Scalable Sensor Localization via Ball-Decomposition Algorithm

Yasushi Kawase*
Tokyo Institute of Technology
Tokyo, Japan
kawase.y.ab@m.titech.ac.jp

Takanori Maehara*
Shizuoka University
Shizuoka, Japan
maehara.takanori@shizuoka.ac.jp

Ken-ichi Kawarabayashi
National Institute of Informatics
JST, ERATO, Kawarabayashi Project
Tokyo, Japan
k_keniti@nii.ac.jp

***Abstract*—We consider a wireless sensor network localization problem, with range-free and anchor-free settings, i.e., each sensor can only detect which sensors are in the neighbor. We observe issues with existing algorithms that cause inaccurate localization, and propose a new decomposition-based algorithm for resolving these issues.**

The proposed algorithm consists of three parts: (1) decomposition of a sensor network into small networks that may have large overlap with other small networks by a randomized ball-decomposition algorithm; (2) localization of each network by MDS-MAP and physical simulation-based local refinement; (3) gluing of small networks by a divide-and-conquer algorithm. Intuitively, our algorithm finds a good localization because it finds almost optimal localization for each small graph, and moreover, it glues them together optimally.

We conduct computational experiments in both synthetic and realistic setting. The proposed algorithm is more accurate, efficient, and memory-saving than existing algorithms. In fact, it accurately localizes 200,000 sensors on European region in 3 hours, whereas other existing algorithms scale only up to 10,000 sensors. Thus, the algorithm can handle problem sizes several dozen times as large as existing algorithms can.

I. Introduction

A. WSN localization problem

Wireless sensor networks (WSNs) are used in many applications such as environmental monitoring [1], vehicle tracking [2], health [3], and smart environments [4]. In many cases, the actual locations of the sensors are unknown, however, location information is necessary for the underlying applications. Imagine a network of sensors scattered across a large building or a forest. Typical tasks for such networks are sending a message to a sensor at a given location (without knowing which sensors are there, or even how to get there), retrieving data from sensors in a given region, and finding other sensors with data in a given range. Most of these tasks require knowing the positions of the sensors, or at least the relative positions among them.

WSNs provide information on spatial temporal characteristics of the physical world. Hence it is important to associate sensed data with locations to produce geographically meaningful data. Location information also supports many fundamental network services, including network routing, topology control, coverage, boundary detection, clustering, etc. Therefore, localization has attracted significant attention and interest.

In this paper, we consider the *WSN localization problem*, which determines the *location* (*position*) of each sensor from the local information of each sensor. More precisely, we consider the *range-free* and *anchor-free* localization problem, which assumes that each sensor only detects the sensors in the neighbor and we do not know the exact locations of any sensors. In other words, we assume that each sensor are not equipped neither ultrasonic sensors or GPS. This is the least informative and the most difficult setting. Our WSN localization problem is described as a graph embedding problem. Let $G = (V, E)$ be an undirected graph. Here V denotes a set of sensors and E denotes the topology of a network, i.e., sensors i and j can *communicate* if and only if $(i, j) \in E$. We refer to a map $x : V \to \mathbb{R}^d$ as a *localization* of G in d-dimensional Euclidean space. The WSN localization problem is described as follows.

Problem 1 (WSN localization problem)**.** Given an undirected graph $G = (V, E)$, find a localization $x : V \to \mathbb{R}^d$ such that $\|x(i) - x(j)\| \leq 1$ if and only if $(i, j) \in E$, where $\|\cdot\|$ denotes the Euclidean norm.

This problem is also known as the *unit disk graph realization problem*, and is known to be NP-hard [5]. Thus, we attempt to find a heuristic solution. See [6] for a survey of localization problems.

B. Related work

The basic method for solving our WSN localization problem is *MDS-MAP*, which is proposed by Shang *et al.* [7]. It applies multidimensional scaling (MDS) [8] to the hop count distance (the smallest number of edges between the vertices) of a given topology. We shall review this algorithm in Section II.

The quality of a localization obtained by MDS-MAP depends on the difference between the hop count distance and the Euclidean distance. If sensors are distributed in a convex region, these two distances might be close and hence it gives a good quality localization. However, in anisotropic networks with non-convex regions, the shortest path may be dramatically bent [9]. Thus the hop count distance would significantly be different from the Euclidean distance and hence it produces a poor quality localization.

* Supported by JST, ERATO, Kawarabayashi Large Graph Project

ISBN 978-3-901882-68-5 © 2015 IFIP

To solve this issue, some extensions of MDS-MAP have been proposed. Shang and Ruml proposed *MDS-MAP(P)* [10], which stands for MDS-MAP using patches of relative maps. MDS-MAP(P) first constructs a local map for each sensor, then it merges local maps one-by-one. Finally it refines the localization by minimizing the difference of the obtained distance and the hop count distance. Liu *et al.* proposed the approximate convex decomposition based method, called *ACDL* [11]. ACDL first decomposes a network into several convex parts by identifying "concave" vertices using the boundary detection algorithm [12]. Then it applies MDS-MAP for each part of the decomposition. The obtained localizations are expected to have good quality since the hop count distance provides a good approximation of the Euclidean distance on each part. Finally it unifies the localizations of all parts of the decomposition.

Localization algorithms based on neural networks have also been proposed [13], [14]. The basic idea is to use non-linear mapping techniques and neural network models, such as self-organizing map (*SOM*). These ideas give rise to dimension reduction of multidimensional data sets, yielding coordinates of sensors that preserve the distance between the data points of the input space and output space (in a 2D plane) as much as possible. Note that the SOM method only finds the relative location even for scaling. Thus we must estimate appropriate scaling using some other method for localization.

There are some other techniques have been employed, such as graph rigidity [15], [16], Ricci flow [17], and semidefinite programming [18], [19]. Note that there are many studies of range-based or anchor-based localization problems, but we do not discuss them in this paper.

C. Contributions

We observe the following two issues in the existing algorithms (particularly MDS-MAP):

1) Existing algorithms use the hop count distance. This is problematic for two reasons. First, the hop count distance does not approximate the Euclidean distance for complicated shapes. Second, the hop count distance takes discrete values, which also causes inaccurate localization.
2) The time and space complexities are expensive. Many existing algorithms scale only up to a few thousand sensors.

In Section II, we discuss the above issues for MDS-MAP and give several examples. In consideration of these issues, we propose an algorithm that overcomes them. Our algorithm consists of three parts.

(A) **Decomposition**: Decompose the original graph into small diameter graphs that *possibly have large overlap* with other small graphs.
(B) **Localization**: Localize each small graph by MDS-MAP with a *local refinement* based on physical simulation.
(C) **Gluing**: Glue small graphs by a divide-and-conquer algorithm with minimizing least square errors.

Intuitively, our algorithm finds a good localization because it finds almost optimal localization for each small graph, and it glues them together optimally. Note that, this three-step framework has some similarity with ACDL [11]; however, our method has the following two major improvements:

1) In decomposition, we allow the decomposed graphs to have *large overlap*. This greatly simplifies decomposition, and improves the accuracy and robustness when gluing the decomposed graphs together. Additionally, we can obtain a small decomposition.
2) In localization, our algorithm employs a *local refinement* that reduces the discreteness issue. This refinement is based on a simple physical simulation. Thus our algorithm finds much better localization for each small graph.

In Section III, we discuss each point in more details. The proposed algorithm has many appealing aspects.

1) The proposed algorithm is simple and easy to implement. It does not require big machineries such as boundary detection, graph rigidity, Ricci flow, or semidefinite programming. It only requires a naive breadth first search (BFS), MDS-MAP, and a simple physical simulation.
2) Compared to other algorithms (MDS-MAP, MDS-MAP(P), ACDL, and SOM), the proposed algorithm yields accurate localization. Our experiments show that our algorithm can obtain a much better graph layout than any other algorithm, even for complicated shapes.
3) The proposed algorithm scales up to 200,000 vertices (distributed on European region) in 3 hours, whereas no existing algorithms scale up to such instances. Although the worst case for the time complexity is equal to that of MDS-MAP, the proposed algorithm runs much faster than MDS-MAP. Our experiments shows that the time complexity of our algorithm is subquadratic whereas MDS-MAP is quadratic. Moreover, the space complexity of our algorithm is linear whereas MDS-MAP is quadratic.

Our algorithm can be applied to localization in more than two dimensional space. However, to simplify the notation, we mainly consider the two dimensional case. In our experiment, we give a result of three dimensional localization. The algorithm can also be extended to range-based or anchor-based localization problems.

The rest of the paper is organized as follows. The MDS-MAP algorithm is reviewed in Section II. The proposed algorithm is given in Section III. Thorough experiments are performed in Section IV, and Section V concludes the paper.

II. MDS-MAP

In this section, we review *MDS-MAP* [7], which is the basic algorithm for the range-free and anchor-free WSN localization problem. We also discuss several issues with this algorithm that have motivated our research.

MDS-MAP is based on multidimensional scaling (*MDS*), which is a standard technique in statistics [8]. Thus we first describe MDS. Suppose that we have Euclidean distance $\hat{d}_{ij}$ for all pairs of vertices i and j. Then, the inner product matrix

Algorithm 1 MDS-MAP.

```
1: Compute the hop count distances $d_{ij}$ for all $i, j \in V$
2: Compute the inner-product matrix $B_{ij} := d_{ij}^2 - d_{io}^2 - d_{jo}^2$
3: Calculate the two largest eigenvalues $\lambda_1, \lambda_2$ and the corresponding eigenvectors $q_1, q_2$
4: Return $x(i) = (\sqrt{\lambda_1} q_1(i), \sqrt{\lambda_2} q_2(i))$ as a localization
```

$B = (B_{ij})$ is obtained by $B_{ij} = \hat{d}_{ij}^2 - \hat{d}_{io}^2 - \hat{d}_{jo}^2$ where o is a specified vertex . Let $x : V \to \mathbb{R}^2$ be the exact localization. Then, B is also expressed as $B_{ij} = x(i)^\top x(j)$. Therefore the localization x is recovered from the eigenvalue decomposition of B as $x(i) = (\sqrt{\lambda_1} q_1(i), \sqrt{\lambda_2} q_2(i))$, where λ_1, λ_2 and q_1, q_2 are the two largest eigenvalues and the corresponding eigenvectors of B, respectively.

Let us consider MDS-MAP for the WSN localization problem. Since we do not have the Euclidean distance for each pair of vertices, we must estimate the Euclidean distance to apply MDS. MDS-MAP approximates Euclidean distance by the hop count distance (the smallest number of edges between two vertices) and applies MDS to obtain localization. The detailed procedure is shown in Algorithm 1. The time complexity of MDS-MAP is $O(|V||E|)$, which is dominated by computing all-pairs hop count distances by BFS $|V|$ times. The space complexity of MDS-MAP is $O(|V|^2)$ since it must hold the all-pairs distances. The accuracy of MDS-MAP is analyzed by Oh *et al.* [20]. They showed that, in a square region, the average localization error is typically $O(1)$.

There are three issues of MDS-MAP that have motivated our research:

1) When a shape is complicated, the hop count distance is far from the Euclidean distance. Thus MDS-MAP produces inaccurate localization [10], [11]; see Figure 1a.
2) Even if a shape is simple, MDS-MAP often produces *biased* localization due to the discreteness of the hop count distance. Specifically an obtained localization may have a striped pattern; see Figure 1b.
3) The time and space complexities of MDS-MAP are $O(|V||E|)$ and $O(|V|^2)$, respectively. This is too expensive for application in real networks.

In the next section, we show that our proposed algorithm solves issues 1 and 3 by a decomposition approach, and issue 2 by local refinement.

III. Our algorithm

As mentioned in Introduction, our algorithm consists of the three steps, **Decomposition**, **Localization**, and **Gluing**. Let us look at more details for each step.

A. Decomposition

A family of vertices $\mathcal{S} \subseteq 2^V$ is said to be a *cover* of V if $\bigcup_{S \in \mathcal{S}} S = V$. Each element $S \in \mathcal{S}$ is called a *sheet*. In the decomposition part, to deal with a complicated-shaped (e.g., non-convex) graph, we decompose a graph into a set of simple-shaped sheets.

Let $B_r(i) := \{j \in V \mid d(i,j) \leq r\}$ be a *ball* of radius r centered at $i \in V$. A ball $B_r(i)$ can be obtained by r steps breadth first search (BFS) starting from i. The proposed algorithm constructs a cover of a graph using small balls by *repeated random sampling*. The detailed procedure is described in Algorithm 2. Note that each sheet obtained by this algorithm is not necessarily simple due to the complicated underlying shape. However, each ball of small radius is relatively easy to localize because it does not contain a pair of vertices with long hop count distance. Furthermore, if some sheets cause incorrect localization, this effect is negligible because each sheet is typically small and most sheets are simple.

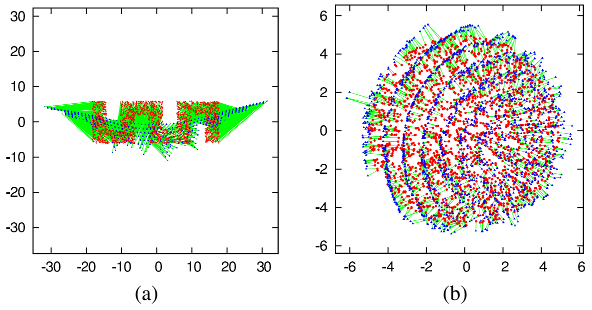


Fig. 1: Localization obtained by MDS-MAP. The red and blue points are the original and obtained locations, and green lines denote errors.

Algorithm 2 Decomposition.

```
1: Initialize $U := V$, $\mathcal{S} := \emptyset$
2: while $U$ is not empty
3:     Pick $i \in U$ at random
4:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{B_r(i)\}$ and $U \leftarrow U \setminus B_{r-1}(i)$
```

Typical decompositions for some graphs are shown in Figure 2. It should be noted that in Figure 2 the radius of a ball is set to $r = 6$ to illustrate decomposition clearly; however in later experiments we set $r = 2$, which gives a better shape in practice (see Section IV-B). Our decomposition approach can be applied to a graph with vertices that are not uniformly distributed, as shown in Figure 2b.

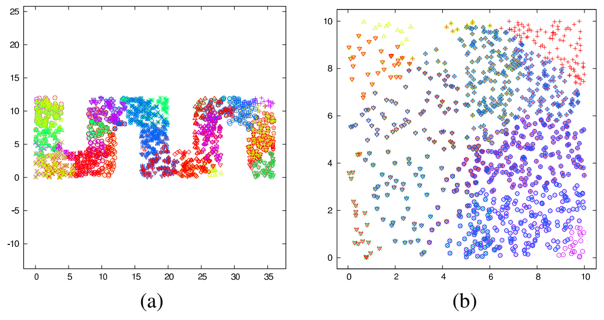


Fig. 2: Typical decomposition of graphs obtained by Algorithm 2. The vertices in the same sheet are the same color and mark.

This decomposition approach provides not only more accurate localization but also a scalable algorithm. In practice, we may assume that the maximum degree of a graph is bounded by a small constant (i.e., each sensor communicates with a few sensors). Thus the number of vertices in each ball is $O(1)$.

Therefore the number of balls is $O(|V|)$, and hence the time complexity of decomposition is $O(|V|)$.

It should be noted that we allow each sheet to have large overlap with other sheets (cf. ACDL [11] does not allow such overlap). This greatly simplifies the decomposition procedure from the decomposition part of ACDL that requires a boundary detection and a convex decomposition procedure. In addition we can get a smaller decomposition and this reduces time and space complexities.

B. Localization

In the localization part, we find a local map of each sheet in our cover. Since each sheet is a ball of small radius, MDS-MAP gives relatively accurate localization; however, as shown in Figure 1b, the result is somewhat biased. Therefore, in this part, we improve MDS-MAP results by using local refinement.

Let $S = B_r(o)$ be a sheet in our cover obtained by the decomposition. Let $G(S) = (S, E(S))$ be the graph induced by S. A localization x of S should satisfy the following constraint.

$$(i, j) \in E(S) \iff \|x(i) - x(j)\| \leq 1. \tag{1}$$

The proposed algorithm improves the initial localization x, which is obtained by MDS-MAP, to satisfy this constraint. To achieve this, we introduce the following potential function.

$$\phi(x) := \frac{k}{2} \sum_{(i,j) \in C(S)} (1 - \|x(i) - x(j)\|)^2 \tag{2}$$

where $C(S)$ is a set of pairs of vertices that *violates* constraint (1) and $k > 0$ is a stiffness parameter. Note that $\phi(x) = 0$ indicates that there are no violated pairs, i.e., $C(S) = \emptyset$. Thus, we can find a good localization by minimizing this potential function.

We minimize this potential function by a force directed method, which is a standard technique in graph drawing [21] and dates back to the seminal work of Tutte [22]. There are many types of force directed methods [21]–[23], and the following method is the simplest *spring model* with friction.

Let us regard vertex i as a (moving) point in the plane with *position* $x(i)$ and *velocity* $v(i)$. Our *physical system* of a force directed method is described as follows:

1) A *spring* of stiffness k is placed between pairs of vertices in $C(S)$. A spring exerts a force opposite to the displacement.
2) *Frictional forces* act to vertices as $-\gamma v(i)$ where $v(i)$ is velocity of a vertex i.

Then the *potential energy* of this system coincides with $\phi(x)$. The force directed method simulates this system up to time T with small time step Δt. We set $k = 2$, $\gamma = 0.1$, $\Delta t = 0.1$, and $T = 100$. When a system is in a steady state, the potential function ϕ attains a local minimum. Note that since ϕ is non-convex, a local minimum is not necessarily the global minimum. The force directed method is shown in Algorithm 3.

Algorithm 3 Force directed method.

```
1: Initialize $v(i) := 0$ ($\forall i \in S$)
2: for $t = 0$ to $T$ by $\Delta t$ do
3:     Compute $F(i)$ ($\forall i \in S$) by (3)
4:     for $i \in S$ do
5:         $v(i) \leftarrow (1 - \gamma \Delta t) v(i) + F(i) \Delta t$
6:         $x(i) \leftarrow x(i) + v(i) \Delta t$
```

Here, a force $F(i) = -\partial \phi / \partial x(i)$ to a vertex i, which is used in the algorithm,is given by

$$F(i) = k \sum_{(i,j) \in C(S)} (1 - \|x(i) - x(j)\|) \frac{x(i) - x(j)}{\|x(i) - x(j)\|}. \tag{3}$$

The complexity of the localization part is $O(|S|^2)$, because it first applies MDS-MAP, which requires $O(|S|^2)$, and the local refinement part requires $O(|S|^2)$ since the number of iterations of the force directed method is estimated as $O(|S|)$ [24] and the complexity per iteration is $O(|S|)$, which is the number of violated pairs (see also Remark 2 below). As we decompose a graph to small balls, typically $O(1)$, the localization requires only $O(1)$ time per ball for the time complexity

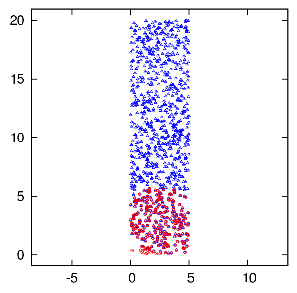
Typical improvement by our refinement is shown in Figures 3c and 3d. Prior to the refinement, a localization obtained by MDS-MAP has 1806 violated pairs, 0.485 average localization error, and 1.11 maximum localization error (see Section IV for the definitions of these criteria). However after the refinement, the localization has no violated pairs, 0.0763 average localization error, and 0.527 maximum localization error. From this example, it can be seen that our physical-based refinement improves the average localization error by a factor six, and the maximum localization error by a factor two. The number of iterations in this refinement is 104.

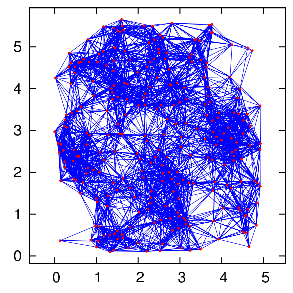
Remark 2. In fact, the number of iterations of force directed method is not well understood yet. Tunkelang [24] experimentally showed the number of iterations is proportional to the number of vertices when we start from random initial location. In our case, since we start from the initial solution obtained by MDS-MAP, we can expect that the number of iterations is smaller than $O(|V|)$. We here experimentally verify this.

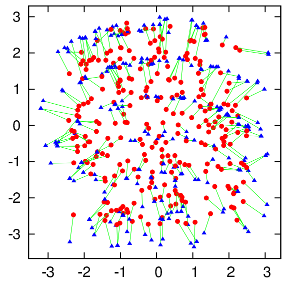
Let us consider randomly generated circle shaped graphs having $|V| = 50$, 100, 200, and 400 vertices. We then apply our local refinement algorithm for these graphs and measure the potential for each iteration. The result is shown in Figure 4 and typical loci of convergence are shown in Figure 5. This shows that the initial solution obtained by MDS-MAP accelerates the convergence. As shown in Table I later, the typical size of each sheet is about a few hundreds. Hence our refinement algorithm converges in about 100–200 iterations.

C. Gluing

Finally, we glue local maps together to construct a global localization. We first show how to glue two sheets. Let S_1, S_2 be two overlapping sheets, and let x_1, x_2 be their corresponding localizations, respectively. To construct a localization of


(a) Decomposition of an I-shape graph with 1000 points, where one component is marked with red.


(b) Original graph of the marked component.


(c) Result of MDS-MAP: potential 93.94, #violate 1806, avg LE 0.485, max LE 1.11.

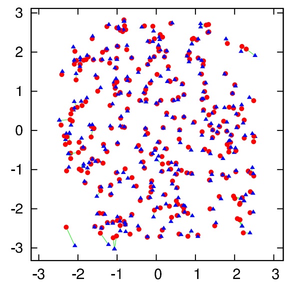

(d) After local refinement: potential 0, #violate 0, avg LE 0.0763, max LE 0.527.

Fig. 3: Typical improvement by Algorithm 3.

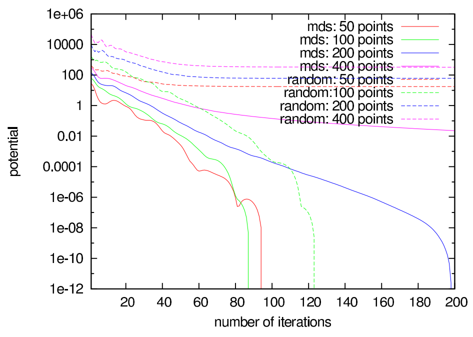


Fig. 4: The number of iterations versus potential value in localization part. We use a circle-shaped graph with constant density (20 sensors per unit circle). Solid lines are for initial locations obtained by MDS-MAP, and the dotted lines are for random initial locations.

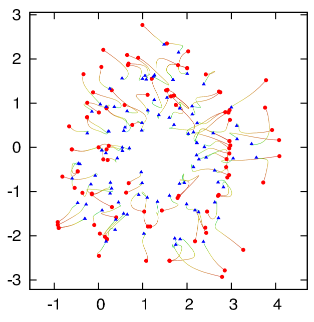
$S_1 \cup S_2$, we first solve the following least squares problem.

$$\min_{P,b} \sum_{i \in S_1 \cap S_2} \| x_1(i) - T_{P,b}(x_2(i)) \|, \tag{4}$$

where $T_{P,b}(x) := Px + b$ is an *Euclidean isometry*, i.e., P is a 2×2 orthogonal matrix and b is a two-dimensional vector. We then construct the localization x_{12} of $S_1 \cup S_2$ as follows.

$$x_{12}(i) = \begin{cases} x_1(i), & i \in S_1, \\ T_{P,b}(x_2(i)), & i \in S_2 \setminus S_1. \end{cases} \tag{5}$$

The problem (4) is known as *Orthogonal Procrustes problem* [25], which is a classical problem in statistics and computational geometry. A closed form solution for this problem is as follows. For $k = 1$ and 2, let $\bar{x}_k$ be the center


(a) MDS-MAP

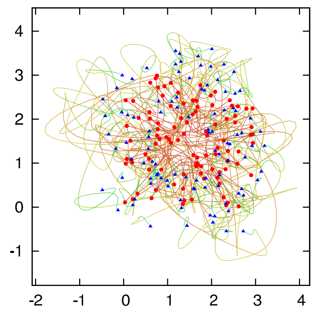

(b) random

Fig. 5: Loci of convergence of local refinement algorithm; (1) starts from MDS-MAP solution and (2) starts from random location. Red points are for initial location and blue points are the converged location. The connecting curves denote the loci.

of gravity (i.e., average) of $\{x(i) \mid i \in S_1 \cap S_2\}$. Let $X_k := [x_k(i) - \bar{x}_k \mid i \in S_1 \cap S_2]$ be a matrix of size $2 \times |S_1 \cap S_2|$. Then, the optimal Euclidean isometry is given by $P = U^\top V$ and $b = \bar{x}_1 - P\bar{x}_2$ where U and V are left and right singular vectors of $X_1^\top X_2$, i.e., U and V are orthogonal matrices that satisfy $U^\top \Sigma V = X_1^\top X_2$ for some diagonal matrix Σ. Once the localization x_{12} is obtained, we refine this map by the force directed method (Algorithm 3). The complete algorithm is shown in Algorithm 4 and the time complexity is expected to be $O(|S_1 \cup S_2|)$.

We then consider how to glue all sheets efficiently. We use the divide-and-conquer framework:

1) divide the sheets into two components,
2) recursively compute localizations for the two components, and
3) glue the two components (by the above procedure).

The issue is that how to divide the sheets. It should satisfy the following two conditions.

a) The overlap of two components is large.
b) The sizes of two components are almost the same, and the overlap of the components is not too large.

Condition a) is important for the accuracy of the localization; there should be an appropriate "margin to paste" between two components. Condition b) is important for the time complexity; if this condition holds, the depth of the recursion is $O(\log |V|)$ and the total complexity becomes $O(|V| \log |V|)$. To satisfy these two conditions, we adopt the following strategy: We pick the (approximately) farthest sheet pair S_1 and S_2, and then divide all sheets into two parts $\mathcal{S}_1$ and $\mathcal{S}_2$ such that each sheet in $\mathcal{S}_i$ is closer to S_i (than S_{2-i}). See Algorithm 5.

Ideally, as mentioned above, our algorithm runs in $O(|V| \log |V|)$ time. However, in practice, the sizes of two components are biased, and hence the time complexity becomes worse; it practically runs in subquadratic time, i.e., $O(|V|^\alpha)$ time for some $1 < \alpha < 2$. Our algorithm requires only $O(|E|) = O(|V|)$ space. These show that our algorithm

Algorithm 4 Glue S_1 and S_2.

```
1: Compute the center $\bar{x}_k$ of $\{x_k(i) \mid i \in S_1 \cap S_2\}$ and $X_k := [x_k(i) - \bar{x}_k \mid i \in S_1 \cap S_2]$ for $k = 1, 2$.
2: Compute the singular value decomposition $U^\top \Sigma V = X_1^\top X_2$
3: Let $P = U^\top V$, $b = \bar{x}_1 - P\bar{x}_2$ and construct $x_{12}$ by (5)
4: Refine $x_{12}$ by the force directed method (Algorithm 3)
```

Algorithm 5 Glue all sheets

```
1: Let $\mathcal{S}$ be sheets to glue.
2: if $|\mathcal{S}| = 1$ return localization by MDS-MAP and the force directed method (Algorithm 3)
3: else
4:     Construct sheets graph $G_\mathcal{S} = (\mathcal{S}, E_\mathcal{S})$ and let $d$ denote the hop count distance in $G_\mathcal{S}$, where $E_\mathcal{S} := \{(S_1, S_2) \in \mathcal{S}^2 \mid S_1 \cap S_2 \neq \emptyset\}$.
5:     Pick $S \in \mathcal{S}$ at random.
6:     Let $S_1 \in \mathcal{S}$ be a farthest sheet from $S$ in $G$.
7:     Let $S_2 \in \mathcal{S}$ be a farthest sheet from $S_1$ in $G$.
8:     $\mathcal{S}_1 := \{S \in \mathcal{S} \mid d(S, S_1) \leq d(S, S_2)\}$.
9:     $\mathcal{S}_2 := \{S \in \mathcal{S} \mid d(S, S_1) > d(S, S_2)\}$.
10:    Compute localization of $\mathcal{S}_1$ and $\mathcal{S}_2$, recursively.
11:    Return a localization of $\mathcal{S}$ by Algorithm 4.
```

is scalable for large scale WSNs. See Section IV for more details.

IV. Experiments

Here we present experimental results for the proposed algorithm. We first show our results in Section IV-A, and then we compare them with existing algorithms in Section IV-C

All experiments were performed on a machine with Intel Xeon E5-2690@2.90GHz processor. Here, the quality is evaluated by the following three criteria: (1) The localization error; (2) The number of violated pairs; (3) The value of the potential function. The localization error is the sum of the Euclidean distances between the obtained location x and the exact location x^*. It should be noted that the exact locations of the sensors were known in the experiments. The range-free and anchor-free localization problems only require a relative location; thus we transform a location by the optimal isometry.

$$\mathrm{LE} := \min_{P,b} \sum_{i \in V} \|x^*(i) - T_{P,b}(x(i))\|.$$

This optimization problem can be solved by the method discussed in Section III-C. Note that the localization error is widely used in the literature [11], [13]; however, large localization error does not always imply poor localization quality because the localization may not be uniquely determined under given connectivity information. In such cases, the number of pairs that violate constraint (1) is often more suitable measure. The value of the potential function (2) is a quantitative version of the number of violated pairs.

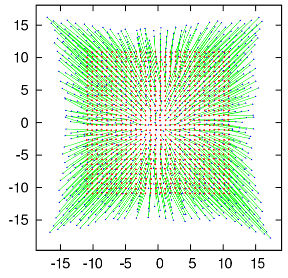
A. Our results

Random placement on a region: We here validate our algorithm for randomly generated graphs. We first specify some region, and then generate sensors uniformly and randomly on the region. The qualities of the obtained localizations are shown in Table I. and the obtained localizations are shown in Figures 9 and 10.

Our algorithm produces good localization for all the experimented graphs. Note that for "⊞-shape 1500," the number of violated pairs is zero, and the maximum localization error is about 2. The reason for this is, as mentioned above, there are many possibilities for feasible localization that satisfy a given topology.

TABLE I: Computation time and quality of localization of randomly placed vertices.

Topology	$\|V\|$	avg degree	time (s)	#sheets	sheets size	potential	violate paris	avg LE	max LE
circle	300	5.49	0.68	45	29.0	2.24e-10	10	0.416	1.33
	1K	18.5	3.22	53	111	0	0	0.0540	0.247
	2K	36.4	15.3	58	231	0	0	0.0232	0.129
snake	2.5K	7.38	9.78	298	40.1	0	0	0.510	1.28
	5K	22.8	23.1	221	140	0	0	0.337	0.835
	10K	46.1	122	221	288	0	0	0.235	1.15
O-shape	300	7.40	0.38	33	35.2	2.93e-7	4	0.597	1.57
	500	12.9	0.89	38	63.0	0	0	0.231	0.640
	1K	24.9	3.94	40	132	0	0	0.231	0.427
⊞-shape	1500	6.47	6.71	200	34.4	0	0	0.625	1.96
	5K	21.3	22.9	233	132	0	0	0.0777	0.294
	10K	42.7	173	243	278	4.12e-7	270	0.0770	0.280


(a) MDS-MAP: potential 113.4, #violate 1581, avg LE 3.77, max LE 9.35.

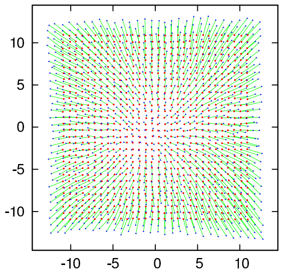
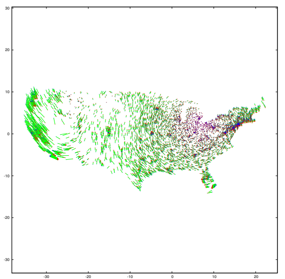

(b) Proposed algorithm: potential 0.71e-9, #violate 3, avg LE 1.38, max LE 2.94.

Fig. 6: Results for grid graphs.

Grid placement: Here, we demonstrate our algorithm for graphs of grid placement which is an example of a realistic placement. As Shang *et al.* [7] showed, with the same connectivity level, MDS-MAP obtains a better location for grid graphs than that for random graphs. The proposed algorithm can further improve this result. Here, a graph is generated as follows. Consider a $k \times k$ grid of width h (i.e., the distance between adjacent sensors). Each vertex is placed on the point of the grid with 10% placement error. With $h = 0.75$, the average connectivity is slightly greater than four.

The results are presented in Figure 6, which show that the proposed algorithm significantly improves MDS-MAP. Our algorithm finds nearly the exact solution relative to the number of violations. This improvement can be explained as follows. Consider the vertices on the lower-left corner and the upper-right corner. The hop count distance between these vertices is $2k$ and the Euclidean distance is $h\sqrt{2}k$. This implies that, if a grid gets larger, the difference between hop count distance and Euclidean distance gets larger. However, since our algorithm


(a) $33,108$ towns and cities in the US: potential 0.03, #violate 4441, avg LE 0.38, max LE 2.24.

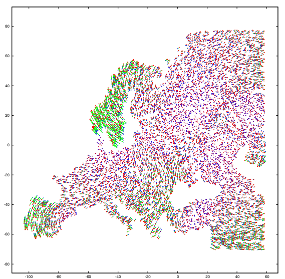
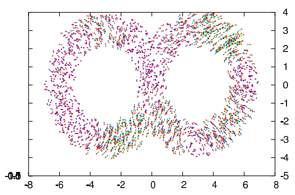

(b) 200,000 random locations on Europe: potential 0.65, #violate 37473, avg LE 0.77, max LE 3.92.

Fig. 7: Results on geographically generated graphs.


(a) x-axis is rotated $0°$ and z-axis is rotated $0°$.

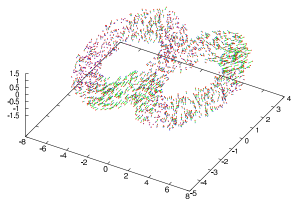

(b) x-axis is rotated $30°$ and z-axis is rotated $30°$.

Fig. 8: Result for random placement of 2000 sensors on a double torus shape: potential 0, #violate 0, avg. LE 0.181, max LE 0.300.

only localizes small diameter sheets by MDS-MAP, it does not have this problem. Note that, for this graph, ACDL coincides with MDS-MAP because ACDL determines a boundary node as convex. Therefore, this simple example demonstrates that the proposed algorithm outperforms ACDL as well.

Realistic placement: We compute localization for a graph of $33,108$ towns and cities in the United States, and a graph of $200,000$ sensors randomly generated on Europe shape region. The results are shown in Figures 7a and 7b. The proposed algorithm outputs the result for the graph of $200,000$ in 3 hours.

Localization on three dimensional networks: For some applications, such as underwater sensor networks and smart buildings, dimensionality plays a crucial role [26]. The proposed algorithm can be extended to a three dimensional localization problem in a straightforward manner. We here demonstrate three dimensional version of our algorithm for a randomly generated graph on a double torus shape. The result is shown in Figure 8.

B. Radius of balls

We compare localizations obtained by our algorithm when the radius of ball r is $1, 2, 3$ and ∞. Here, $r = \infty$ means that we do not decompose the graph. We use three circle-shaped graphs with different densities. The results are presented in Table II, which show that $r = 2$ gives better computational time and localization errors for the graphs. We performed the same experiments for other shaped graphs and we obtained similar results.

TABLE II: Computational time in seconds and localization errors for circle-shaped graphs.

$\lvert V\rvert$	avg. degree	Measure	$r=1$	$r=2$	$r=3$	$r=\infty$
300	5.49	#violate	**0**	10	1	252
		avg LE	**0.32**	0.42	0.50	0.69
		max LE	**0.98**	1.33	1.37	1.72
		time(s)	1.77	**0.68**	1.38	0.81
1000	18.5	#violate	**0**	**0**	**0**	167
		avg LE	6.1e-2	**5.0e-2**	6.4e-2	0.15
		max LE	0.26	**0.23**	0.24	0.39
		time(s)	24.3	2.8	**2.4**	3.7
5000	182	#violate	**0**	**0**	**0**	**0**
		avg LE	8.3e-3	**8.1e-3**	1.1e-2	1.0e-2
		max LE	**5.0e-2**	**5.0e-2**	7.6e-2	6.1e-2
		time(s)	3577.7	**128.4**	150.1	229.2

C. Comparison with existing algorithms

Here, we compare the proposed algorithm with some existing algorithms. We implement the following existing algorithms: (1) MDS-MAP by Shang *et al.* [7]. (2) MDS-MAP(P) by Shang and Ruml [10]. (3) ACDL by Li *et al.* [9]. Since the implementation of convex decomposition (particularly, boundary detection) in ACDL is complicated, we manually decompose the domain into some convex parts. (4) SOM by Giorgetti *et al.* [13]. Since SOM only finds the relative location, we find the optimal scaling to minimize localization error by solving an optimization problem.

We compare the quality of localization and computational time of the algorithms. The results are shown in Figures 9 and 10 and summarized in Table III.

As can be seen here, the proposed algorithm outperforms all other algorithms. In particular, the number of violations is significantly lower. A brief comparison with each algorithm is given as follows.

1) MDS-MAP produces a poorer localization than other algorithms. In particular, for the snake shape, since the hop count distance does not approximate Euclidean distance, its localization is quite poor.
2) MDS-MAP(P) is slightly better than MDS-MAP, however, its performance is still poorer than the proposed algorithm. Furthermore, MDS-MAP(P) is too expensive because it must solve an optimization problem for each merge step.
3) ACDL produces the same localization as MDS-MAP for circle and O-shape graphs, because the convex decompositions of the graphs obtained by ACDL are the same shape as those obtained by MDS-MAP. ACDL outputs a good localization for the snake shape; however it outputs a poor localization for the ⊞-shape. The reason is that gluing of ACDL is very sensitive to the quality of the MDS-MAP for each convex part because the convex parts have only small overlap. In contrast, the proposed algorithm allows to have large overlap for each sheet. Thus it can glue the sheets more robustly.
   In our experiments, we manually give a convex decomposition of the given graphs because convex decomposition is difficult to implement. With good convex decomposition, ACDL is the fastest algorithm. However, the convex

TABLE III: Localization errors; the bold values denote the best results.

Topology	$\|V\|$	Measure	Proposed	MDS-MAP	M.-MAP(P)	ACDL	SOM
circle	300	#violate	**10**	887	806	887	305
		avg LE	0.416	1.29	1.38	1.29	**0.338**
		max LE	1.33	3.90	3.67	3.90	**0.945**
		time(s)	0.68	**0.138**	5.43	**0.138**	0.833
O-shape	300	#violate	**4**	1378	1055	1378	832
		avg LE	**0.597**	2.60	1.72	2.60	0.859
		max LE	**1.57**	4.25	2.76	4.25	2.65
		time(s)	1.129	**0.38**	7.694	**0.144**	0.733
snake	2500	#violate	**0**	29909	8609	9539	10193
		avg LE	**0.510**	9.950	4.77	1.22	11.6
		max LE	**1.28**	27.0	12.0	3.30	34.0
		time(s)	9.78	21.8	6450	**4.00**	22.0
⊞-shape	1500	#violate	**0**	6155	5082	8391	1542
		avg LE	0.625	2.63	3.10	7.27	**0.388**
		max LE	**1.96**	5.58	8.15	20.3	2.23
		time(s)	6.71	4.07	831	**1.81**	8.27

TABLE IV: Computational time in seconds for circle-shaped graphs with constant density (30 sensors per unit circle). '—' denotes that it takes over 1 hour (3600 seconds).

$\|V\|$		1K	4K	7K	10K	40K	70K	100K
time(s)	proposal algorithm	2.44	**14.3**	**35.2**	**51.4**	**498.7**	**1242.0**	**2335.5**
	MDS-MAP, ACDL	**1.35**	93.8	486.7	1463.2	—	—	—
	MDS-MAP(P)	111.0	1628.9	—	—	—	—	—
	SOM	35.1	570.5	—	—	—	—	—

decomposition may be the bottleneck.

4) SOM only finds a relative location even for scaling. Thus we first compute optimal scaling for SOM. In this case, the localization of SOM is better than the existing algorithms for square-like shapes; however, it performs poorly for the snake shape. The proposed algorithm always outperforms SOM.

Computing time and space: We randomly generate circle-shaped graphs with constant density for $|V| = 1{,}000$ to $100{,}000$, and then measure the computational time required to localize each graph. We performed this experiment 10 times to calculate the average computation time. The results are shown in Table IV. The results show that the proposed algorithm scales up to 100,000 vertices in 40 minutes at this setting. The total time complexity is subquadratic of $|V|$ (the exponent is about 1.72), which coincides with our analysis in Section III-C. Our proposed algorithm required only 80MB to localize the graph with 10,000 sensors, where MDS-MAP required 2,739MB. The algorithm required only 910MB even for the graph with 100,000 sensors. Thus, our algorithm can handle problem sizes several dozen times as large as the existing algorithms can.

V. Conclusion

In this paper, we propose a simple and scalable algorithm to localize a WSN using only connectivity. The proposed algorithm consists of three parts: (1) Sensors are decomposed into small sheets that possibly have large overlap with other sheets; (2) MDS-MAP and a physical simulation-based local refinement are performed to obtain a local map of each sheet; (3) A global map is obtained by gluing each local map by divide-and-conquer algorithm. Our algorithm finds a good localization because it finds almost optimal localization for each small graph, and it glues them together optimally. The proposed algorithm is easy to implement, and the obtained localization significantly outperforms existing algorithms in accuracy. Our algorithm produces much accurate localizations than other existing algorithms using similar time and memory even when the input of our algorithm is several dozen times larger than that of other existing algorithms.

References

[1] L. M. L. Oliveira and J. J. P. C. Rodrigues, "Wireless sensor networks: a survey on environmental monitoring," *JCM*, vol. 6, no. 2, pp. 143–151, 2011.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda *et al.*, "A line in the sand: a wireless sensor network for target detection, classification, and tracking," *Comput. Netw.*, vol. 46, no. 5, pp. 605–634, 2004.

[3] M. H. Yaghmaee, N. F. Bahalgardi, and D. A. Adjeroh, "A prioritization based congestion control protocol for healthcare monitoring application in wireless sensor networks," *WPC*, vol. 72, no. 4, pp. 2605–2631, 2013.

[4] C. Herring and S. M. Kaplan, "Component-based software systems for smart environments," *IEEE Personal Commun.*, vol. 7, no. 5, pp. 60–61, 2000.

[5] H. Breu and D. G. Kirkpatrick, "Unit disk graph recognition is NP-hard," *Comput. Geom.*, vol. 9, no. 1, pp. 3–24, 1998.

[6] Y. Liu, Z. Yang, X. Wang, and L. Jian, "Location, localization, and localizability," *JCST*, vol. 25, no. 2, pp. 274–297, 2010.

[7] Y. Shang, W. Ruml, Y. Zhang, and M. P. Fromherz, "Localization from mere connectivity," *MobiHoc*, pp. 201–212, 2003.

[8] T. F. Cox and M. A. Cox, *Multidimensional Scaling*. CRC Press, 2010.

[9] M. Li and Y. Liu, "Rendered path: range-free localization in anisotropic sensor networks with holes," *MobiCom*, pp. 51–62, 2007.

[10] Y. Shang and W. Ruml, "Improved MDS-based localization," *INFOCOM*, vol. 4, pp. 2640–2651, 2004.

[11] W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang, "Approximate convex decomposition based localization in wireless sensor networks," *INFOCOM*, pp. 1853–1861, 2012.

[12] Y. Wang, J. Gao, and J. S. Mitchell, "Boundary recognition in sensor networks by topological methods," *MobiCom*, pp. 122–133, 2006.

[13] G. Giorgetti, S. K. Gupta, and G. Manes, "Wireless localization using self-organizing maps," *IPSN*, pp. 293–302, 2007.

[14] L. Li and T. Kunz, "Localization applying an efficient neural network mapping," *Autonomics*, pp. 1–9, 2007.

[15] S. Lederer, Y. Wang, and J. Gao, "Connectivity-based localization of large-scale sensor networks with complex shape," *TOSN*, vol. 5, 2009.

[16] Y. Wang, S. Lederer, and J. Gao, "Connectivity-based sensor network localization with incremental delaunay refinement method," *INFOCOM*, pp. 2401–2409, 2009.

[17] M. Jin, S. Xia, H. Wu, and X. Gu, "Scalable and fully distributed localization with mere connectivity," *INFOCOM*, pp. 3164–3172, 2011.

[18] P. Biswas, K.-C. Toh, and Y. Ye, "A distributed SDP approach for large-scale noisy anchor-free graph realization with applications to molecular conformation," *SISC*, vol. 30, pp. 1251–1277, 2008.

[19] S. Ji, K.-F. Sze, Z. Zhou, A. M.-C. So, and Y. Ye, "Beyond convex relaxation: A polynomial–time non–convex optimization approach to network localization," *INFOCOM*, 2013.

[20] S. Oh, A. Montanari, and A. Karbasi, "Sensor network localization from local connectivity: Performance analysis for the MDS-MAP algorithm," *ITW*, pp. 1–5, 2010.

[21] I. Tollis, P. Eades, G. Di Battista, and L. Tollis, *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall New York, 1998, vol. 1.

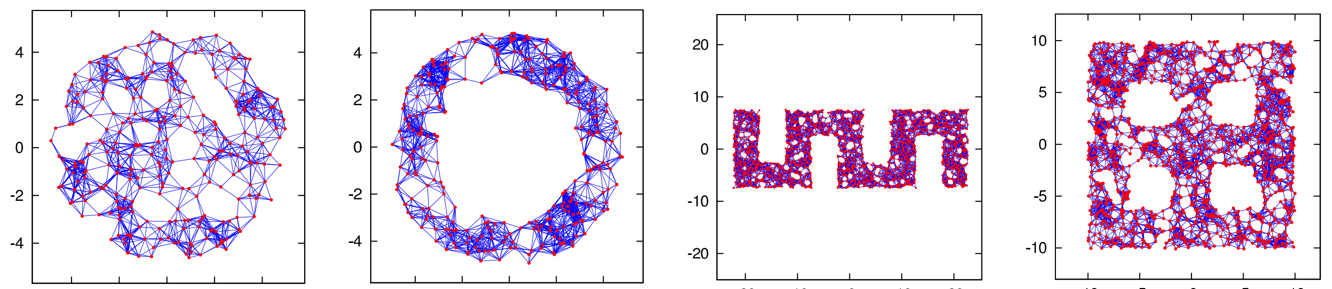
[22] W. T. Tutte, "How to draw a graph," *Proc. London Math. Soc.*, vol. 13, no. 52, pp. 743–768, 1963.

[23] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *SPE*, vol. 21, no. 11, pp. 1129–1164, 1991.

[24] D. Tunkelang, "A numerical optimization approach to general graph drawing," Ph.D. dissertation, IBM, 1999.

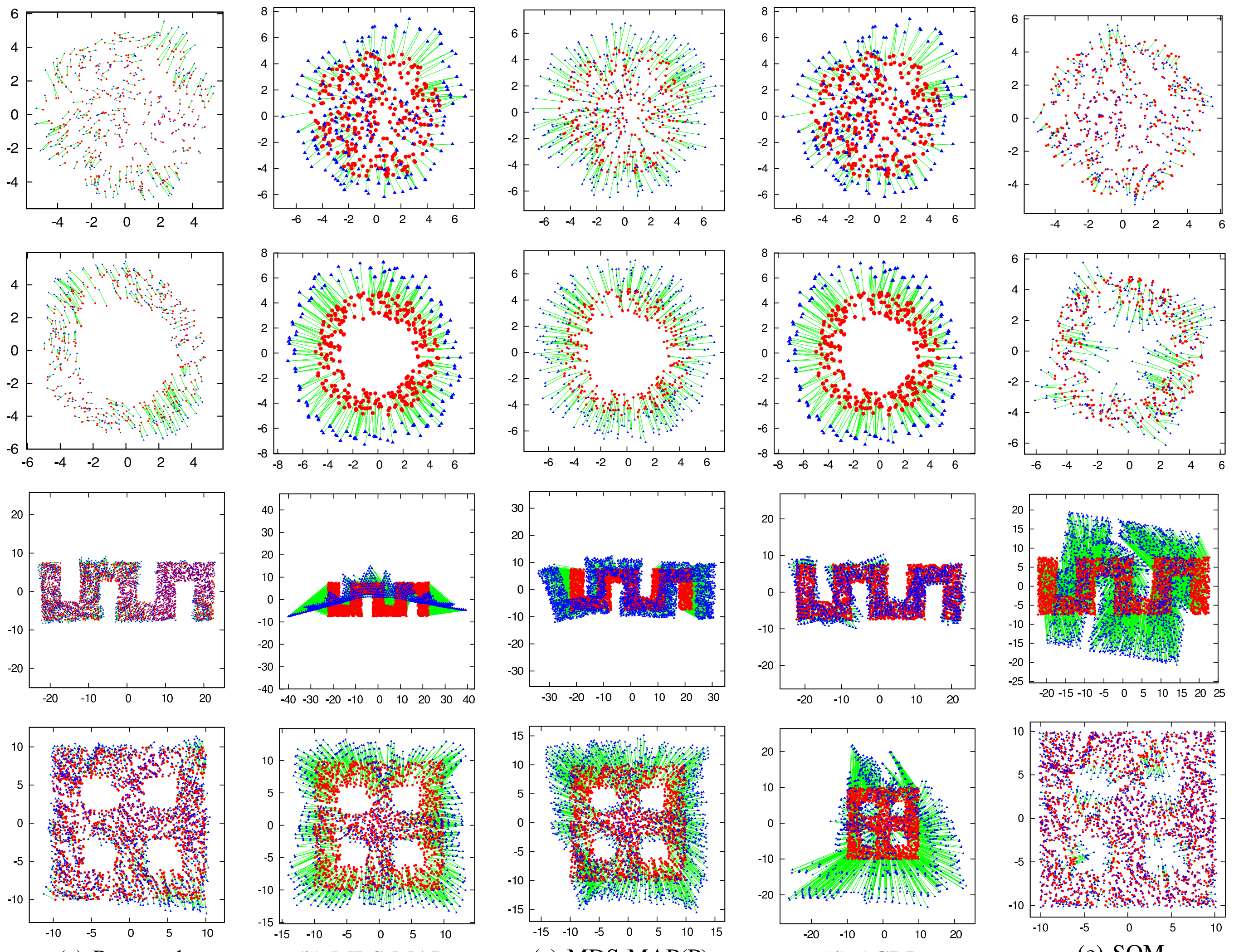
[25] J. C. Gower and G. B. Dijksterhuis, *Procrustes problems*. Oxford University Press, 2004, vol. 3.

[26] G. S. Kuruoglu, M. Erol, and S. Oktug, "Three dimensional localization in wireless sensor networks using the adapted multi-lateration technique considering range measurement errors," *GLOBECOM*, pp. 1–5, 2009.



(a) circle 300 (b) O-shape 300 (c) snake 2500 (d) ⊞-shape 1500

Fig. 9: The original graphs for the comparison of algorithms. The red points denote original location, and the blue lines denote communicating pairs.



(a) Proposed (b) MDS-MAP (c) MDS-MAP(P) (d) ACDL (e) SOM

Fig. 10: Comparison of the algorithms for graphs in Figure 9.