1+\delta_1$ **then**
5: Node p identifies itself as a concave node.
6: **else if** $c_k(p) \leq 1-\delta_2$ **then**
7: Node p identifies itself as a convex node.
8: **end if**

C. Approximate Convex Decomposition (ACD)

The key to decompose the network is to find the segment lines. We first formally define such segment lines.

Definition 3. *A segment line is a non-empty connected component where only the two end nodes of this line are boundary nodes and belong to different boundary branches.*

In the process of ACD, we want to identify the segment lines that can reduce the concavity $C_k(V)$, of the network, and decompose the network into convex subsections.

Theorem 1. *A segment line can reduce the concavity of the network if it contains concave node(s).*

Proof: If a segment line l contains a concave node p who has the largest concavity (see Fig. 6), then the network will be decomposed into two subsections, of which the segment line l is a boundary. Node $p\prime$ is the intersection node of segment line l and $\partial N_k(p)$; and $p\prime$ *splits* the perimeter $D_k^p(p_1,p_2)$ into two subsets: $D_k^p(p_1,p\prime)$ and $D_k^p(p\prime,p_2)$. Obviously, we have $D_k^p(p_1,p\prime) \subset D_k^p(p_1,p_2), D_k^p(p\prime,p_2) \subset D_k^p(p_1,p_2)$, therefore

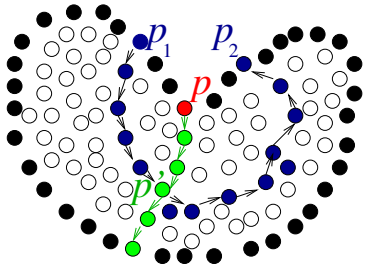


Fig. 6. The nodes on a segment line are marked green. Node $p\prime$ is the intersection node of perimeter $D_3^p(p_1,p_2)$ and the segment line, and $p\prime$ decomposes $D_3^p(p_1,p_2)$ into two sub-perimeters: $D_3^p(p_1,p\prime)$ and $D_3^p(p\prime,p_2)$. The segment line decomposes the network into two subsections, and thus the concavity of p is reduced.

$max\{|D_k^p(p_1,p\prime)|, |D_k^p(p\prime,p_2|\} < |D_k^p(p_1,p_2)|$ holds. Consequently, the concavity of node p for each subsection is less than the original value; thus line l reduces the concavity. ■

As shown in Fig. 6, visually, the segment line decomposes the network into two subsections, each of which has a smaller concavity as compared with the original network. To find segment lines, we define *cut* $\mathcal{C}(p_1,p_2)$ of two concave nodes p_1, p_2, where p_1, p_2 belong to different boundary branches. Let distance $d(s_i,s_j)$ be the minimum hop count between node s_i and s_j. Node $s \in \mathcal{C}(p_1,p_2)$ if $d(s,p_1) = \min_{\forall i} d(s,p_i)$, $d(s,p_2) = \min_{\forall i \&\& i \neq 1} d(s,p_i)$ and $|d(s,p_2) - d(s,p_1)| \leq 1$.

Note that $\mathcal{C}(p_1,p_2)$ can be easily identified by a flooding of all the concave nodes. Each concave node p floods the network and builds a shortest path tree. When a node s receives a message from a concave node p, there are two cases: 1) If s has not received a message from any concave node, s will join the tree and broadcast the message; or, 2) If s has received a message from another concave node, s will discard the message. Eventually, a tree rooted at concave node p will be constructed and we call this tree *cut tree*, denoted by $CT(p)$. Two cut trees $CT(p_1)$ and $CT(p_2)$ whose roots p_1, p_2 belong to different boundary branches may *meet*. We denote these nodes where the two trees meet as *cut* $\mathcal{C}(p_1,p_2)$(see Fig. 5 (d)); and the nodes that lie on the cut are called cut nodes. Intuitively, each cut node has the smallest distance to its root among all concave nodes. Especially, we define a cut-pair (p,q) as two neighboring cut nodes $p,q \in \mathcal{C}(p_1,p_2)$ who belong to different cut trees, where p_1 is the root of p and p_2 is the root of q. Note that in Sec.III-B, each node s who is k hops from a concave node computes $H_k(s)$. Obviously, to reduce the concavity of a concave node down to below $1+\delta_1$, the segment line should pass at lease one node s who satisfies $\frac{H_k(s)}{\pi \times k} < 1+\delta_1$. We call such node *candidate segment node*. If a cut-pair $(p,q) \in \mathcal{C}(p_1,p_2)$ satisfies that each of the two shortest paths from p_1 to p and q_1 to q will pass at least one candidate segment node respectively, these two paths together with the path from p_1 to p_2 forms a line which can reduce the concavities of p and q, and decompose the network into two subsections S_1, S_2. We call such line as *candidate segment line*, on which the nodes are boundary nodes of S_1, S_2. However, there can be more than one *candidate segment line*; and the nodes (specifically, the cut-pair (p,q)) on such line

may be concave for one subsection, say, S_1. If we treat a cut-pair (p, q) as a *dummy* node $p\prime$, then we only consider the concavity at $p\prime$ in S_1.

Lemma 2. *For two cut-pairs $(p, q), (l, s) \in \mathcal{C}(p_1, p_2)$, if $d(p, p_1) + d(q, p_2) < d(l, p_1) + d(s, p_2)$, then we have $c_k(p\prime) < c_k(l\prime)$ where $l\prime$ is the dummy node by* merging l, s.

As such, we can obtain a segment line which crosses a cut-pair (p, q) and satisfies $d(p, p_1) + d(q, p_2) = \max_{\forall(l,s)\in\mathcal{C}(p_1,p_2)}\{d(l, p_1) + d(s, p_2)\}$. If $c_k(p\prime) > 1 + \delta_1$, we extend the shortest path from p_1 to p (or p_2 to q) until the path *meets* a boundary node.

Lemma 3. *If the segment line between two concave nodes is the shortest path between them, then all nodes on the segment line are not concave.*

Note that cut nodes exist only when two concave nodes belong to different boundary branches, see Fig. 5(d) for an example. Concave node p and $p\prime$ are on the same boundary branch, connecting p and $p\prime$ cannot reduce their concavities. However, p and q are not on the same boundary branch, it is possible to reduce their concavities by connecting p and q.

It is noted that one possible problem here is that, for some concave nodes (e.g., $q\prime$ in Fig. 5(d)), there are no *cut* nodes corresponding to them. To deal with this, for such concave node p, we find a *best* boundary node q which satisfies: 1) q is on the tree rooted at p; 2) the shortest path from p to q will cross at least one candidate segment node; and 3) the concavity of q is maximum among all boundary nodes who satisfy condition 1). Obviously, the shortest path from p to q can reduce the concavity of p down to below $1 + \delta_1$.

Theorem 4. *All the subsections generated in above way are approximately convex.*

Proof: As a segment line decomposes a network into two subsections, of which the segment line is a boundary node, we only need to prove that after ACD, all nodes (including concave nodes and nodes on segment lines) have a concavity of less than $1 + \delta_1$.

Generally, for a concave node p_1, there are two possible cases:

CASE 1: There is no cut nodes associated with p_1. This may happen when all concave nodes adjacent to p_1 are on the same boundary branch. For this case, our strategy is to construct a segment line by connecting p_1 with the *best* boundary node q, as described above. Such strategy clearly guarantees that the segment line can reduce the concavity of p_1 while each node on the segment line has a concavity of no greater than the given threshold value $1 + \delta_1$.

CASE 2: There are cut nodes formed by p_1 and another concave node, say p_2. If the concavity of the dummy node, which is formed by a cur-pair $p, q \in (p_1, p_2)$, is less than $1 + \delta_1$, the shortest paths between p, p_1 and q, p_2 will serve as a segment line, by which the concavities of p_1 and p_2 are reduced down to below $1 + \delta_1$. At the same time, the concavity of each node on the shortest path is smaller than $1 + \delta_1$. Otherwise, we extend the path between q, p_1 (or q, p_2) until the shortest path hits a boundary (or a segment line). By doing so, the concavities of p_1, p_2 and the dummy node are reduced.

Overall, after ACD, the concavities of all concave nodes are reduced down to below $1 + \delta_1$, and the concavity of each node on the obtained segment line is also less than $1 + \delta_1$. That is, all subsections are approximately convex. ■

D. Local Relative Map Establishment

So far we have decomposed the network into several convex subsections. We assume here that, there exists a localization *coordinator* (or simply coordinator which can be an arbitrary node) within each subsection, and the coordinator is in charge of the process of local relative map establishment. Let the number of sensors in Sec_i be n_i. Within one subsection, each node sends its neighbor list to the coordinator (this process has a message complexity of $O(n_i \log_2 n_i)$), which is subject to solving the all-pairs shortest paths problem in undirected graph with integer weights [16]. Next take the distance matrix as an input, the coordinator applies an improved MDS algorithm to establish a local relative map.

Given an $n_i \times n_i$ pairwise distance matrix D_i of Sec_i, MDS first constructs the inner product matrix $B_i = -\frac{1}{2}HD_iH$, where $H = I - \frac{1}{n_i}e^Te$, I is an n_i orders unit matrix and e is an n_i-dimensional vector of all ones. MDS then conducts spectral decomposition on matrix $B_i = Q\Lambda Q$, where Λ is the eigenvalues diagonal matrix and Q is the eigenvectors matrix of B_i. The complexity of spectral decomposition on matrix B_i is $O(n_i^3)$.

Note that in MDS-based localization scheme, only the first m largest eigenvalues are used. Thus we can use the power method of a matrix only m times to obtain these m eigenvalues and eigenvectors, instead of all eigenvalues and eigenvectors. The power method (also known as the power iteration) of a matrix B is designed for extracting the dominant eigenvalue (i.e., the first eigenvalue with the largest magnitude) and the corresponding eigenvector. Repeating power method m times can derive the first m largest eigenvalues and eigenvectors.

For each subsection Sec_i, it will execute Algorithm 2 to localize those nodes in Sec_i.

Algorithm 2 Local Map Establishment

1: **for all** convex subsection Sec_i **do**
2: Coordinator p of Sec_i computes pairwise distance matrix D_i in Sec_i and the inner product matrix of D_i, denoted by B_i .
3: Initialize:$\lambda_0 = 0, q_0 = e$
4: **for** $k = 1$ to m **do**
5: p uses power method on $B_i - \sum_0^{k-1} \lambda_l q_l q_l^T$ to extract the largest eigenvalue λ_k and eigenvector q_k.
6: **end for**
7: The locations of node s_j in Sec_i are given as: $X_{ij1} = \sqrt{\lambda_1}q_{1j}, \ldots, X_{ijm} = \sqrt{\lambda_m}q_{mj}$
8: **end for**

Theorem 5. *For an $n_i \times n_i$ inner product matrix B_i, the computational complexity of Algorithm 2 is $O(n_i^2)$, which is an order less than that of MDS-MAP.*

Proof: Suppose the n_i eigenvalues of a $n_i \times n_i$ matrix B are ordered by $\lambda_1 > \lambda_2 > \ldots \lambda_{n_i}$ and its eigenvectors are $q_1, q_2, \ldots, q_{n_i}$ correspondingly, we have $Q = \lambda_1 q_1 q_1^T + \lambda_1 q_2 q_2^T + \cdots + \lambda_{n_i} q_{n_i} q_{n_i}^T$ immediately. Now we start with an arbitrary nonzero n_i-dimensional vector x_0. The matrix B is obviously symmetrical, whose n eigenvectors are orthogonal and can be viewed as a set of bases of n-dimensional space. Therefore the vector x_0 can be represented by the combination of $q_1, q_2, \ldots, q_{n_i}$, namely

$$x_0 = a_1 q_1 + a_2 q_2 + \cdots + a_{n_i} q_{n_i}$$

Then, we construct a series of iterated vectors:

$$x^1 = Bx_0, x^2 = Bx^1, \ldots, x^k = Bx^{k-1} = B^k x_0,$$

where

$$\begin{aligned} x^1 &= Bx_0 = a_1 Bq_1 + a_2 Bq_2 + \cdots + a_n Bq_{n_i} \\ &= a_1\lambda_1 q_1 + a_2\lambda_2 q_2 + \cdots + a_{n_i}\lambda_n q_{n_i}, \\ x^k &= B^k x_0 = a_1 B^k q_1 + a_2 B^k q_2 + \cdots + a_{n_i} B^k q_{n_i} \\ &= a_1\lambda_1^k q_1 + a_2\lambda_2^k q_2 + \cdots + a_{n_i}\lambda_{n_i}^k q_{n_i} \\ &= \lambda_1^k\{a_1 q_1 + a_2(\frac{\lambda_2}{\lambda_1})^k q_2 + \cdots + a_n(\frac{\lambda_{n_i}}{\lambda_1})^k q_{n_i}\} \end{aligned}$$

Since $|\lambda_1| > |\lambda_j| (j = 2, 3, \ldots, n_i)$, when k becomes larger, $x^k \approx a_1\lambda_1^k q_1, \frac{x^k}{x^{k-1}} \approx \lambda_1$. We can iterate the system by $x^k = \frac{Bx^{k-1}}{\|Bx^{k-1}\|_\infty}$ (for each iteration, this takes n_i^2 times multiplication operations, n_i^2 times addition operations, and one time division operation. Thus the computational complexity is $O(n_i^2)$) until $|\parallel x^k \parallel_\infty - \parallel x^{k-1} \parallel_\infty| < eps$, where eps is the given precision value and the scalar factor $\parallel Bx^{k-1} \parallel_\infty = max\{Bx^{k-1}\}$ is used to prevent the iterated vectors from getting extremely large or small. Thus we obtain the first largest eigenvalue $\lambda_1 = \parallel Bx^{k-1} \parallel_\infty$ and its corresponding eigenvector $q_1 = x^k$. The overall computational complexity of this step is $O(k \times n_i^2) = O(n_i^2)$. We subtract $\lambda_1 q_1 q_1^T$ from B and repeat the procedure for matrix $B - \lambda_1 q_1 q_1^T$ (this will take $n_i^2 + n_i$ times multiplication operations and n_i^2 times subtract operations) and derive the second largest eigenvalue λ_2 and eigenvector q_2. Consequently, this step takes a computational complexity of $O(k' n_i^2)$ where k' is the iteration number. It is noted that this process is iterative and may take a long time to enter a very stable status. Such high stability is however not important to our approximate localization, so we artificially set a maximum of 50 iterations to the algorithm. That is, this step has a computational complexity of $O(n_i^2)$. Repeat the procedure for matrix $B - \lambda_1 q_1 q_1^T - \lambda_2 q_2 q_2^T$ (this takes $2 \times (n_i^2 + n_i)$ times multiplication operations and $2 \times n_i^2$ times subtract operations). Finally we obtain the position of the j-th node $X_{j1} = \sqrt{\lambda_1} q_1(j), X_{j2} = \sqrt{\lambda_2} q_2(j)$. Overall, the computational complexity of the algorithm is $O(n_i^2)$. ■

E. Global Map Establishment

After assigning virtual coordinates within each convex subsection, we now combine them to form a global map. For every two adjacent subsections, there are some nodes on their common segment line, and these nodes are assigned two virtual coordinates. We find a linear transformation for these common nodes using their two virtual coordinates. Based on the linear transformation,we combine these two subsections into a bigger one. Note that the fact of non-overlapping partitions determines a unique way of putting adjacent partitions together. This way we are able to recovery the global layout of the network.

In this subsection, we introduce a time round scheme to minimize the time cost for global map establishment. To clarify our statement, we use ACG (Adjacent Constraint Graph) of a network [8]. Here in ACG, each vertex i represents a subsection Sec_i; and two vertexes i, j are neighborhood if Sec_i and Sec_j are adjacent. Let d_i denote the degree of vertex i which indicates how many subsections adjacent to subsection Sec_i. Obviously, d_i equals the number of segment lines in Sec_i. In addition, each vertex i is assigned a weight w_i. w_i equals the node number in Sec_i and will be used in our merging process. For two neighboring vertex i and j, if $d_i > d_j$ or $d_i = d_j, w_i > w_j$, vertex i merges vertex j which means the coordinates of nodes in Sec_j will be transformed in terms of the coordinate system in Sec_i. and these two vertexes is treated as one larger vertex with degree $d_i = d_i + d_j - 1$ and weight $w_i = w_i + w_j - n_{ij}$. Here n_{ij} is the number of nodes on their common segment line.

Our algorithm starts with finding the linear transformation between two neighboring vertexes i and j, and then conducts the merging process round by round. In each round, two neighboring vertexes will merge together; and the number of vertexes will be reduced by fifty percent. The merging process will end when all vertexes are merged as one vertex; and the global map establishment is completed (see Fig. 5(g)). The algorithm 3 shows the process of global map establishment.

Theorem 6. *The time complexity of global map establishment is at most $O(\log_2 N)$ times round.*

Proof: We only prove the time complexity for the worst case, namely, Chain-shape network (See Fig.7), is $O(\log_2 N)$ where the number of vertexes (convex subsections) is $O(N)$. After the first round, there are $\frac{N}{2}$ vertexes; and after the second round, $\frac{N}{4}$. After $O(\log_2 N)$ rounds, all vertexes are merged together. Thus the global map establishment is completed. ■

Algorithm 3 Global Map Establishment

```
while The number of subsections is larger than one do
    For every two adjacent subsections Sec_j and Sec_k,
    if d_j < d_k or (d_j = d_k and w_j < w_k) then
        Sec_k = Sec_k ∪ Sec_j.
        d_k = d_k + d_j − 1, w_k = w_k + w_j − n_kj.
    end if
end while
```

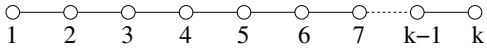


Fig. 7. ACG of Chain-shape network.

Topology	Scheme	ALE	5-prtl Err.	50-prtl Err.	95-prtl Err.	Max Err.
Flower	MDS-MAP	2.09	0.48	1.91	4.06	6.32
	CATL	0.79	0.68	0.67	1.45	2.64
	ACDL	0.62	0.10	0.69	1.36	1.56
Snake	MDS-MAP	11.18	0.29	12.06	27.19	32.42
	CATL	9.04	0.24	3.82	26.14	30.41
	ACDL	0.55	0.09	0.53	0.69	1.71

TABLE I
LOCALIZATION ERRORS OF ACDL, MDS-MAP AND CATL

IV. PERFORMANCE EVALUATION

A. Simulation Setup

We evaluate our algorithm on several network topologies. Due to space limitation, we only present some representative results here. In this section, by default, nodes are uniformly distributed and have the same communication range. Two nodes are connected if and only if the Euclidean distance between them is no greater than a given communication radio range R.

We compare our algorithm with MDS-MAP [15] and CATL [17]. For comparison, two metrics are used in this paper: *localization error (LE)* and *average localization error (ALE)*. *LE* of node p is defined as the Euclidean distance between the estimated location of p and its location. Besides, we refer to *ALE* as the ratio of the mean localization error of each node to the communication range R. To have absolute locations, we randomly deploy three beacon nodes equipped with GPS. The default parameters for the algorithms are $k = 4, \delta_1 = 0.4, \delta_2 = 0.3$.

B. Performance Evaluation of Algorithms under Different Scenarios

We present our results in two different forms. First, we show in Fig. 8 and 9 the localization error of each individual node for these two different network shapes. Second, we summarize the statistical localization error information of the two networks. We show in Table I five kinds of localization errors, the *ALE*, 5-percentile, 50-percentile, 95-percentile and the maximum of the localization errors.

Fig. 8 shows the localization results using MDS-MAP (see Fig. 8(b)), CATL (see Fig. 8(c)), and ACDL (see Fig. 8(d)) on Flower-shape network. In Fig. 8(b)-(d), we plot both the true position and estimated position of a node. We use a light-blue line to connect these two positions and the length of the line represents the sheer localization error, i.e., the longer the line is, the larger the error. We can see that in general MDS-MAP, CATL, and ACDL are all with reasonable localization accuracies, where MDS-MAP is slightly worse. Looking into Table I, we found that the ALE of MDS-MAP is 2.09, CATL is 0.79 and ACDL is 0.69. This is not surprising as both CATL and ACDL are targeting on irregular shapes. ACDL is better than CATL in ALE, 5% Err, 95% Err, and Max Err, and only slightly worse than CATL in 50% Err.

We next examine the performance of MDS-MAP, CATL and ACDL on Snake-shape network (see Fig. 9). As different from the Flower-shape network, there is a long and narrow neck in the topology. We see that MDS-MAP and CATL can not faithfully recover the network layout while the result using ACDL is acceptable. Looking into Table I, we observe, furthermore, that the ALE and Max Err of MDS-MAP and CATL are more than one order larger than ACDL. The reason is that CATL localizes the network in an iterative fashion. After each iteration, newly localized nodes will serve as beacon nodes and flood their locations through the network. Other nodes then calculate their locations using multilateration. The localization errors after each round will be accumulated. Obviously, the node farther from beacon nodes tends to have a larger error. ACDL is an MDS-based algorithm; and ACDL does not suffer from the accumulated errors. Therefore the ALE and Max Err of ACDL will be much smaller than CATL.

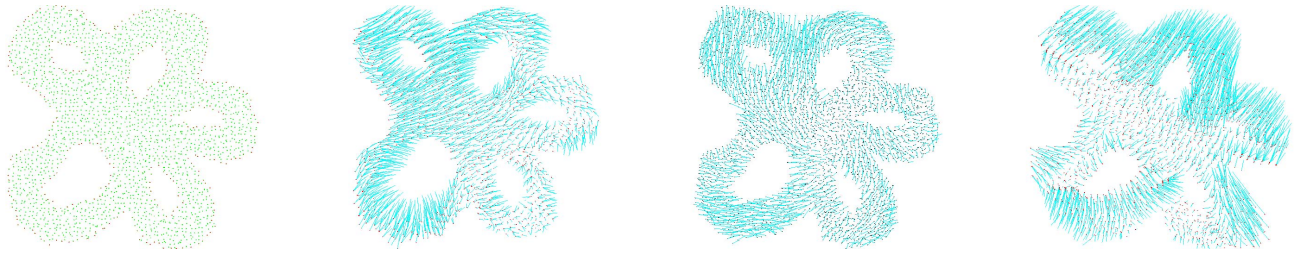
In summary, MDS-MAP does not perform well for irregular network. The performance of CATL depends on the choice of beacon nodes. In addition, CATL uses an iterative multilateration scheme and localization errors are accumulated round by round. As opposed to previous two methods, ACDL decomposes the network into convex subsections and uses MDS to localize nodes in each subsection. As a result, ACDL can recover the network layouts with small errors.

Overall, ACDL decomposes any irregular-shaped network into several convex subsections, followed by conducting MDS within each convex subsection. Since hop count distance between the pair of nodes correlates well with its true Euclidean in convex subsection, ACDL is stable in localization accuracy and consistently outperforms MDS-MAP and CATL.

V. CONCLUSION

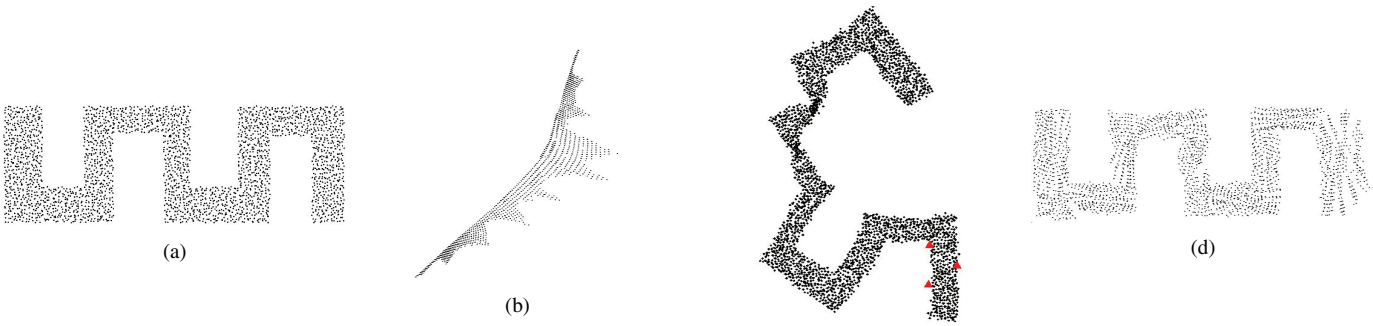
We have proposed a novel connectivity based algorithm, Approximate Convex Decomposition based Localization (ACDL), for localization of wireless sensor networks in irregular shape networks. We overcame a series of difficulties in concave/convex node identification, network decomposition, MDS computation and global map reconstruction. All these were achieved in a way that is discrete, distributed, and low message and computation overhead. Our experiments show that ACDL is significantly better than MDS-MAP for 60% - 90% in localization error. ACDL is also better than CATL for 20% - 30% in most cases. Especially CATL performs poor (worse than MDS-MAP) if the beacon nodes are not selected carefully.

We believe that ACDL has room for improvement. Besides, ACDL may incorporate with other localization algorithms to further improve localization accuracy. We plan to explore the possibility of using localization results to facilitate data processing [5]–[7] in sensor networks. These will be our future work.



(a) (b) (c) (d)

Fig. 8. Flower, 2422 nodes, average degree 12.31. (a) Original map; (b) MDS-MAP; (c) CATL; (d) ACDL.



(a) (b) (c) (d)

Fig. 9. Snake, 2759 nodes, average degree 8.17. (a) Original map; (b) MDS-MAP; (c) CATL (the three beacon nodes are marked in dark red.); (d) ACDL.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 60803115, Grant 60873127, Grant 61073147, and Grant 61173120; by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the Fundamental Research Funds for the Central Universities under Grant 2011QN014; by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the Youth Chenguang Project of Wuhan City under Grant 201050231080; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). Dan Wang's work was supported by Hong Kong PolyU/A-PJ19, A-PK95, A-PL23, 4-BC01, 4-BC02, 4-BC03, and RGC/GRF PolyU 5305/08E. The corresponding author of this paper is Hongbo Jiang.

REFERENCES

[1] I. Borg and P. Groenen. *Modern Multidimensional Scaling, Theory and Applications*. New York, Springer-Verlag, 2005.

[2] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Comm. Magazine*, 7(5), 2000.

[3] D. Dong, Y. Liu, and X. Liao. Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods. In *Proceedings of ACM MOBIHOC*, 2009.

[4] S. P. Fekete, A. Kroller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *Proceedings of the Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[5] H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan. Continuous multi-dimensional top-k query processing in sensor networks. In *Proceedings of IEEE INFOCOM*, 2011.

[6] H. Jiang, S. Jin, and C. Wang. Parameter-based data aggregation for statistical information extraction in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 59(8), 2010.

[7] H. Jiang, S. Jin, and C. Wang. Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 2011.

[8] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu. Connectivity-based skeleton extraction in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(5), 2010.

[9] M. Jin, S. Xia, H. Wu, and X. Gu. Scalable and fully distributed localization with mere connectivity. In *Proceedings of IEEE INFOCOM*, 2011.

[10] S. Lederer, Y. Wang, and J. Gao. Connectivity-based localization of large scale sensor networks with complex shape. In *Proceedings of IEEE INFOCOM*, 2008.

[11] M. Li and Y. Liu. Rendered path: Range-free localization in anisotropic sensor networks with holes. In *Proceedings of ACM MOBICOM*, 2007.

[12] Y. Liu, Z. Yang, X. Wang, and L. Jian. Location, localization, and localizability. *Jounral of Computer Science and Technology*, 25(2), 2010.

[13] O. Saukh, R. Sauter, M. Gauger, P. J. Marron, and K. Rothernel. On boundary recognition without location information in wireless sensor networks. In *Proceedings of ACM/IEEE IPSN*, 2008.

[14] Y. Shang and W. Ruml. Improved mds-based localization. In *Proceedings of IEEE INFOCOM*, 2004.

[15] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity information. In *Proceedings of ACM MOBIHOC*, 2003.

[16] A. Shoshan and U. Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Proceedings of IEEE FOCS*, 1999.

[17] G. Tan, H. Jiang, S. Zhang, and A. Kermarrec. Connectivity-based and anchor-free localization in large-scale 2d/3d sensor networks. In *Proceedings of ACM MOBIHOC*, 2010.

[18] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proceedings of ACM MOBICOM*, 2006.

[19] Y. Wang, S. Lederer, and J. Gao. Connectivity-based sensor network localization with incremental delaunay refinement method. In *Proceedings of IEEE INFOCOM*, 2009.