WCS: Weighted Component Stitching for Sparse Network Localization

Tianyuan Sun, Yongcai Wang, *Member, IEEE*, Deying Li, *Member, IEEE*, Zhaoquan Gu, *Member, IEEE*, and Jia Xu, *Member, IEEE*

***Abstract*—Network location is one of the critical issues and a challenge in wireless sensor and ad hoc networks, in particular when networks are sparse. However, even in highly sparse networks, there exist well-connected subgraphs while the distribution of the networks is random. This paper introduces weighted component stitching (WCS) to find redundantly rigid components with high redundant ratios, which can be used to generate reliable local realization. Finding and ranking the redundantly rigid components is an NP-hard problem (a reduction from maximum quasi-clique). Here, we introduce a series of theorems and algorithms to carry out WCS efficiently. More precisely, we prove that each graph has a determinant number of redundantly rigid components, each redundantly rigid component is covered by a set of basic redundant components (BRCs), and each BRC contains one redundant edge. We apply constraints to merge the BRCs to form components with higher redundancy ratio and develop a greedy algorithm to merge BRCs to form locally mostly redundant components (LMRCs). Finally, we give the approximation ratio. The local coordinates of nodes are calculated by optimization in each LMRC and are synchronized with weights to produce the global coordinates of nodes in the network to overcome the sparseness of subgraphs. Extensive experiments demonstrate significant improvements in accuracy (45%–64%) using our WCS method over the state-of-the-art algorithms under various settings of network sparseness and ranging noises.**

***Index Terms*—Network localization, graph realization, component stitching, redundantly rigid, sparse network.**

I. Introduction

Network *localization*, which infers node locations by ranging measurements among nodes is an important problem in ad-hoc and wireless sensor networks. It has attracted great attentions for wide applications in location-based services, monitoring, and surveillance etc. Existing studies investigate network localization mainly from two aspects: (1) *localizability properties* [1]–[8], which investigates the conditions of whether a given network or a subset of nodes can be uniquely localized when the distance measurements are given. It is mainly based on the rigidity theory; (2) *localization algorithms*, which are different kinds of algorithms to calculate node locations based on the distance measurements. In the localization algorithm aspect, three kinds of algorithms have been proposed: 1) optimization-based algorithms [9]–[12]; 2) distributed algorithms [2], [13], [14]; 3) component stitching-based algorithms [15]–[17]. These algorithms will be introduced in Section II.

A key remaining challenge of network localization is the hardness to obtain accurate localization in sparse networks, which are very general application scenarios. The wireless nodes have limited ranging scope, but are generally widely spread in space. In this paper, we consider localization in randomly distributed sparse networks, where nodes have small average degree (i.e., $\leq \Delta$). The network topology calculation is focused, without using anchors.

Existing network localization algorithms generally have good performances in networks with dense edge measurements (enabling the graph to satisfy global rigid condition [1]). But they provide unsatisfactory localization results when the distance measurements are sparse. This is because the non-global rigid graphs cannot be uniquely realized [1], [2]. Because of under-determinant optimization, especially in the case when there are no anchors, the errors in the sparse parts may pollute the optimization of the whole graph.

But well-connected components exist even in highly sparse random networks due to the uneven node distribution. The local coordinates of nodes in the well-connected components can be calculated rather reliably. We therefore propose weighted component stitching (WCS). WCS finds the subgraphs with high redundant ratios and partitions the graph into components ranked by the redundancy ratios. The local coordinates in each component are calculated by local optimization algorithms [10]–[12], and then the local coordinates of nodes are synchronized with weights to generate network coordinates by component rotation and transition. Wang *et al.* [18] study this problem in sparse UAV network for formation tracking.

But we show the problem to find and rank the redundant rigid components is NP-hard, by reducing from the maximum-quasi clique problem. So it is infeasible to find and rank the components with different redundant ratios precisely. We therefore propose theorems and algorithms to approach the

Manuscript received January 3, 2018; revised July 5, 2018; accepted August 10, 2018; approved by IEEE/ACM Transactions on Networking Editor T. He. Date of publication September 13, 2018; date of current version October 15, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 11671400, Grant 61672524, Grant U1636215, and Grant 61572492, in part by the Fundamental Research Funds for the Central University, and in part by the Research Funds of Renmin University of China under Grant 2015030273. *(Corresponding author: Yongcai Wang.)*

T. Sun, Y. Wang, and D. Li are with the Department of Computer Sciences, Renmin University of China, Beijing 100872, China (e-mail: suntyruc@163.com; ycw@ruc.edu.cn; deyingli@ruc.edu.cn).

Z. Gu is with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China (e-mail: zqgu@gzhou.edu.cn).

J. Xu is with The Graduate Center & Hunter College, City University of New York, New York, NY 10065 USA (e-mail: jia.xu@hunter.cuny.edu).

Digital Object Identifier 10.1109/TNET.2018.2866597

weighted component stitching (WCS) efficiently. The detailed technical contributions of this paper are as following:

1) The problem to find and rank the redundant components of different redundant ratio (RR) is proved NP-hard by reducing from the maximum quasi-clique problem.
2) We investigate the distribution of redundant rigid components in a graph and prove Extreme Redundant Components (ERC) are distinct and determinant. All the redundant rigid components must be within the ERCs.
3) We prove each ERC is covered by the union of a set of Basic Redundant Rigid Component (BRCs). Each BRC is 1-redundant rigid. An efficient algorithm to detect the BRCs by bipartite matching is proposed.
4) Necessary and sufficient conditions to merge BRCs to form more redundant components are proposed. An efficient greedy algorithm is developed to merge BRCs to form *Locally Most Reliable Components (LMRC)*, whose approximation ratio is derived.
5) Efficient algorithm for weighted component stitching is designed by iteratively updating rotation matrices of components and global coordinates of nodes. The RRs of different LMRCs are utilized as weights in weighted component stitching.
6) Extensive simulations show dramatical accuracy improvement (45 %-64 %) of the proposed algorithm than the state-of-the-art of component stitching algorithms under different network connectivity and noise level settings.

The remaining parts of this paper are organized as following. Related works are introduced in Section II. Problem model of WCS is presented in Section III. Network redundancy structure is investigated in Section IV. BRC detection, merging algorithms, and component based weighting algorithms are presented in Section V, VI respectively. Simulation results are presented in Section VII. The paper is concluded with remarks in Section VIII.

II. Background and Related Works

A. Problem Formulation

$G=(V,E)$ represents a graph consisting of $|V|=n$ nodes and $|E|=m$ edges. An edge $(i,j)\in E$ if the distance between v_i and v_j can be measured. The measured distance is denoted by d_{ij}, which can be obtained through signal round-trip time [19] or by signal time of arrival(TOA) [20]. The distance measurements are noisy and sparse. The *network localization problem*, which is also known as *graph realization* problem is to find d-dimensional embedding $\{p_1,p_2,\dots,p_n\}$ ($p_i\in\mathbb{R}^d$) such that $\|p_i-p_j\|_2=d_{ij}$ for all $(i,j)\in E$. In noisy case, it can be characterized as a *stress* minimization problem [15].

$$Stress(p_1,p_2,\dots,p_n)=\sum_{(i,j)\in E}(\|p_i-p_j\|_2-d_{ij})^2 \quad (1)$$

B. Related Works

Network localization problem has attracted great research attentions. Existing works can be roughly classified into two categories: 1) theoretical fundamentals; and 2) localization algorithms.

1) Theoretical Fundamentals: Some fundamental theories and conditions for network localization has been presented in the literature.

a) Localizability conditions: Whether a network can have unique localization solution is a crucial problem, which is investigated by rigidity theory by Jackson and Nixon [1], Goldenberg *et al.* [2], [3] and Aspnes *et al.* [21], and Yang *et al.* [4], [5] et al. A network can be uniquely localizable if and only if the underlying graph is global rigid [22]. In 2-D networks, global rigid requires the graph to be 3-connected [6] and redundant rigid [5]. Redundant rigid means the graph is rigid after removing any one edge. The necessary and sufficient condition for a 2-D graph being rigid is given by Laman [7] in 1979. Later in 1992 and 1997, Hendrickson and Jacobs proposed matching based [8] and Pebble game algorithms [7] respectively to test 2-D graph rigidity in polynomial time. Connelly [23] showed that the Laman condition is only necessary in $\mathbb{R}^d, d\geq 3$. Necessary and sufficient conditions for judging graph rigidity in higher dimension by the rank of stress matrix are proposed by Connelly [23], Gortler *et al.* [24], which shows a graph in $\mathbb{R}^d, d\geq 3$ is rigid if and only if $rank(\Omega)=n-1-d$, where Ω is the associated stress matrix derived from the graph. Polynomial time random algorithms are proposed for verifying graph rigidity in higher dimensions.

Yang *et al.* have shown the necessary [4] and sufficient condition [5] for a node in the network to be uniquely localizable. Sun *et al.* [25] propose metrics to evaluate the risks of flip and flex ambiguity under ranging noises. Shames *et al.* [26] study the solution uniqueness condition in cooperative localization and mapping problems. Shamsi *et al.* [27] study the conditions for correct network localization by SDP relaxation. They show the sparse triangulation graph can insure the correctness of SDP relaxation. The essence is that the triangulation graph is rigid. *But in this paper, we consider more general sparse networks, which may not satisfy the triangulation graph condition.*

b) Performance bounds: Cram'er-Rao lower bound (CRLB) [28], [29] which is also known as Fisher information inequality is mainly used to establish the lower bound of location error for network localization under noisy distance measurements. Shen *et al.* [30] derive the limits of network navigation accuracy using equivalent Fisher information analysis. Another work of them [31] derive squared position error bound (SPEB) based on Fisher information for wideband wireless network localization in harsh multipath environments. A recent work of Mazuelas *et al.* [32] investigate the location error bound using soft range information. Aspnes *et al.* [21] present complexity bound of network localization under different network parameters.

2) Localization Algorithms: Localization algorithms are summarized into three categories:

a) Optimization-based algorithms: Network localization problem is naturally an optimization problem to minimize (1). Different optimization algorithms such as Multi-dimensional Scaling (MDS) [9], SMACOF [10], Semi-definite programming (SDP) [11], and G2O [12] are proposed. MDS [9] solves this problem by matrix decomposition, which requires the network is a complete graph. SMACOF [10] is a global optimization algorithm, which constructs a quadratic function to approximate the *stress function* in each step. SDP [11] approaches localization by relaxing the problem to a convex semidefinite programming problem. The speed of centralized SDP is not satisfactory. Further results to relax the single semi-definite matrix cone into a set of small-size semidefinite matrix cones are presented in [33]. G2O [12] seeks the optimal location by gradient descent. They perform well in well-connected networks, but performances drop when networks become sparse and no anchors are used.

b) Distributed algorithms: Distributed algorithms also call great attentions. Robust Triliteration is proposed in [13], which starts from three anchors (or localized nodes) and finds robust quadrangle to locate the unvocalised vertex. Liu *et al.* [34] propose error management method in distributed localization to exclude outliers during sequential localization. Cota-Ruiz *et al.* [35] propose distributed algorithm to let each node iteratively solve a set of local spatially-constrained programs. Goldenberg *et al.* propose SWEEP [2] to localize nodes by constructing biliteration graphs. Khan *et al.* study distributed algorithms to use minimal number of anchors and by noisy measurements [36], [37]. Shi *et al.* [38] propose Sequential Greedy Optimization to solve SDP in distributed way. Soares *et al.* [39], Simonetto and Leus [40] propose convex relaxation of the network localization problem and fully distributed algorithms to solve. Piovesan and Erseghe [41] recently propose distributed algorithm which transits the convex relaxation to the original on-convex formulation to approach the local minimum of the original problem.

c) Component stitching algorithms: Component stitching algorithms calculate node local coordinates in components; synchronize components by rotating and transiting to produce the node coordinates of the network. The centralized algorithms [9]–[12] and distributed algorithms [36]–[42] can all be adopted in the component localization. Representative component stitching algorithms include CALL [17], ETOC [14], AAAP [15], ARAP [15] and ASAP [16]. ETOC [14] and CALL [17] study how to localize components by anchors, which haven't used the iteratively component synchronization. Differently, AAAP [15], ARAP [15] and ASAP [16] adopt local location calculation in each components, and coordinate synchronization by iteratively adjusting the rotation matrices of components. Component stitching generally infer network topology without using external anchors. The component stitching can effectively avoid cumulative errors. But in sparse networks, the inaccurate components will pollute the synchronization results of the network.

III. Model of Weighted Component Stitching

A. Mathematical Model

We firstly give some definitions for clearly presenting the WCS model.

Definition 1 (Edge Set Independence [7]): The edges of a graph $G = (V, E)$ are independent *in $\mathbb{R}^2$ if and only if no subgraph $G' = (V', E')$ has more than $2|V'| - 3$ edges.*

Definition 2 (Laman Graph [7]): A graph is Laman Graph *if the edges of E are independent and $|E| = 2|V| - 3$.*

Definition 3 (Rigid Graph): A graph is rigid if and only if it contains a spanning Laman Graph as a subgraph.

Definition 4 (Redundant Rigid Graph): A graph $G = (V, E)$ with realization in $\mathbb{R}^2$ having $n \geq 2$ nodes is redundantly rigid *if and only if it remains rigid after the removal of any one edge $(i, j) \in E$.*

Existing works [15], [16] divide components by the one-hop neighbors of each node. In WCS, we propose to find and rank the redundant rigid components in the graph. To present the WCS model, we firstly assume the redundant rigid components can be successfully detected and are ranked by a proposed redundant ratio (RR) metric, which are denoted by $G^c = \{G^c_1, G^c_2, G^c_3, \ldots, G^c_L\}$. The superscript c indicates G^c_k is a component and the subscript k is the component index. The local coordinates of nodes in G^c_k are calculated by algorithms such as SDP [11], MDS [9], or SMACOF [10]. Node i's local coordinates in G^c_k is denoted by $q^{c,k}_i$. We use r_k to denote the RR metric of the component G^c_k. Let $w_k = f(r_k)$ be the synchronization weight calculated based on the RR metric. Then the cost function for weighted component synchronization in WCS is defined as:

$$F = \min_{P, R^c_1, \ldots, R^c_L} \{ \sum_{k=1}^{L} \sum_{(i,j) \in G^c_k} f(r_k) \| (p_i - p_j) - R^c_k (q^{c,k}_i - q^{c,k}_j) \|^2 (R^c_k)^{\mathrm{T}} R^c_k = I \} \quad (2)$$

where $P = \{p_1, p_2, \ldots, p_n\}$ are the global coordinates to calculate; R^c_k is the rotation matrix of the component G^c_k. The problem is to determine P and R^c_1 to R^c_L such that the global coordinates are best synchronized with the local coordinates in the components.

B. Redundant Ratio: (RR)

Redundant ratio is designed to evaluate the edge redundancy in a component. A redundant rigid graph $G(V, E)$ must be rigid, it contains a rigid spanning graph (Laman graph) [42], denoted by $\mathcal{L}$. Edges in $\mathcal{L}$ are necessary edges to guarantee rigid; denoted by $\widetilde{\mathcal{E}}$. Edges in $E - \widetilde{\mathcal{E}}$ are redundant edges, which are denoted by $\mathcal{F} = E - \widetilde{\mathcal{E}}$. In $\mathbb{R}^2$, in a redundant rigid graph with n nodes, its Laman spanning graph contains exactly $2n-3$ edges [43]. Therefore, the number of redundant edges is $|\mathcal{F}| = |E| - (2n - 3)$.

Definition 5 (Redundant Ratio (RR)): Considering a redundant rigid graph $G = (V, E)$ with $|V| = n$, if it has a redundant edge set $\mathcal{F}$, the redundant rate (RR) *is calculated as $r(G) = \frac{|\mathcal{F}|}{C(n,2)}$, where $C(n, 2) = \frac{n(n-1)}{2}$. So $r(G) = \frac{2|\mathcal{F}|}{n(n-1)}$*

The proposed RR metric $r(G)$ is a variation of the traditional edge density metric $\gamma(G)$, which is the ratio of edges in G to the number of edges in a complete graph, i.e., $\frac{2|E|}{n(n-1)}$ [44]. So $r(G) = \gamma(G) - \frac{2n-3}{C(n,2)}$. The reason is that $\gamma(G)$ cannot discriminate the complete graphs with different number of vertices. Let K_4 and K_5 be complete graphs with four and five vertices respectively. We have $\gamma(K_4) = \gamma(K_5)$ and $r(K_5) > r(K_4)$. This is a desired property because K_5 has more constraints which is more reliably realized than K_4.

C. NP Hardness

The key problem is how to find the redundantly rigid components and rank them by RR.

Definition 6 (Locally Most Reliable Component (LMRC)): A LMRC is a redundant rigid component that 1) any subgraph of it will have smaller RR than itself, and 2) any graph covering it will have smaller RR than the LMRC.

Theorem 1: Finding the LMRC in a given graph $G = (V, E)$ is NP-hard.

Proof: The traditional determination problem of maximum quasi-clique is to find the maximum γ-dense component, i.e, to find the connected component G' with the maximum number of vertex, s.t., $\frac{|E|}{C(n,2)} \geq \gamma$. Note that $r(G') = \gamma(G') - \frac{2n'-3}{C(n',2)}$. Then we prove by contradiction. Given γ, if the problem of finding the component with highest RR can be solved in polynomial time, i.e., $\gamma - \frac{2n-3}{C(n,2)}$ is the maximum in the detected connected subgraph G^*, then $G*$ must be the component with the maximal number of vertices, since the larger is n, the smaller is $\frac{2n-3}{C(n,2)}$ when $n > 3$. So we find the component with the maximum number of vertices of density γ in polynomial time. This is contradict because finding the maximum γ-dense component is known NP-complete [45]. □

Since finding the most redundant LMRCs is NP-hard, it is not trivial to find and rank the redundant components. We investigate the structural properties of the redundantly rigid graph to propose efficient methods to find the components with different RRs.

IV. Redundant Rigid Graph Characteristics

At first, how the redundantly rigid components may distribute in a graph G is investigated.

Lemma 1: Given two redundantly rigid graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, $G_1 \bigcup G_2$ is redundantly rigid if and only if $|V_1 \bigcap V_2| \geq 2$.

Proof: By definition, both of the graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ contain a spanning Laman Graph as a subgraph, denoted as L_1 and L_2. Supposing $|V_1 \cap V_2| = k$, then $|V(L_1 \cap L_2)| = k$. It has been proven that $L_1 \cup L_2$ is rigid when $k \geq 2$ [46]. So $L_1 \cup L_2$ is rigid when $|V_1 \bigcap V_2| \geq 2$.

For each edge e in graph $G_1 \bigcup G_2$, $G_1 \bigcup G_2 - e = (G_1 - e) \bigcup (G_2 - e)$. We can find the Laman Graph $L'_1 \subseteq G_1 - e$ and $L'_2 \subseteq G_2 - e$ when G_1 and G_2 are redundantly rigid. What's more, $|V(L'_1 \bigcup L'_2)| \geq 2$. So $L'_1 \bigcup L'_2$ is rigid, and $G_1 \bigcup G_2$ is redundantly rigid. □

By Lemma 1, if two redundantly rigid subgraphs share not less than two vertices, their union is redundantly rigid. So a redundantly rigid component is *disjoint* only if it has less than two vertices with others. Two concepts of Extremely Redundant Rigid Component and Basic Redundant Rigid Component are defined:

Definition 7 Extremely Redundant Rigid Component, ERC: A component G_0 in graph G is called an Extremely Redundant Rigid Component if there is no other redundantly rigid components, whose subgraph is G_0.

Definition 8 Basic Redundant Rigid Component, BRC: A redundant rigid component G' in graph G is a Basic Redundant Rigid Component if any subgraph $G'' \subset G'$ is not redundantly rigid.

An ERC is a disjoint redundant rigid subgraph. It remains rigid if removing any one edge from it. A ERC may contains many number of redundant edges. A BRC, instead, is a small redundant rigid component in ERC which contains only one redundant edge.

Lemma 2: Given a rigid graph $G = (V, E)$ with redundant edges, there is a determinate set of disjoint ERCs: $C = \{ERC_1, ERC_2, ERC_3 \ldots ERC_k\}, k \geq 0$, which covers all redundant edges in G, and $\forall ERC_i, ERC_j \in C, E(ERC_i) \cap E(ERC_i) = \varnothing$.

Proof: Consider the rigid graph contains k redundant edges. If the ERC set isn't unique, there must be two sets $C_1 = \{ERC'_1, ERC'_2, ERC'_3 \ldots ERC'_{k1}\}$ and $C_2 = \{ERC''_1, ERC''_2, ERC''_3 \ldots ERC''_{k2}\}$, which are not exactly the same. So $\exists ERC''_i \in C_2, \forall ERC'_j \in C_1, ERC''_i \neq ERC'_j$. If $|V(ERC''_i \bigcap ERC'_j)| \geq 2$, the graph $ERC''_i \bigcup ERC'_j$ is also redundant rigid, which is inconsistent with the definition of ERC is the largest redundant component; if $\forall j$, $|V(ERC''_i \bigcap ERC'_j)| < 2$, the redundant edge in ERC''_i is not covered by any ERCs in C_1, this is contradict with that $ERCs$ in C_1 cover all the redundant edges. So there is only one set of disjoint ERCs covering all the redundant edges. □

From Lemma 2, when a set of ERCs are found, the set of ERCs are disjoint and determinate. The ERCs can be found by Pebble game [7] in polynomial time. But in ERCs, we should further detect the denser subgraphs, because the denser subgraph has more reliable local realization than the ERC. Therefore, we consider LMRC detection in each ERC. We firstly show that each ERC is covered by a set of BRCs.

Theorem 2: For any $ERC_i \in G$, there is a combination $BRC_1, BRC_2, BRC_3, \ldots, BRC_K \in G$, such that $BRC_1 \bigcup BRC_2 \bigcup BRC_3 \bigcup \ldots \bigcup BRC_K = ERC_i$.

Proof: For any $ERC_i \in G$, we can find a Laman Graph $L_i \subset ERC_i$ and the accompanying redundant edges. Then by finding the Hungarian tree of each redundant edge [8], a combination of BRCs can be found, denoted as $BRC_1, BRC_2, BRC_3, \ldots, BRC_K$. $C' = BRC_1 \bigcup BRC_2 \bigcup BRC_3 \bigcup \ldots \bigcup BRC_K$ and $C' \subseteq ERC_i$. If $E(ERC_i - C') \neq \emptyset$, for each $e \in ERC_i - C'$, $ERC_i - e$ is rigid and contains a Laman Graph L_e. Then we can find a BRC in $L_e + e$, denoted as BRC_e. Add BRC_e into the combination C' and the combination can cover all edges in ERC_i, so that there must be $C' = ERC_i$. □

So the key problem is how to find the BRCs, whose distribution implies the distribution of the redundant edges. The BRCs be merged efficiently to find the dentist components in the ERC.

V. Finding and Merging the BRCs

We present an efficient algorithm based on Bipartite Matching to find the BRCs. The input is a graph G. The ERCs in G are detected by Pebble game [7]. The remaining problem is how to detect BRCs in each ERC. So we consider $G = (V, E)$ is an ERC in Algorithm 1.

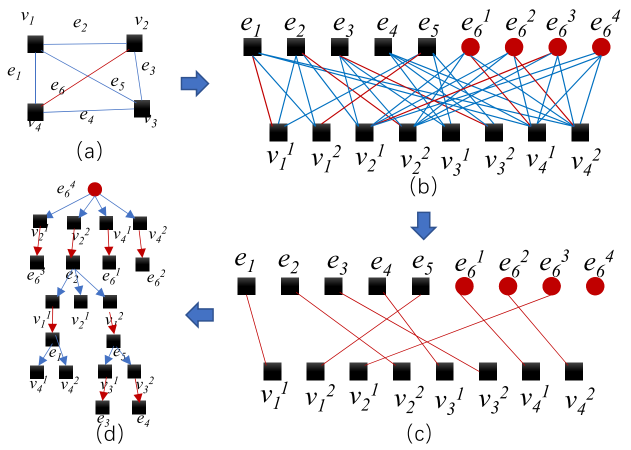


Fig. 1. Example of matching-based redundant rigidity test. (a) An example K_4 graph; (b) The bipartite graph model of K_4. Three additional copies of e_6 are added for matching the vertex freedoms; (c) The matching result, in which, e_6^4 cannot find a match, so e_6 is a redundant edge. The matching results are also highlighted in (b); (d) the Hungarian tree rooted at e_6^4. All edges e_1 to e_5 appear on the tree, so they are dependent with the redundant edge e_6.

A. Bipartite Matching to Detect Independent Edges

Bipartite Matching-based rigidity test method [8] was traditionally designed to test whether a given graph is rigid or not. We improve the method to output the BRC set when there are multiple redundant edges in an ERC.

The bipartite matching-based method [8] converts a graph $G = (V, E)$ into a bipartite graph $B(G) = \{V_1, V_2, \mathcal{E}\}$, where $V_1 = \{e \in E\}$ is the original edge set; $V_2 = \{v_1^1, v_1^2, v_2^1, v_2^2, \cdots, v_n^1, v_n^2\}$ are two copies of the original nodes. $\mathcal{E} = \{(e, v_i^1), (e, v_j^1), (e, v_i^2), (e, v_j^2) : e = (v_i, v_j) \in E\}$ connects the edges of G with the two copies of their incident vertices. Let's denote $G_{i,j}$ the graph formed by adding an edge $(i, j) \in E$ three times into G (to match the three always-existing node freedom, i.e., rigid rotation and transition in 2D space). For example, in Fig.1(a), if three copies of edge $e_6 = (2, 4)$ is added into G, it forms a graph $G_{2,4}$. The corresponding bipartite graph of $G_{2,4}$ is shown in Fig.1(b), which is denoted $B(G_{2,4})$. The edge $e_{i,j}$ is independent with other edges if the bipartite graph $B(G_{i,j})$ has a perfect matching from V_1 to V_2 [8, Th. 2.8]]. Fig.1 (a)(b) gives an example of transforming complete graph K_4 to a bipartite graph during testing the edge independence of e_6.

The rigidity test starts from an independent set $\mathcal{L}$ which includes only one edge, and checks independence with other edges. If an edge $e \in G$ is independent with $\mathcal{L}$, $\mathcal{L}$ is expanded to $\mathcal{L} \cup e$, otherwise e is detected to be redundant. When e is detected redundant, let $\mathcal{V}_1$ be the vertex set in V_1 which is in the Hungarian tree rooted at e and not matched. Then any edge in $\mathcal{V}_1$ can be replaced by e without violating the set independence of $\mathcal{L}$. Fig.1 (c) shows e_6^4 is unmatched, so e_6 is redundant. Fig.1 (d) shows the Hungarian tree rooted at e_6. All edges appear in this tree is exchangeable with e_6 without violating the set independence of $\mathcal{L}$.

The Hungarian tree provides the influence scope of a redundant edge. It works when there is only one redundant edge, because the detected redundant edges need to be discarded for keeping the edge set independent [8]. We propose a method to give the influence scope for all the redundant edges.

B. Algorithm to Detect the BRCs

A BRC detection algorithm in Algorithm 1 is designed to find BRCs based on the Bipartite Matching method. Two edge sets are designed: $E_{redundant}$ and $E_{independent}$, which are updated by Bipartite Matching. Finally, for each edge e in $E_{redundant}$, we can find a BRC_e by Hungarian tree, getting a set of BRCs, denoted as $\mathcal{B} = \{BRC_1, BRC_2, \ldots, BRC_{|\mathcal{F}|}\}$, when there are $|\mathcal{F}|$ redundant edges.

Algorithm 1 BRCs Detect Algorithm

```
1: Input: A rigid graph $G = (V, E)$
2: Output:$\mathcal{B} = \{BRC_1, BRC_2, \ldots, BRC_K\}$
3: $\mathcal{B} \leftarrow \emptyset$, $E_{redundant} \leftarrow \emptyset$,$E_{independent} \leftarrow \emptyset$;
4: for each edge $e \in E$ do
5:     if $e$ are independent with edges in $E_{independent}$ then
6:         $E_{independent} \leftarrow E_{independent} + \{e\}$;
7:     else
8:         $E_{redundant} \leftarrow E_{redundant} + \{e\}$;
9:     end if
10: end for
11: for each edge $e \in E_{redundant}$ do
12:     Find $BRC_e$ by Hungarian tree;
13:     $\mathcal{B} \leftarrow \mathcal{B} + \{BRC_e\}$;
14: end for
15: return $\mathcal{B}$
```

Each BRC contains one redundant edge. We prove the union of $|\mathcal{F}|$ detected BRCs covers the $|\mathcal{F}|$ redundant edges of a ERC.

Lemma 3: Given a rigid graph G, a combination of BRCs can be found by Algorithm 1, denoted as $\mathcal{B} = \{BRC_1, BRC_2, BRC_3, \ldots, BRC_{|\mathcal{F}|}\}$. Then, $\forall ERC_i \subseteq G$, $ERC_i \in \mathcal{B}$.

Proof: Given a rigid graph G, a spanning Laman Graph L can be determined and a combination of BRCs can be found through the above method, denoted as $\mathcal{B}' = \{BRC_1', BRC_2', BRC_3', \ldots, BRC_k'\}$.

If $\exists ERC' \notin \mathcal{B}'$, then $E(\mathcal{B}' - ERC') \neq \emptyset$. Let $e' \in E(\mathcal{B}' - ERC')$, there must be $e' \in L$. Otherwise, there must be a BRC generated by e'. What's more, $\forall BRC_i'$, $|E(BRC_i' \bigcap L)| = 2|V(BRC_i')| - 3$. For any spanning Laman Graph L_i, $|E(L_i \bigcap \mathcal{B}')| \leq |E(L \bigcap \mathcal{B}')|$. So, if removing edge e', the module $L - \mathcal{B}'$ can't find more edge to form a new Laman graph. And $\forall ERC_k$, the edge $e' \notin ERC_k$, i.e., $\forall ERC \subseteq G$, $ERC \in \mathcal{B}$. □

A subset of k BRCs will contain k redundant edges and each BRC contains one distinct redundant edge. So that, to find a highly redundant component, we should find a large set of BRCs (i.e. k BRCs) whose union covers a small number of vertex (i.e., n vertices), so that $\frac{k}{C(n,2)}$ can be maximized. This leads to the BRC merging problem, which is to merge BRCs to find more redundant components.

C. BRC Merging

Lemma 4: Let G_i be a BRC and G' be an arbitrary graph. $R_{ratio}(G_i \cup G') > R_{ratio}(G_i)$ only if G' is redundant rigid.

Proof: Suppose $R_{ratio}(G_i)$ is the RR of G_i. $R_{ratio}(G_i \cup G') = \frac{|\mathcal{F}(G)_i|+|\mathcal{F}(G')|}{C(n(G_i)+n(G'))}$ which is larger than $R_{ratio}(G_i) = \frac{|\mathcal{F}(G)_i|}{C(n(G_i)+n(G'))}$ only if $|\mathcal{F}(G')| > 0$. So G' must be redundantly rigid. □

Therefore, we only need to consider merging BRC with other BRCs without the need to consider merging BRC with non-redundant subgraphs. After merging cannot be further proceeded, we will obtain a set of subgraphs which have better redundant ratio than all its merged subgraphs and they cannot be further merged with others, which are LMRCs.

Mathematically, let's denote the set of detected BRCs by $\mathcal{B} = \{BRC_1, BRC_2, BRC_3, \ldots, BRC_K\}$, i.e., there are K BRCs. Let $\mathcal{C} \subseteq \mathcal{B}$, be a BRC subset, i.e., $\mathcal{C} = \{BRC_1^c, BRC_2^c, BRC_3^c, \ldots, BRC_t^c\}$ with $t \leq K$. If the BRCs in $\mathcal{C}$ can be merged, we will get a merged graph $G^C = BRC_1^c \cup BRC_2^c \cup BRC_3^c \cup \ldots \cup BRC_t^c$. To reach a LMRC, it is equivalent to find among all subsets $\mathcal{C} \subseteq \mathcal{B}$, such that $r(G^{\mathcal{C}}) = \frac{t}{C(|V(G^{\mathcal{C}})|,2)}$ is maximized.

$$\mathcal{C} = \arg\max_{\mathcal{C} \subseteq \mathcal{B}} r(G^{\mathcal{C}}) \tag{3}$$

D. Conditions for Components Merging

Let G_i^c and G_j^c be two redundant rigid components. These two components can be merged only if the merged component has larger redundant ratio than any of them. So the condition to merge two components is:

$$R_{ratio}(G_i^c \cup G_j^c) > \max\left\{R_{ratio}(G_i^c), R_{ratio}(G_j^c)\right\} \tag{4}$$

Supposing the number of redundant edges in G_i^c, G_j^c and $G_i^c \cup G_j^c$ are e_1, e_2 and e_3 respectively. The number of redundant edges after merging equals to the number of redundant edges before merging, i.e., $e_3 = e_1 + e_2$. By the definition of the redundant ratio, the Equation (4) is equivalent to:

$$\begin{aligned} \frac{2e_3}{|V_i^c \cup V_j^c|(|V_i^c \cup V_j^c|-1)} &> \frac{2e_1}{|V_i^c|(|V_i^c|-1)} \\ \frac{2e_3}{|V_i^c \cup V_j^c|(|V_i^c \cup V_j^c|-1)} &> \frac{2e_2}{|V_j^c|(|V_j^c|-1)} \end{aligned} \tag{5}$$

By adding both sides of Equation (5), we get:

$$|V_i^c|\left(|V_i^c|-1\right)+|V_j^c|(|V_j^c|-1) > |V_i^c \cup V_j^c|(|V_i^c \cup V_j^c|-1) \tag{6}$$

Whether two redundant components can be merged can be checked by (6) according to the number of vertices before and after the component merging. For example, if two redundantly rigid components contain both 10 vertices, (6) is true only if their merged component has less than 13 vertices, i.e., the two components have more than 7 overlapped vertices.

E. The Dynamic Merging Graph

Utilizing the merging condition, the component merging problem is converted to node merging problem on a dynamic graph. For a graph containing K components, each component can be treated as a vertex, with its redundant ratio as the vertex attribute. Two components are connected by an edge if the Equation (5) for merging these two components can be satisfied. In the initial state, there are K vertices and edges indicating whether two components can be merged.

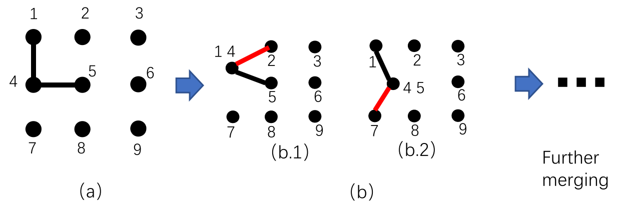


Fig. 2. An example to illustrate the BRC merging problem.

Two components selected to be merged will be aggregated into one node and the edges to other components will be updated. The number of vertices will be reduced by one and the edges will be updated based on whether the merged node can be further merged with the other nodes. It should be noted that even if the original two nodes have no edge to another node, after they are merged, the merged node may have an edge to the node. So the merging graph may change after a merging decision, which makes the merging problem challenging.

Fig.2 shows an example. The initial graph contains nine components. (1,4) and (4,5) satisfy condition (5), which are connected by edges. Fig.2(b) shows two options of merging two nodes. Fig.2(b.1) and Fig.2(b.2) are the updated graphs if (1,4) or (4,5) is merged respectively. Note that, in Fig.2(b.1), node 2 and the merged node of (1,4) satisfy the merge condition, but node 2 doesn't satisfy the condition with either 1 or 4.

F. Greedy Merging and Updating Algorithm

An algorithm is designed by greedly one-one merging, which is listed in Algorithm 2. The input of the algorithm is the set of BRCs. A component graph $\mathcal{G}(V^c, E^c)$ is firstly initialized by treating each BRCs as a vertex. Component i is denoted by G_i^c. An edge $(i, j) \in E^c$ if condition (5) is satisfied for BRC i and j. The neighbors of each BRC i, i.e., $N(i)$ are BRCs sharing at least one node with BRC i.

The algorithm outputs a set of LMRCs. In each step, the two components whose union has the highest RR will be selected to be merged. After the merge operation, the edges from the merged component's to the neighboring components are checked and updated. The merging process repeats until no two components can be further merged.

Because checking the merging condition is based only on the number of vertices in a component, the component graph initialization has complexity $O(K^2)$, where K is the number of BRCs. Finding the merging option with the largest RR has complexity $O(K^2)$, so the overall complexity of the Algorithm 2 is $O(K^3)$.

We analyze the approximation ratio of RR found by algorithm 2 to the optimal RR of the most dense subgraph in the ERC. Let n_{min} denote the minimum nodes number in any BRC, i.e., the minimum BRC size. It is generally 4 or 5. Let m_{max} denote the maximum number of redundant edges

Algorithm 2 Greedily Merge and Update Algorithm

```
1: Input: $\mathcal{B} = \{BRC_1, BRC_2, \ldots, BRC_K\}$
2: Output:$G^c = \{G_1^c, G_2^c, \ldots, G_L^c\}$
3: Initialize dynamic graph $\mathcal{G}(V^c, E^c)$ by BRCs set $\mathcal{B}$;
4: Initialize $G^c = V^c$
5: while $E^c \neq \emptyset$ do
6:     $\{i, j\} := \max_{(i,j)\in E_d} r(G_i^c \cup G_j^c)$;
7:     $T = r(G_i^c \cup G_j^c)$
8:     if $T > r(G_i^c)$ and $T > r(G_j^c)$ then
9:         $G_i^c \leftarrow G_i^c \cup G_j^c$;
10:        $G^c \leftarrow G^c - \{G_j^c\}$;
11:        Update neighboring edges of $G_i^c$ in $\mathcal{G}(V^c, E^c)$;
12:    end if
13: end while
14: return $G^c$
```

in the considered ERC. Then, given an arbitrary ERC, there must be a subgraph which has the largest **Redundant Ratio**, whose RR is denoted as OPT. Algorithm 2 will greedily merge components until a LMRC cannot be further merged with others. The RR of the LMRC is bounded in theorem 3.

Theorem 3: The redundant ratio provided by algorithm 2 is at least $max\{\frac{1}{n_{min}(n_{min}-1)}, \frac{1}{m_{max}}\}$ *times of the* OPT.

Proof: Algorithm 2 merges the components for larger **Redundant Ratio**. In the worst case, any two BRCs cannot be merged, the result RR of the LMRC is $\frac{1}{n_{min}(n_{min}-1)}$. Since OPT must be less than 1, so the approximation factor is at least $\frac{1}{n_{min}(n_{min}-1)}$. On the other hand, the most number of redundant edges that the OPT has is m_{max}. In the worst case, the LMRC has one redundant edge, so the approximation factor is $max\{\frac{1}{n_{min}(n_{min}-1)}, \frac{1}{m_{max}}\}$. □

VI. Weighted Component Stitching

After finding LMRCs, local coordinates of nodes are calculated in LMRCs by SMACOF algorithm and the local coordinates are synchronized to calculate the global coordinates of nodes. In practice, many parts in the graph are not redundantly rigid. For component stitching, we make an integration of ARAP [15] and the WCS objective in (2). The nodes in sparse parts form patches as in ARAP [15] and are synchronized with the LMRCs by (7).

$$\begin{aligned}F = \min_{P,R_1,\ldots,R_1^c,\ldots} \{&\sum_{i=1}^{n}\sum_{j\in N_i} \|(p_i - p_j) - R_i(q_i^i - q_j^i)\|^2 \\ &+\sum_{l=1}^{L}\sum_{(i,j)\in G_l^c} f(r_l)\|(p_i - p_j) - R_l^c(q_i^{c,l} - q_j^{c,l})\|^2 \\ &: R_i^{\mathrm{T}} R_i = I, R_l^{c\mathrm{T}} R_l^c = I\} \end{aligned} \quad (7)$$

In (7), $P = \{p_1, p_2, p_3 \ldots, p_n\}$ represents the desired global coordinates of nodes. q_i^i and q_j^i are local coordinates in the patch G_i, i.e., the subgraph induced by the one-hop neighbor of i. The local coordinates in such kinds of patches are equally weighted. The second part is weighted component synchronization. If L LMRCs are detected, the weights for local coordinates in each LMRC are assigned according to RRs of the LMRCs. By setting $f(r_l) > 1$, we guarantee the redundant components have larger impacts than the ordinary patches in synchronization.

A. The Synchronization Algorithm

The weighted component synchronization in (7) can be solved using an Alternating Least-Squares (ALS) method. It is composed by the iteration of two phases: 1) a local phase to solve R and R_c; and 2) a global phase to solve P.

In the local phase, we assume P is fixed to solve R_i and R_k^c in (7). The initial value of P can be obtained by solving the linear equations as in ARAP [15]. Given P, we need to solve rotation matrix R_i and R_k^c through the following optimization problems, which is a typical point cloud matching problem.

$$\begin{aligned} R_i &= \arg\min_R\{\|P_i - R_i Q_i\|_F^2 : R^T R = I\} \\ R_k^c &= \arg\min_R\{\|P_k - R_k^c Q_k^c\|_F^2 : R_k^{cT} R_k^c = I\} \end{aligned} \quad (8)$$

The rotation matrices in (8) is efficiently calculated using Iterative Closest Point (ICP) algorithm [47]. The ICP algorithm uses Sigular Value Decomposition to calculate the rotation matrix for aligning two point clouds. Its complexity is $O(n_c^2)$ for a component having n_c nodes, which is highly efficient in this problem.

In the global phase, we assume all R_i and R_k^c are fixed to solve P by Eqn (7). This is obtained by setting the gradient $\frac{\partial F}{\partial p_i} = 0, \forall i = 1, 2, \cdots, n$, which helps to set up n linear equations of p_i and p_j, where $p_j \in N_i$.

$$\begin{aligned}\sum_{j\in N_i}(p_i - p_j) = &\frac{1}{2 + \sum_{G_l^c:(i,j)\in G_l^c} f(r_l)} \\ &\cdot(\sum_{j\in N_i}[R_i(q_i^i - q_j^i) + R_j(q_i^j - q_j^j) \\ &+ \sum_{G_l^c:(i,j)\in G_l^c} f(r_l)R_l^c(q_i^{c,l} - q_j^{c,l})]) \end{aligned} \quad (9)$$

Note that q_j^i is the jth node's position in the local coordinate system of G_i. $\sum_{G_l^c:(i,j)\in G_l^c} f(r_l)$ is the summation in all components that contains edge (i, j). By solving these n linear equations, the global coordinates are obtained, which are then substituted into (7) for solving R_i and R_l^c. The iteration repeats until convergence of P or reaching a maximum number of iteration times.

B. Complexity Analysis of WCS

Theorem 4: Given a sparse graph with n *vertices and* m *edges, with the node degree bounded by* Δ*, the presented weighted component stitching algorithm for localizing nodes in the network has complexity of* $O((\Delta n)^3 + h \cdot n^3)$*, where* h *is the maximum iteration time in the synchronization step.*

Proof: Given a sparse graph $G = (V, E)$, $|V| = n$ and $|E| = m$, where $m \leq \Delta n$. Firstly, we find a combination of BRCs, denotes as $BRC_1, BRC_2, BRC_3, \ldots, BRC_K$.

TABLE I
THE TIME COMPLEXITY FOR DIFFERENT ALGORITHMS

Algorithms	WCS	ARAP	SDP	SMACOF
Complexity	$O((\Delta n)^3 + hn^3)$	$O(h \cdot n^3)$	$O(m^{3.5})$	$O(h \cdot n^2)$

We finish this step by Bipartite Matching Algorithm, whose time complexity is $O((n+m)^3)$=$O(((\Delta+1)n)^3)$. Then, we merge these K BRCs into L LMRCs by Algorithm 2, $L \leqslant K < n$. And the time complexity of this step is $O(K^3)$. Finally, we correct the global coordinates of all nodes by LMRCs and every node's local coordinate. To solve Eqn (9), we need to obtain all modules, including $(q_i^i - q_i^i)$ and $(q_i^c - q_j^c)$. We need a traversal of all edges in all local coordinates. The time complexity of this process is $O((n+L)m)$, which is equivalent to $O(\Delta n^2)$. The time complexity of the method we use to solve the linear system of equations is $O(n^3)$. Because the maximum iteration times is h (which is set 100 in simulations), the time complexity of this step is $O(hn^3)$. Therefore, the mostly time consuming step is the BRC generation and weighted component stitching, so that the total time complexity is $O((\Delta n)^3 + hn^3)$. □

We compared the time complexity of WCS, ARAP, SDP and SMACOF. The result is shown in table 1.

C. Selection of the Weighting Function $f(r_l)$

It is obvious that the weighting function $w_l = f(r_l)$ will impact the synchronization result. There are different ways to design the weighting function. We investigated four types of weighting functions, namely, *linear function, logarithmic function, cubic function and exponential function*. These four forms of weighting functions are designed as below:

$$\begin{aligned} f_{linear}(r) &= 1 + \alpha * r \\ f_{logarithmic}(r) &= 1 + ln(1 + \alpha * r) \\ f_{cubic}(r) &= (1 + \alpha * r)^3 \\ f_{exponential}(r) &= 1 - e + e^{1+\alpha * r} \end{aligned} \tag{10}$$

The parameters α in the Equation (10) is a scaling factor, which can be changed to control the scale of the weights. We compare how the different weighting schemes will impact the localization accuracy. To construct representative weighting functions, we adjust the parameters α, so that the designed four types of weighting functions are given as shown in the Fig. 5. The impacts of the weighting functions are investigated by simulations. It shows that WCS by all these weighting schemes outperform traditional CS method. Among them the linear weighting scheme performs the best.

VII. PERFORMANCE ANALYSIS AND SIMULATION

A. Validity of the RR Metric

We firstly verify the validity of the proposed RR metric on the prediction quality of local coordinates. Three other potential metrics are compared with the RR metric.

1) Average node degree in the component, i.e., $r(G_i^c) = \frac{\sum_{V_i \in G_i^c} d(V_i)}{|V(G_i^c)|}$;

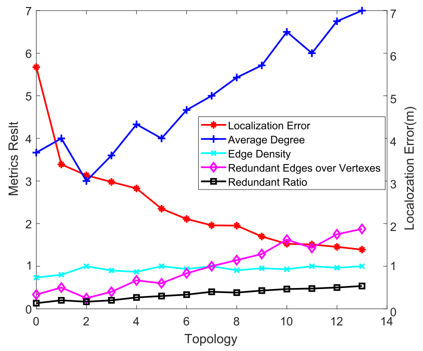


Fig. 3. The Localization Error and Metrics' Performance in Different Topologies.

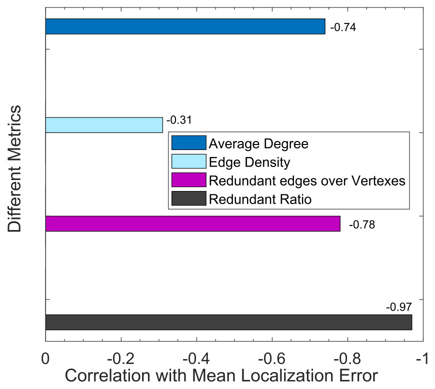


Fig. 4. The correlation coefficient between localization errors and these four metrics.

2) Edge density in the component, i.e., $r(G_i^c) = \frac{2|E(G_i^c)|}{|V(G_i^c)|(|V(G_i^c)|-1)}$

3) Number of redundant edges over number of, i.e., $r(G_i^c) = \frac{|E(G_i^c)|-2(|V(G_i^c)|-3)}{|V(G_i^c)|}$.

Different network topologies are generated to evaluate both the location error and the values of different metrics. The network parameters are controlled so that the location errors keep reducing. The experiments are carried out by Matlab2017b. One hundreds nodes are deployed randomly in a 100*100 area. The maximum ranging radius is controlled to generate graphs of different sparsity. In each parameter setting, 100 graphs are randomly generated to evaluate the mean location error and the mean values of the metrics. The results are shown in Fig. 3.

The x-axis in Fig. 3 indicates different network topologies. The average localization errors under these settings are shown by the red line which are monotonously decreasing in these settings. Among the four metrics, only RR provides a monotone increasing curve. The correlation coefficients of the localization errors with the four metrics are also calculated. The result is shown in Fig. 4. It shows that RR and the localization error have the strongest negative correlation, i.e., −0.97, which provides a better indication from edge redundancy to the local realization quality than the other metrics.

B. Impacts of the Weighting Function

Experiments were carried out to compare the impacts of weighting functions to the localization results. We also investigates how weighting in patches using different metrics

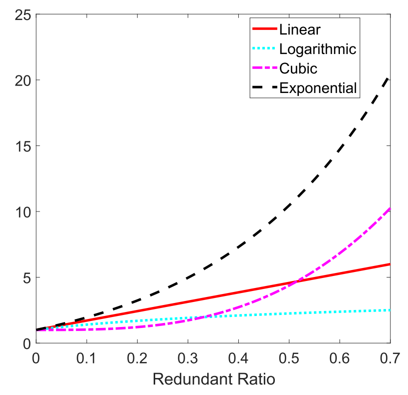


Fig. 5. Four forms of weighting functions.

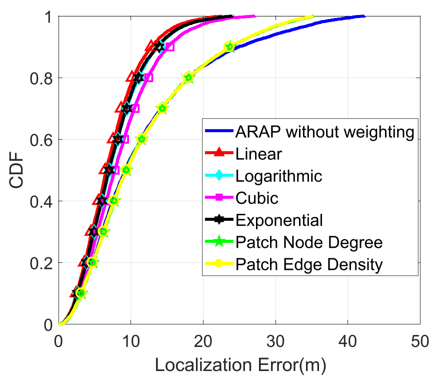


Fig. 6. The CDF of localization results.

will impact the localization results. Therefore, we compare the localization results of the following six weighting schemes with that of ARAP.

1) linear component weighting $f_{linear}(r)$;
2) logarithmic component weighting $f_{logarithmic}(r)$;
3) cubic component weighting $f_{cubic(r)}$;
4) exponential component weighting $f_{exponential}(r)$;
5) patch weighting by average patch node degree $f(G_i) = \frac{\sum_{V_i \in G_i} d(V_i)}{|V(G_i)|}$;
6) patch weighting by patch edge density $f(G_i) = \frac{|E(G_i)|}{C(|V(G_i)|,2)}$

Note that the cost function in 5) 6) for patch weighting is $F = \min_{P,R_1,\ldots,R_n} \{\sum_{i=1}^{n} \sum_{j \in N_i} f(G_i) \|(p_i - p_j) - R_i(q_i^i - q_j^i)\|^2 : R_i^{\mathrm{T}} R_i = I\}$

We've documented the localization error of each node as $\|p_i - p_i'\|$, where p_i' is the estimated localization of v_i; p_i is the ground-truth location. Then we present the experimental results in the form of cumulative distribution, as shown in the Fig. 6. In simulation, 100 nodes are randomly deployed into an area of $100(m) \times 100(m)$. We control the maximum ranging distance and each experiment are run in 100 random topologies for calculating the cumulative distribution.

In the result of Fig.6, we compare the cumulative probability density function (CDF) of location errors of ARAP (which is the state-of-the-art component-stitching algorithm) with the

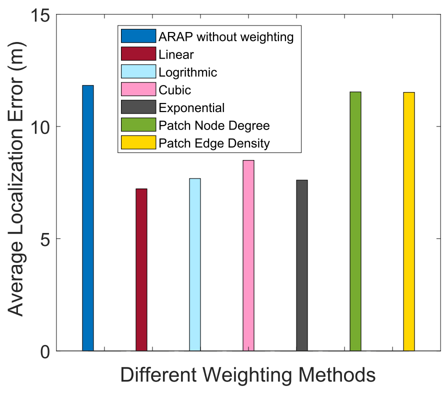


Fig. 7. Average localization error comparison.

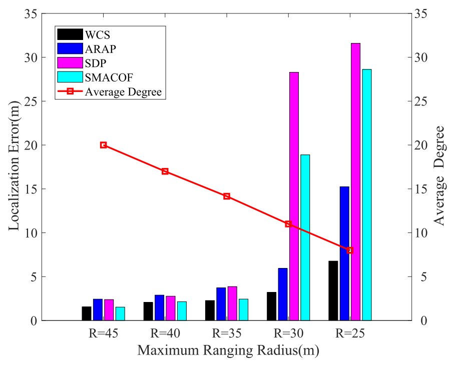


Fig. 8. Selected algorithms' performances with different edge density.

CDFs of six kinds of weighting schemes. The results in Fig.5 show that patch-based weighting by either 5) or 6) improves the localization accuracy only a little comparing with the result of ARAP. But all the four component-based weighting schemes perform much better than the patch-based weighting and ARAP. Among the four component-based weighting schemes, linear weighting performs the best. This is due to the linear correlation feature of RR and the location error as shown in Fig.3.

The average localization errors of these seven methods are further plotted in Fig.7. It shows that the linear component weighting scheme reduces the average localization error 38% than that of ARAP, while the patch-based methods reduce about 2.6%, showing the effectiveness of WCS.

C. Performance Comparison in Various Networks

1) Simulation Settings: WCS is compared with the state-of-the-art of component stitching method ARAP [15], and two centralized network locating method SDP [11] and SMACOF [10]. The centralized algorithms treat the whole graph as one component to conduct location optimization.

The considered network settings include: 1) edge density; 2) ranging noises. In simulation, we generate networks by random geometric graphs. 50 to 100 nodes were randomly deployed into an area of $100(m) \times 100(m)$. The edge density of the graph is controlled by the ranging radius R. $d_{i,j}$ between nodes v_i and v_j can be measured if $\|p_i - p_j\| < R$. The distance measurements are impacted by noises

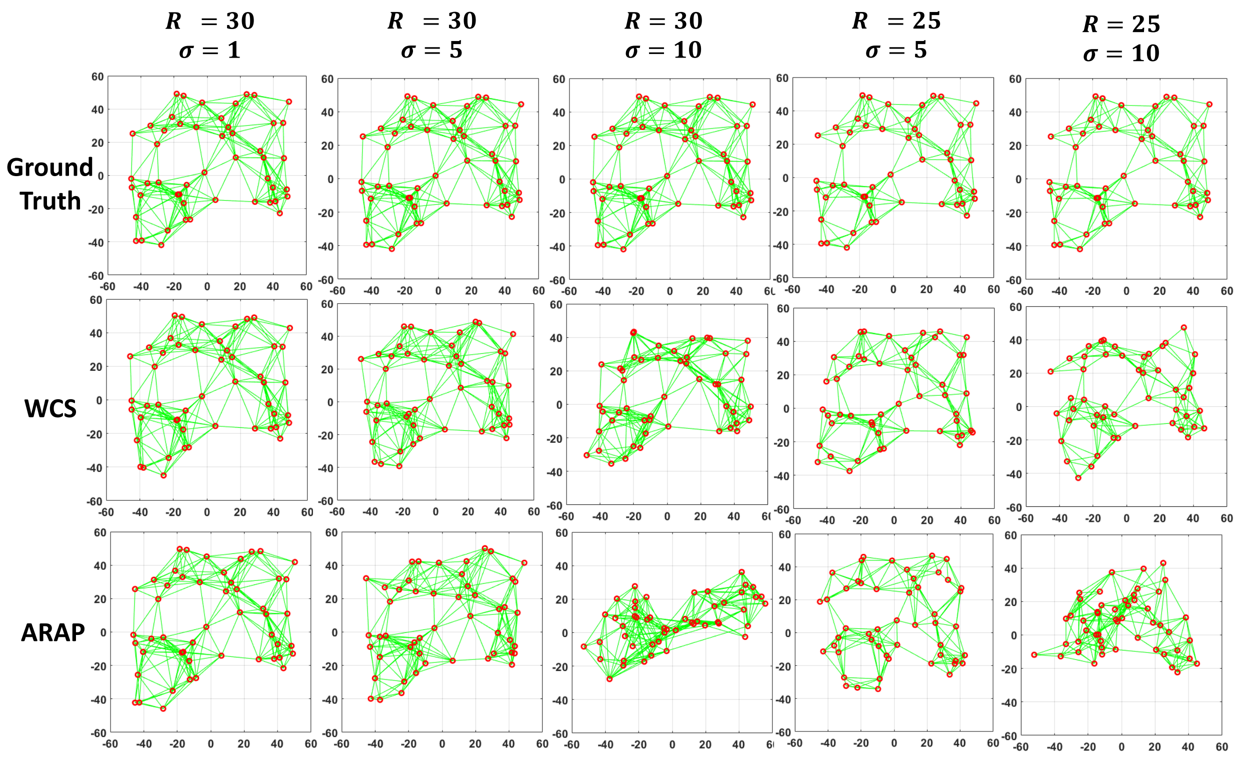


Fig. 9. Comparing the localization performances of different algorithms in different network size, ranging radius, and noise level settings. The proposed LMRC-based weighted component stitching method provides the most accurate and reliable results in these comparisons.

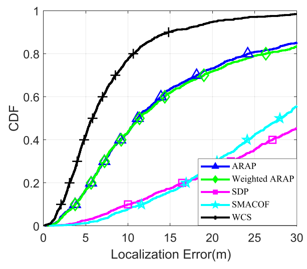


Fig. 10. $R = 25$, $\sigma = 1$.

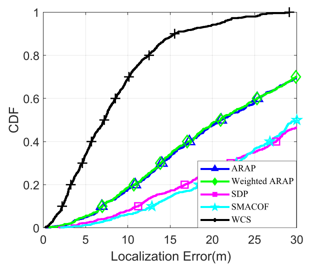


Fig. 11. $R = 25$, $\sigma = 5$.

$d_{i,j} = d_{j,i} = \|p_i - p_j\| + \varepsilon_{ij}$, $\varepsilon_{ij} \sim N(0,\sigma^2)$, where the noise level is controlled by varying σ.

Firstly, we compare WCS with other algorithms for graphs with different edge densities. The result is shown in Fig.8. By varying R in $\{45, 40, 35, 30, 25\}$, we control the edge density, i.e., the average node degree to change from 20 to 8 as shown by the red curve. When edges are dense, all the algorithms perform well in such dense graph, providing small localization error. However, when the edges become sparse, the performances of SDP and SMACOF drop sharply, showing less tolerance to the edge sparseness. WCS provides the best localization accuracy and robustness in sparse networks, which is better than ARAP. In later sections, we will analyze algorithm performances in sparse network settings when the centralized algorithms perform not well.

2) Performance in Sparse Networks: The average normalized error of localization is evaluated as the accuracy metric.

$$Err = \frac{\sum_{i=1}^{n} \|p_i - p_i'\|}{n} \tag{11}$$

where p_i' is the estimated localization of v_i; p_i is the ground-truth location; n is the number of nodes.

3) Visualize the Performance Difference: Fig.9 visualizes the localization results of different algorithms for different network size, sensing radius, and noise level settings. For page limitation, we only show the comparison with ARAP. SDP and SMACOF perform worse than ARAP as given in Fig.8. The parameter settings are given in the top row. In the result, there are three rows of figures: (1) ground truth; (2) results of WCS; (3) result of ARAP.

- When $R = 30, \sigma = 1$, which is the case with highest edge density and lowest noise level, WCS, ARAP and SMACOF provides localization results very close to the ground truth.
- When the ranging noises increase to $R = 30, \sigma = 5$, we can see the results of ARAP become worse, but WCS provides rather reliable localization result. When noises increase to $R = 30, \sigma = 10$, the results of ARAP is

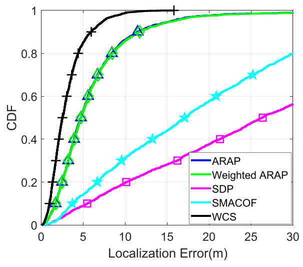


Fig. 12. $R = 30$, $\sigma = 5$.

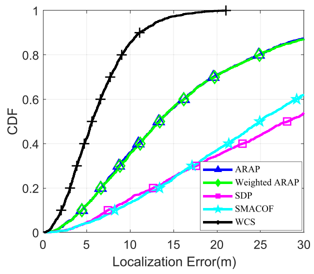


Fig. 13. $R = 30$, $\sigma = 10$.

TABLE II
THE AVERAGE LOCALIZATION ERROR IN DIFFERENT PARAMETER SETTING

Parameter Setting	%	WCS	WARAP	ARAP	SDP	SMACOF
$R = 30, \sigma = 5$	**45.8%**	**3.21**	5.86	**5.93**	28.29	18.88
$R = 30, \sigma = 10$	**62.2%**	**6.36**	16.67	**16.78**	30.01	26.91
$R = 25, \sigma = 5$	**55.6%**	**6.77**	15.18	**15.25**	31.59	28.61
$R = 25, \sigma = 10$	**64.6%**	**8.87**	25.18	**24.92**	33.16	31.78

unsatifactory, but the result topology of WCS is still meaningful.

- When the network becomes sparse, i.e., when $R = 25, \sigma = 5$, WCS shows better results than ARAP. The advantage is more clearly seen when $R = 25, \sigma = 10$. Although the location errors of WCS also increase, it provides much better results than ARAP.

4) Statistical Results: We conduct further experiments to compare the localization performances of different algorithms statistically under various parameter settings. In each setting, we carried out 100 experiments by generating network topologies randomly under the parameter setting, and calculated the cumulative distribution function of the localization errors.

The results are shown in Fig.10 to Fig.13. It can be seen that WCS provided the best performances in all these sparse network settings among all the algorithms. Weighted ARAP is to add weighted synchronization scheme to the synchronization of patches, using the patch's RR metrics as weights. The average location errors are summarized in Table.II.

Overall, WCS method reduces the location error more than 40% than the state-of-the-art ARAP algorithm in sparse and noisy network settings. It performs much better than that of the centralized algorithms. It also performs as well as other algorithms in dense networks. The results verify the importance to divide components by finding the LMRCs and to utilize RR metric for synchronization with weights.

VIII. CONCLUSION

This paper proposes to find redundant components in sensor networks to enhance the robustness of component stitching based network localization. It shows a 2-D graph has limited number of determinant ERCs and each ERC can be covered by a set of BRCs. It proposes a bipartite matching based algorithm to find a set of BRCs. Each BRC contains exactly one redundant edge and indicates the influence scope of the redundant edge. By a proposed redundancy ratio metric, a greedily merging method is developed to merge the BRCs into the locally most reliable components (LMRCs). Then the local coordinates of nodes are calculated in each LMRC and a weighted synchronization scheme is developed. It shows more than 45% location accuracy improvement than the state-of-the-art component stitching methods in sparse and noisy network settings. We show the WCS algorithm has complexity $O((\Delta n)^3 + hn^3)$. So it is suitable for reliable localization in sparse networks. The complexity improves when the average node degree is high. Further studies to exclude the non-rigid components and to utilize the of out-of-range information are potential directions.

REFERENCES

[1] B. Jackson and A. Nixon, "Stress matrices and global rigidity of frameworks on surfaces," *Discrete Comput. Geometry*, vol. 54, no. 3, pp. 586–609, Oct. 2015.

[2] D. K. Goldenberg *et al.*, "Localization in sparse networks using sweeps," in *Proc. MobiCom*, New York, NY, USA, 2006, pp. 110–121.

[3] D. K. Goldenberg *et al.*, "Network localization in partially localizable networks," in *Proc. IEEE INFOCOM*, vol. 1, Mar. 2005, pp. 313–326.

[4] Z. Yang, Y. Liu, and X.-Y. Li, "Beyond trilateration: On the localizability of wireless ad-hoc networks," in *Proc. INFOCOM*, Apr. 2009, pp. 2392–2400.

[5] Z. Yang and Y. Liu, "Understanding node localizability of wireless ad hoc and sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[6] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *J. Comput.*, vol. 2, pp. 135–158, Sep. 1973.

[7] D. J. Jacobs and B. Hendrickson, "An algorithm for two-dimensional rigidity percolation: The pebble game," *J. Comput. Phys.*, vol. 137, no. 2, pp. 346–365, 1997.

[8] B. Hendrickson, "Conditions for unique graph realizations," *SIAM J. Comput.*, vol. 21, no. 1, pp. 65–84, 1992.

[9] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc. INFOCOM*, vol. 4, Nov. 2004, pp. 2652–2661.

[10] S. Korkmaz and A. J. van der Veen, "Robust localization in sensor networkswith iterative majorization techniques," in *Proc. IEEE ICASSP*, Apr. 2009, pp. 2049–2052.

[11] A. M.-C. So and Y. Ye, "Theory of semidefinite programming for sensor network localization," *Math. Program.*, vol. 109, nos. 2–3, pp. 367–384, Sep. 2006.

[12] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^{2}o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 3607–3613.

[13] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proc. SenSys*, New York, NY, USA, 2004, pp. 50–61.

[14] X. Wang, Y. Liu, Z. Yang, J. Liu, and J. Luo, "ETOC: Obtaining robustness in component-based localization," in *Proc. ICNP*, Oct. 2010, pp. 62–71.

[15] L. Zhang, L. Liu, C. Gotsman, and S. J. Gortler, "An as-rigid-as-possible approach to sensor network localization," *ACM Trans. Sensor Netw.*, vol. 6, no. 4, 2010, Art. no. 35.

[16] M. Cucuringu, Y. Lipman, and A. Singer, "Sensor network localization by eigenvector synchronization over the Euclidean group," *ACM Trans. Sen. Netw.*, vol. 8, no. 3, Aug. 2012, Art. no. 19.

[17] X. Wang, J. Luo, Y. Liu, S. Li, and D. Dong, "Component-based localization in sparse wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, pp. 540–548, Apr. 2011.

[18] Y. Wang, T. Sun, G. Rao, and D. Li, "Formation tracking in sparse airborne networks," *IEEE J. Sel. Areas Commun.*, to be published.

[19] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. MOBICOM*, New York, NY, USA, 2001, pp. 166–179.

[20] H. Shen, Z. Ding, S. Dasgupta, and C. Zhao, "Multiple source localization in wireless sensor networks based on time of arrival measurement," *IEEE Trans. Signal Process.*, vol. 62, no. 8, pp. 1938–1949, Apr. 2014.

[21] J. Aspnes *et al.*, "A theory of network localization," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1663–1678, Dec. 2006.

[22] R. Connelly, T. Jordán, and W. Whiteley, "Generic global rigidity of body—Bar frameworks," *J. Combinat. Theory, B*, vol. 103, no. 6, pp. 689–705, 2013.

[23] R. Connelly, "Generic global rigidity," *Discrete Comput. Geometry*, vol. 33, no. 4, pp. 549–563, 2005.

[24] S. J. Gortler, A. D. Healy, and D. P. Thurston, "Characterizing generic global rigidity," *Amer. J. Math.*, vol. 132, no. 4, pp. 897–939, 2010.

[25] T. Sun, Y. Wang, D. Li, W. Chen, and Z. Gu, "Robust component-based network localization with noisy range measurements," in *Proc. ICCCN*, 2018.

[26] I. Shames, T. H. Summers, F. Farokhi, and R. C. Shekhar, "Conditions and strategies for uniqueness of the solutions to cooperative localization and mapping problems using rigidity theory," in *Proc. CDC*, Dec. 2015, pp. 6127–6132.

[27] D. Shamsi, N. Taheri, Z. Zhu, and Y. Ye, "Conditions for correct sensor network localization using SDP relaxation," in *Discrete Geometry and Optimization* (Fields Institute Communications). Heidelberg, Germany: Springer, 2013, pp. 279–301.

[28] M. Z. Win *et al.*, "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, May 2011.

[29] N. Patwari *et al.*, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[30] Y. Shen, S. Mazuelas, and M. Z. Win, "Network navigation: Theory and interpretation," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 9, pp. 1823–1834, Oct. 2012.

[31] Y. Shen and M. Z. Win, "Fundamental limits of wideband localization—Part I: A general framework," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 4956–4980, Oct. 2010.

[32] S. Mazuelas, A. Conti, J. C. Allen, and M. Z. Win, "Soft range information for network localization," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3155–3168, Jun. 2018.

[33] Z. Wang, S. Zheng, Y. Ye, and S. Boyd, "Further relaxations of the semidefinite programming approach to sensor network localization," *SIAM J. Optim.*, vol. 19, no. 2, pp. 655–673, Jan. 2008.

[34] J. Liu, Y. Zhang, and F. Zhao, "Robust distributed node localization with error management," in *Proc. 7th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, New York, NY, USA, 2006, pp. 250–261.

[35] J. Cota-Ruiz, J.-G. Rosiles, P. Rivas-Perea, and E. Sifuentes, "A distributed localization algorithm for wireless sensor networks based on the solutions of spatially-constrained local problems," *IEEE Sensors J.*, vol. 13, no. 6, pp. 2181–2191, Jun. 2013.

[36] U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed sensor localization in random environments using minimal number of anchor nodes," *IEEE Trans. Signal Process.*, vol. 57, no. 5, pp. 2000–2016, May 2009.

[37] U. A. Khan, S. Kar, and J. M. F. Moura, "DILAND: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1940–1947, Mar. 2010.

[38] Q. Shi, C. He, H. Chen, and L. Jiang, "Distributed wireless sensor network localization via sequential greedy optimization algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3328–3340, Jun. 2010.

[39] C. Soares, J. Xavier, and J. Gomes, "Simple and fast convex relaxation method for cooperative localization in sensor networks using range measurements," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4532–4543, Sep. 2015.

[40] A. Simonetto and G. Leus, "Distributed maximum likelihood sensor network localization," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1424–1437, Mar. 2014.

[41] N. Piovesan and T. Erseghe, "Cooperative localization in WSNs: A hybrid convex/nonconvex solution," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 1, pp. 162–172, Mar. 2018.

[42] B. Jackson, "Notes on the rigidity of graphs," School Math. Sci., Queen Mary, Univ. London, Tech. Rep., Jan. 2002.

[43] B. Jackson and T. Jordán, "Connected rigidity matroids and unique realizations of graphs," *J. Combinat. Theory, B*, vol. 94, no. 1, pp. 1–29, 2005.

[44] R. Diestel, *Graph Theory*, 5th ed. New York, NY, USA: Springer, Oct. 2010.

[45] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski, "On the maximum quasi-clique problem," *Discrete Appl. Math.*, vol. 161, nos. 1–2, pp. 244–257, Jan. 2013.

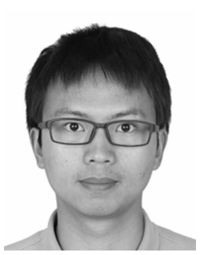
[46] A. García and J. Tejel, "Augmenting the rigidity of a graph in R^2," *Algorithmica*, vol. 59, no. 2, pp. 145–168, 2011.

[47] P. J. Besl and D. N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

Tianyuan Sun received the B.S. degree from the Department of Computer Sciences, Renmin University of China, in 2015, where he is currently pursuing the master's degree. His research interests include network localization algorithms and vision-based perception algorithms for robot and VR.

Yongcai Wang received the B.S. and Ph.D. degrees from the Department of Automation Sciences and Engineering, Tsinghua University, in 2001 and 2006, respectively. He was an Associate Researcher with NEC Laboratories, China, from 2007 to 2009. He was a Research Scientist with the Institute for Interdisciplinary Information Sciences, Tsinghua University, from 2009 to 2015. He was a Visiting Scholar with Cornell University in 2015. He is currently an Associate Professor with the Department of Computer Sciences, Renmin University of China. His research interests include network localization and combinatorial optimization and applications.

Deying Li received the M.S. degree in mathematics from Huazhong Normal University in 1988 and the Ph.D. degree in computer science from the City University of Hong Kong in 2004. She is currently a Professor with the Department of Computer Science, Renmin University of China. Her research interests include wireless networks, mobile computing, and algorithm design and analysis.



Zhaoquan Gu received the bachelor's and Ph.D. degrees in computer science from Tsinghua University in 2011 and 2015, respectively. He is currently a Professor with the Cyberspace Institute of Advanced Technology, Guangzhou University, China. His research interests include wireless networks, distributed computing, and big data analysis.

Jia Xu received the Ph.D. degree from RWTH Aachen. She was a Project Leader and a Senior Researcher with the Language Technology Group, DFKI, Germany. She was an Assistant Professor with the Institute for Interdisciplinary Information Sciences, Tsinghua University. She was an Associate Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. She is currently an Assistant Professor with the Department of Computer Sciences, City University of New York. Her research interests include machine learning and network sciences.