Multi-scale Anchor-free Distributed Positioning in Sensor Networks

Bastian Katz and Dorothea Wagner

***Abstract*—Positioning is one of the most fundamental problems in sensor networks: Given the network's connectivity graph and some additional local information on measured distances and/or angles, the goal is to recover the nodes' positions. Varying the assumptions regarding the nature and the quality of the measurements, there has been extensive research for both hardness results and practical, distributed, positioning schemes. This paper addresses these issues for a setting that appears to be most likely in real-world scenarios in the future – nodes can *roughly* measure distances and *relative* angles. We will show that this problem is $\mathcal{NP}$-hard like most positioning problems even for arbitrarily small errors. We will also propose an algorithm combining robustness to erroneous measurements and scalability in a completely distributed fashion and provide simulation results for networks of up to 128k nodes with varying errors.**

I. Introduction

In recent years, expectations of large-scale wireless sensor networks becoming a practical solution for many tasks in monitoring and data-gathering increased notably [1]. Despite the extensive research in both theory and practice, we still know the complexity of some of the most fundamental problems only with a high degree of idealization. Moreover, we also lack algorithms that could realize the vision of thousands of nodes that calibrate and process data in a distributed way, combining robustness and scalability.

Reconstructing the nodes' positions is one of those fundamental problems that arise early in sensor networks, known as the *positioning* or *localization* problem [2], [3]: On the one hand, there certainly is a need to know the nodes' positions for basic network operations from topology control [4] to geographic routing [5], [6] as well as for the obvious demand to know where data or events are sensed in a network. On the other hand, when talking about sensor networks, we cannot hope for any powerful infrastructure like GPS due to comparably high costs and size requirements for receivers and the restriction that GPS does not work indoors. Many localization algorithms assume the presence of a fraction of nodes which know their position, so-called *anchor-nodes* [7]–[11], but often, it will even be impossible to provide these beacons. Thus, *anchor-free* localization recently received more attention [9], [12]–[15], i. e., positioning solely based on known characteristics of the wireless channel and information that nodes realistically can measure locally about their relative positions. For example, nodes can estimate distances using the *received signal strength indicator (RSSI)* or directions using multiple receivers or directional antennas [16]. Anchor-free localization turned out to be a hard problem in almost any setting; we give a short overview on results in Table I. Localization is trivial if distances and directions between connected nodes are known and computationally easy if only directions are known. Given only distances, the problem becomes $\mathcal{NP}$-hard. Realizations of *unit disk graphs (UDG)* are even hard to approximate and remain hard if one knows either distances or directions. Recently, Basu et al. [21] have shown that the trivial case where distances and absolute directions are given becomes hard in the presence of arbitrarily small errors. This paper closes an annoying gap with the proof that the more realistic realization problem with known *relative* distances and directions is also $\mathcal{NP}$-hard in the presence of arbitrarily small errors.

The complexity of positioning problems, however, is not the only problem we face. We identified the following important issues which are not solved satisfactorily at the same time by any current positioning algorithm.

Distributed computation: One of the most natural requirements for algorithms in sensor network localization is that an algorithm has to be performed in a distributed way. Although the parallelism of computation in wireless networks could be seen as a benefit, most algorithms either are centralized or try to break down centralized approaches to a distributed environment. Apparently, only a quite small class of approaches really work fully distributedly [14], [15], [21].

TABLE I
Summary of hardness results of finding a valid embedding for a network

Realization Problem	*Complexity*
1-hop distances & directions (relative or absolute)	trivial
1-hop directions (relative or absolute)	in $\mathcal{P}$ (folklore, [13])
1-hop distances	$\mathcal{NP}$-hard [17]
Unit Disk Graph (UDG) only	$\mathcal{NP}$-hard [18]
UDG approximation	$\mathcal{NP}$-hard [19]
UDG with 1-hop distances	$\mathcal{NP}$-hard [20]
UDG with 1-hop directions	$\mathcal{NP}$-hard [13]
Absolute directions, distances, arbitrarily small errors	$\mathcal{NP}$-hard [21]
Relative directions, distances, arbitrarily small errors	$\mathcal{NP}$-hard (here)

B. Katz and D. Wagner are with the Faculty of Informatics, Universität Karlsruhe, Germany. e-mail:{bastian.katz, dorothea.wagner}@kit.edu

This work was partially supported by the German Research Foundation (DFG) within the Research Training Group GRK 1194 "Self-organizing Sensor-Actuator Networks".

Energy awareness/congestion: Many algorithms that work in a distributed manner, and thus seem to be well-suited for sensor networks, in fact demand all nodes to exchange data with all their neighbors as long as the algorithm is running. Among them are all approaches that use any local optimization, e. g. [14], [15]. Although these algorithms certainly are distributed, they ignore the fact that in sensor networks, this communication scheme is quite costly, as it causes interference and other side-effects of congestion.

Scalability and independence of deployment region: Comparing approaches for sensor network localization with the vision of large-scale networks [22], the most striking shortcoming of many proposed algorithms is that first, the issue of scalability almost never seems to be addressed at all, and second, algorithms highly rely on implicit assumptions made regarding the deployment of nodes, like the convexity of the deployment region. Typically, algorithms are evaluated on scenarios with 200 to 1000 nodes in a rectangular area, leaving the question open whether the algorithm works in different setups, especially how much the size of the network influences quality and runtime.

Our contribution to this facet of the localization problem is a novel algorithm scheme that overcomes these problems by combining the best centralized localization techniques with the powerful idea of multi-scale optimization: The proposed algorithm distributedly reduces the positioning problem to small subproblems which can be solved locally. Solutions are aggregated hierarchically, allowing for multi-scale optimization in overlay networks without any of the disadvantages mentioned above.

This paper is organized as follows: In Section II, we give a short description of the problem setup and some preliminaries. Section III proves the hardness of the localization problem with distances and relative directions in the presence of arbitrarily small errors. Our algorithm is presented in Section IV along with simulation results in Section V. Conclusions are presented in Section VI.

II. Preliminaries

Throughout this paper, we model a network as an undirected graph $G = (V, E)$ with an embedding $\mathbf{p} : V \to \mathbb{R}^2$ and an orientation $\mathbf{o} : V \to [0, 2\pi)$ of the nodes. With respect to this embedding and orientation, distances $d_\mathbf{p}$ and relative directions $\omega_{\mathbf{p},\mathbf{o}}$ between pairs of nodes $u \neq v \in V$ are canonically denoted as

$$\begin{aligned} d_\mathbf{p}(u,v) &:= |\mathbf{p}(u) - \mathbf{p}(v)| \text{ and} \\ \omega_{\mathbf{p},\mathbf{o}}(u,v) &:= \measuredangle(\mathbf{p}(v) - \mathbf{p}(u)) - \mathbf{o}(u) \pmod{2\pi} \ . \end{aligned}$$

Note that, unlike distances between nodes, directions are not symmetric. Thus, the input of a localization problem, i. e., local measurements on distances and directions, contains per edge one distance, but two direction measurements, one for each node incidence. We therefore model the input distances as function $d : E \to \mathbb{R}_+$ and the input directions as function $\omega : E \to [0, 2\pi)^2$. To facilitate readability, we will rather use $\omega(u, v)$ and $\omega(v, u)$ as above for an edge $\{u, v\} \in E$ instead of defining $\omega(\{u, v\})_1$ and $\omega(\{u, v\})_2$.

We denote a node v's 1-hop neighborhood in a graph G with $\mathcal{N}(G, v)$, its k-hop neighborhood with $\mathcal{N}^k(G, v)$ (both including v). A set of nodes $V' \subset V$ is called a *dominating set*, if for each node v, either v itself or one of v's neighbors is in V'.

III. Hardness Result

Recently, Basu et al. [21] have shown that the localization problem is $\mathcal{NP}$-hard if nodes know distances and *absolute* angles, i. e., nodes measure angles against a common north pole, both with an arbitrarily small error. We extend this problem to the less artificial case where nodes measure angles against their respective axis, i. e., without assuming any global knowledge:

*Problem 1 (*Error-Realization*):* Given a graph $G = (V, E)$, edge lengths $d : E \to \mathbb{R}_+$, relative edge directions $\omega : E \to [0, 2\pi)^2$ and small ϵ, $\delta > 0$, is there an embedding $\mathbf{p} : V \to \mathbb{R}^2$ and an orientation $\mathbf{o} : V \to [0, 2\pi)$, such that for all u, v with $\{u, v\} \in E$

$$\frac{d_\mathbf{p}(u,v)}{d(u,v)} \in [1-\epsilon; 1+\epsilon] \text{ and}$$

$$\omega_{\mathbf{p},\mathbf{o}}(u,v) - \omega(u,v) \in [-\delta; \delta] \mod 2\pi \ ?$$

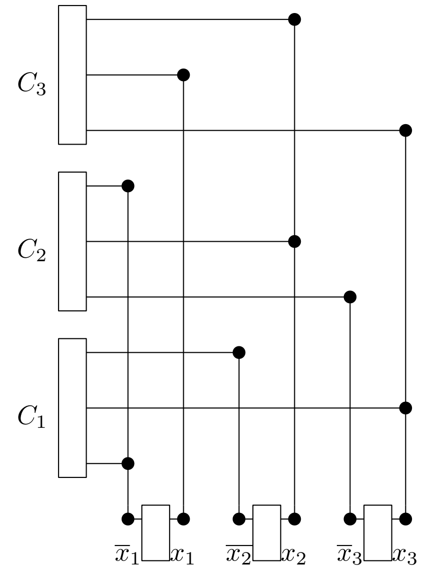
In the following, we prove that it is $\mathcal{NP}$-hard to find an embedding such that the measured distances and angles do not differ from the embedding by more than given, arbitrarily small factors and angles, respectively:

Theorem 1: Error-Realization is $\mathcal{NP}$-hard even for fixed, arbitrarily small error bounds ϵ, δ.

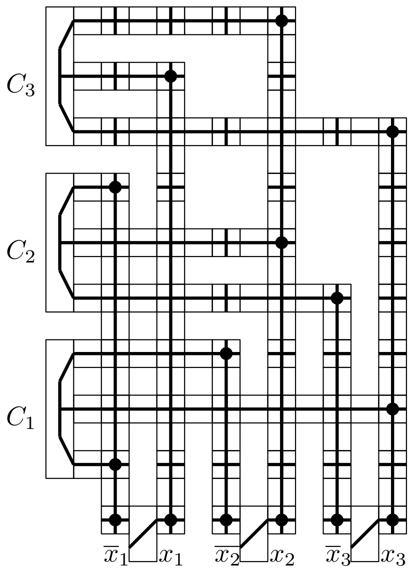
Proof: We prove this theorem by a reduction from 3Sat: Given an instance of 3-Sat, we draw the corresponding instance canonically as shown in Figure 1 with the building blocks, i. e., variables, wires, crossings, connectors and clauses. From this drawing, we derive an input to our Error-Realization problem as follows.

First, we observe how to design an input to the Error-Realization problem in order to force a graph to be embedded with fixed angles in any valid embedding: Let a graph G have a cycle of nodes we want to be realized as a polygon with prescribed angles. If we choose the input directions ω to all point outward (or all inward) by an angle of δ with respect to an arbitrary orientation of the nodes (see Figure 2), every valid embedding of G that differs from ω by no more than δ for any edge-node incidence, embeds G as a polygon according to the given angles. We will exploit this fact in the following.

Based on the arguments above, we construct variables by building a *rectangle* of four nodes. We then introduce an additional inner edge that can deviate from being parallel to the rectangle's sides by δ in both directions (see Figure 3). We also assign lengths to all edges, namely $a, b, c \in \mathbb{R}_+$ such that the triangle shown in Figure 3 is valid. One can see that in every valid embedding of the six nodes, first, angles



(a) 3SAT as Schematic drawing



(b) 3SAT from building blocks

Fig. 1. Schematic drawing of a 3SAT instance (a) and in a grid-like fashion with building blocks, i. e., variables, wires (horizontal/vertical), crossings, connections and clauses (b).

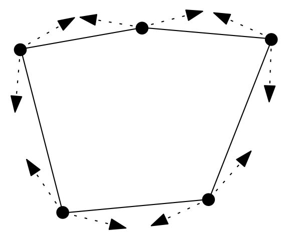


Fig. 2. A polygon with fixed angles. Input directions are dotted and differ all by δ from a polygon with the given angles.

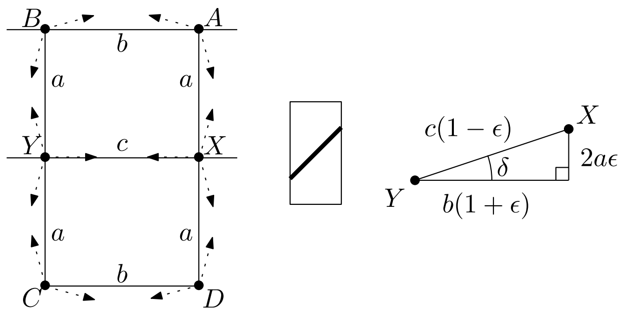


Fig. 3. The gadget for variables has fixed angles at vertices A, B, C and D. The edge YX can deviate from being parallel to AB (right). Input edge lengths are constructed for given ϵ, δ (left).

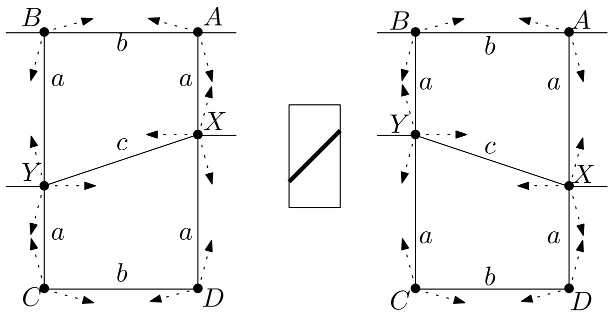


Fig. 4. Variables have only two valid embeddings, corresponding to truth values false (left) and true (right).

must comply with one of the drawings in Figure 4, second, the edges AB and CD are maximally stretched, i. e., have lengths $(1+\epsilon)b$, while the edge XY has its minimum length of $(1-\epsilon)c$, and third, either edges XA and YC have length $(1+\epsilon)a$ and BY and DX have lengths $(1-\epsilon)a$ or vice versa. We will call the former a `true` assignment, the latter a `false`. Now, the edges XA and BY correspond to the literals x and $\overline{x}$ respectively, i. e., a length $(1+\epsilon)a$ corresponds to a `true` literal, length $(1-\epsilon)a$ to `false`. Wiring, connecting and crossing is comparably easy. The gadgets are shown in Figure 5(a) to 5(d). Essentially, they all are rectangles with fixed angles, which therefore must have the same lengths for opposite edges. Variables and wires have all directions pointing outward, while in crossings and connectors, directions point inward by δ. These two kinds of gadgets are put together alternately (see Figure 1(b)). Note that all rows have heights of $\approx a$, but the width of columns is $\approx a$ only for columns corresponding to literals, whilst its necessary for the other columns containing horizontal links to have width $\approx b$ like variables.

The last gadget is a clause, shown in Figure 5(e). If we choose x, such that

$$(1-\epsilon)x = (3(1+\epsilon)+2(1-\epsilon))\cdot a \Leftrightarrow (5+\epsilon)a = (1-\epsilon)x \ ,$$

i. e., even with the edges with length $\approx x$ having the shortest possible length $(1-\epsilon)x$, there must be no more than two 'short' edges on the right side, that is, at least one connected literal must be `true`, and more are always possible. On the other hand, it is easy to see how a valid assignment can be

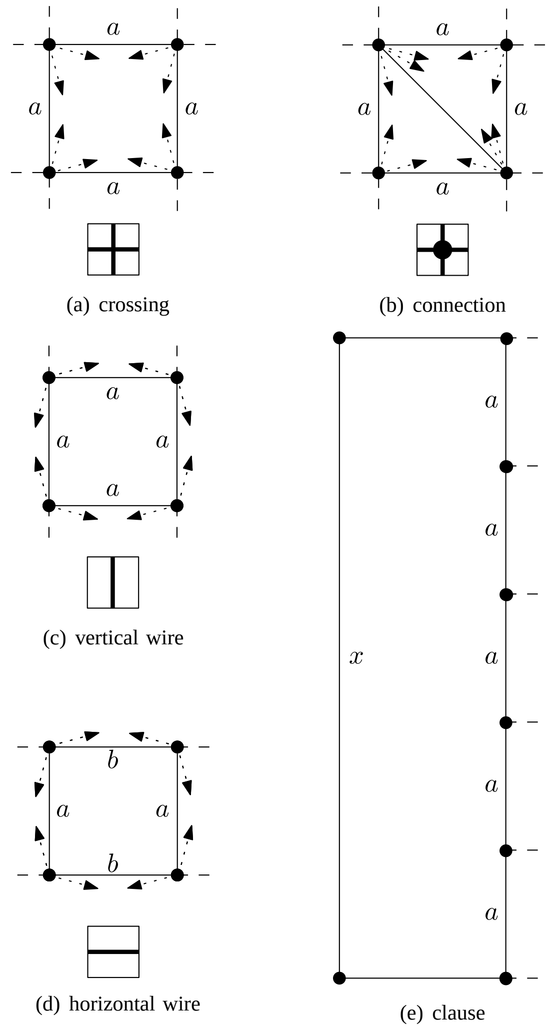


Fig. 5. Gadgets. Symbols correspond to blocks in Figure 1(b).

transformed into a valid embedding: Rows that correspond to an occurrence of a literal have the respective height, the other rows have height $(1+\epsilon)a$. Columns that correspond to a literal again have the width induced by the literal's value, other columns have width $(1+\epsilon)b$. ■

IV. Hierarchical Localization

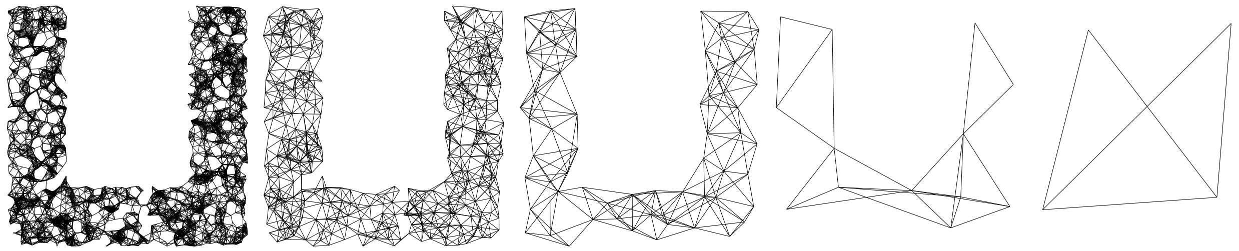
Although almost every anchor-free localization problem is hard – especially in the presence of errors, recent work has shown strengths and weaknesses of existing heuristic approaches: Obviously, any kind of purely incremental algorithm cannot prevent an increase in errors, but there are many other techniques that work quite well in some sense. The first observation is, that we have many techniques at hand that could be used to localize small networks, although the best of them typically work in a centralized manner and sometimes are highly complex, e. g., use linear or semidefinite programming as in [23]. They seem to be unsuitable for larger networks, but on the other hand, they can still be used to let nodes localize some reasonably small neighborhood.

For those algorithms that really work in a distributed way, a typical approach is to first find a *folding free* embedding, i. e., an embedding without overlappings of different parts of the network, and then perform any kind of local stress minimization. Examples are AFL [14] and EIGEN [15]. In order to embed the network folding free, the former spreads nodes between five heuristically chosen reference nodes, the latter uses a distributed version of a spectral graph layout algorithm. Both these approaches implicitly rely on the assumption that the deployment area has some regular shape, e. g., is convex ([14]). Moreover, these approaches (and more) extensively use local stress minimization to gain a good localization. This can certainly be done distributedly, but it incorporates mass pairwise communication – continuously, every node has to send its current position and receive positions from its neighbors. We do believe that this kind of mass concurrent communication has to be restricted as far as possible for any localization scheme to be applied in real networks, since this really is the worst case in terms of network congestion and energy consumption.

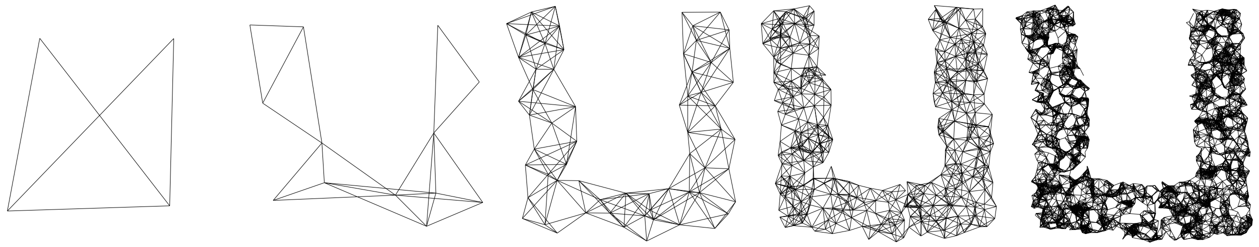
Our approach, in contrast, tries to overcome the mentioned problems following two basic ideas: First, rather than being overly distributed, it seems reasonable for us to let a sample of nodes localize some (constant-size) neighborhood in a centralized fashion. This minimizes the communication overhead without violating the restriction that nodes only have small memory and computational power. Second, because iteratively putting those local solutions together would still lead to increasing errors in large networks, we adapt the idea of graph filtration and multi-scale graph layout, which is known from graph drawing [24]. Graph filtration here denotes the process to successively restrict a graph to a fraction of nodes, i. e., starting with all nodes, a filtration is a sequence $V = \mathcal{V}_0 \supset \mathcal{V}_1 \supset \cdots \supset \mathcal{V}_k = \{\hat{v}\}$. In graph drawing, the usual way is then to find a layout *top-down*: Given a layout for the nodes from $\mathcal{V}_i$, the nodes from $\mathcal{V}_{i-1} \setminus \mathcal{V}_i$ are placed according to some good guess based on the layout of $\mathcal{V}_i$; the resulting layout is refined by local optimization and so on. In [25], a variant has been proposed for sensor network localization, but again only in a centralized fashion. Moreover, this algorithm works only top-down, guessing initial distances and directions by dead reckoning. It thus cannot work for arbitrarily large networks, as these initial errors will become arbitrarily high.

Our approach consists of both a bottom-up and a top-down stage: In the bottom-up pass, we first choose a fraction of the nodes to collect and localize their neighborhood. For those pairs of chosen nodes that are close to each other, we introduce virtual edges (i. e., multi-hop connections), deriving distances and directions from the local solutions (see Figure 6). This filtration step is done recursively on the resulting overlay network until we end up with a single node, to which we assign arbitrary coordinates.

In the top-down stage, knowing local solutions and coordinates for the chosen subset of nodes is enough to assign coordinates to all nodes of one abstraction level. Optionally, local problems can be solved again to refine local solutions



(a) Bottom-up stage: The input network (left) is filtered repeatedly until we reach an instance, which can be solved by a single node: On every level, a dominating set of nodes is chosen, virtual edges span between chosen nodes that lie in some neighborhood. Edge distances and directions are derived from local solutions on the current level.



(b) Top-down stage: Starting with the positions of the topmost level, the nodes of the next levels can be positioned successively using local solutions on the respective level.

Fig. 7. Two stages of the hierarchical localization.

when the chosen subset of nodes knows their final positions. This process is depicted in Figure 7. Note that a main difference to other approaches is that in the top-down stage every node is assigned a position only once, without any refinement when other nodes have been placed, saving a lot of communication.

A more detailed view on this technique is given in Algorithm 1: Given a connected graph $G = (V, E)$ together with erroneous edge lengths $d : E \to \mathbb{R}_+$ and relative edge directions $\omega : E \to [0, 2\pi)$, we first choose a dominating subset of nodes $V' \subset V$. For the nodes in V', we use any central algorithm to localize their 3-neighborhood. For pairs of nodes that mutually lie in this neighborhood,

$$E' := \left\{ \{u, v\} \in \binom{V'}{2} \mid u \in \mathcal{N}^3(G, v) \right\} \quad ,$$

we derive distances $d' : E' \to \mathbb{R}_+$ and directions $\omega' : E' \to [0, 2\pi)$ from the local solutions. We recursively apply the algorithm to $G' = (V', E')$ and, given a localization for V' from this recursion, localize the nodes in $V \setminus V'$ using the (refined) local solution of the closest node in V'.

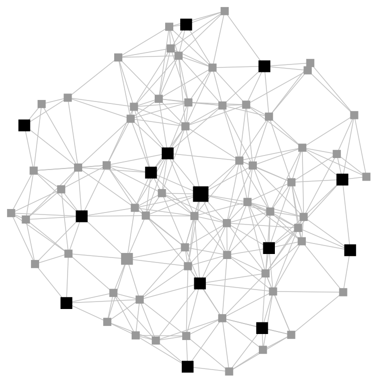
Note that this scheme still has two degrees of freedom: The choice of the algorithm applied to local problems and the choice of V'. We leave the choice of the algorithm open, but assume that we select V' in the following way: For a fixed number k of rounds, every node which neither has a neighbor in V' nor is itself a member of V' selects itself with a fixed probability $0 < c < 1$. In an additional round, every node still fulfilling this condition selects itself with probability 1. This clearly ensures that a dominating set V' is chosen after $k + 1$ rounds, and it further ensures that we can bound every node's probability to be selected independently of the network's structure by

Algorithm 1 LOCALIZE$(G = (V, E), d, \omega)$

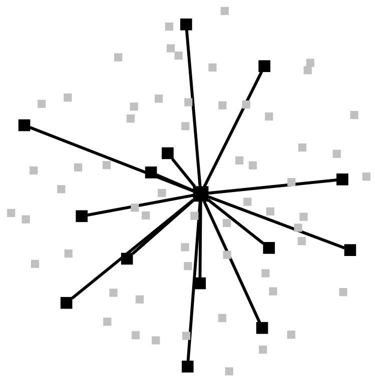
```
 1: if $|V| = 1$ then
 2:     return any localization
 3: end if
 4: choose dominating subset $V' \subset V$
 5: $E' \leftarrow \emptyset$
 6: for all $v \in V'$ do
 7:     $G_v \leftarrow G[\mathcal{N}^3(G, v)]$
 8:     $(\mathbf{p}_v, \mathbf{o}_v) \leftarrow$ SOLVECENTRALLY$(G_v, d, \omega)$
 9:     for all $u \in V' \cap \mathcal{N}^3(G, v)$ do
10:         $E' \leftarrow E' \cup \{\{v, u\}\}$
11:         $d'(v, u) \leftarrow d_{\mathbf{p}_v}(v, u)$
12:         $\omega'(v, u) \leftarrow \omega_{\mathbf{p}_u, \mathbf{o}_u}(v, u)$
13:     end for
14: end for
15: $(\mathbf{p}', \mathbf{o}') \leftarrow$ LOCALIZE$(G' = (V', E'), d', \omega')$
16: for all $v \in V'$ do
17:     for all $u \in \mathcal{N}^3(v) \setminus V'$ do
18:         $(\mathbf{p}(u), \mathbf{o}(u)) \leftarrow (\mathbf{p}(v), \mathbf{o}(v)) \oplus (\mathbf{p}_v(u), \mathbf{o}_v(u))$
19:     end for
20: end for
21: return $(\mathbf{p}, \mathbf{o})$
```

$$c \le P[v \in V'] \le 1 - c \cdot (1 - c) \quad ,$$

since at least in the case that a node is not chosen in the first round while one of its neighbors is, the node will not be selected at all. As an example, if $c = 1/2$ and $k = 1$, each node is chosen at least with probability $1/2$ and at most with probability $3/4$, as there is at least a $1/4$ chance that one



(a) An active node (center) collects its 3-hop neighborhood by local communication. Other active nodes are drawn black, too.



(b) Routes to close active nodes are stored. They become neighbors on the next-level overlay network. Distances and directions are derived from local solution.

Fig. 6. Filtering from a node's point of view.

neighbor is chosen in the first round while the node itself is not.

Moreover, the construction of the network G' guarantees connectivity: For any node $v \in V$ let $\mathrm{dm}(v) \in V'$ denote the selected node that is closest to v in G (its dominator, i. e., either v itself or one of v's neighbors). For any path $v_1, \ldots, v_l$ in G, the nodes $\mathrm{dm}(v_i)$ and $\mathrm{dm}(v_{i+1})$ have at most three hops distance and are thus connected by an edge in G' (see Figure 8).

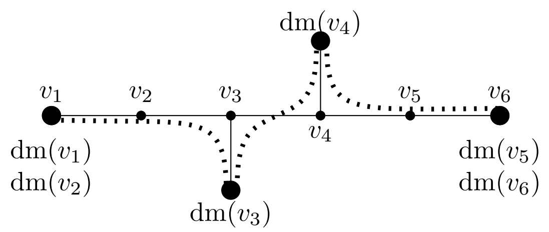


Fig. 8. A path $v_1, \ldots, v_6$. dominators of successive nodes are at most three hops apart and the graph stays connected.

This algorithm terminates with expected $\mathcal{O}(\log D)$ recursions if G has diameter D: For a node v_i on a path $v_1, \ldots, v_l$, there is a constant lower bound $p > 0$ on the probability that $v_i, v_{i+2} \in V'$ and $v_{i+1} \notin V'$, which means that the distance between $\mathrm{dm}(v_i) = v_i$ and $\mathrm{dm}(v_{i+2}) = v_{i+2}$ shrinks from 2 to 1. We have at least $\lfloor l/3 \rfloor$ such triples for which this *independently* holds, and thus the expectation for the number of nodes on the path which introduce a shortcut is at least $p \cdot \lfloor l/3 \rfloor$ and the expectation for the length of the shortest path from $\mathrm{dm}(v_1)$ to $\mathrm{dm}(v_l)$ is shorter than l by a constant factor.

As the expected number of remaining nodes in each recursion decreases at least by the factor $1 - c \cdot (1 - c)$, we know that during all recursions we have a total of $\mathcal{O}(|V|)$ invokings of SOLVECENTRALLY:

$$|V| \cdot \sum_{i=0}^{k} (1 - c \cdot (1-c))^i \leq \frac{|V|}{c \cdot (1-c)}$$

Although it is not possible to give a constant bound on the maximum node degree or on the maximum size of a three-hop neighborhood during the execution, our experiments on geometric graphs with their close interrelation between Euclidean and graph distances have shown that both node degrees and problem sizes did not increase in the overlay networks (see Section V).

A. Distributed Algorithm

As claimed, Algorithm 1 can completely be implemented in a distributed way; an outline is given in Algorithm 2. Starting with a setup where every node knows distance and direction measurements $d_{\mathcal{N}}, \omega_{\mathcal{N}}$ for its one-hop neighbors $\mathcal{N}$, every node performs the $k + 1$ rounds of the selection algorithm, each including a communication with its direct neighbors. Afterwards, the nodes have to share information with their 3-neighborhoods to let close selected nodes know each other. If a node has chosen to be active on the next level, it applies the centralized algorithm to the current neighborhood and then knows its neighbors on the next level (with estimated distances and directions) as well as the hops to route information there in the current neighborhood. This procedure is repeated until on some level the node either has no more neighbors or it decided not to join the next level. In the first case, the node localizes itself to some arbitrary position, in the latter, the node waits until it receives a position from some other node. In any case, as soon as a node knows a localization, it sends localizations to its neighbors on all levels in which the node was active, based on its own position and the local solutions.

Note that on each level, each node has to communicate only a constant number of times with its neighbors on that level while the number of levels is in $\mathcal{O}(\log D)$. On the other hand, distances between communicating nodes increase with the level. However, empirically, in geometric graphs, the decreasing number of active nodes compensates the growing distances of neighbors easily and thus, the number of one-hop messages sent is in $\mathcal{O}(|V| \log D)$. Nevertheless, routing communication between higher level neighbors is a crucial point: If we denote the overlay networks as $G = G_0, G_1, \ldots G_k$

Algorithm 2 LOCALIZEDISTRIBUTED($\mathcal{N}, d_{\mathcal{N}}, \omega_{\mathcal{N}}$)

```
$\mathrm{lvl} \leftarrow 0$
$\mathcal{N}_0 \leftarrow \mathcal{N}$, $d_0 \leftarrow d_{\mathcal{N}}$, $\omega_0 \leftarrow \omega_{\mathcal{N}}$
active $\leftarrow$ true
while active do
    perform dominating set selection algorithm
    exchange 3-hop information with $\mathcal{N}_{\mathrm{lvl}}$ neighbors
    if node is selected then
        solve localization problem on $\mathcal{N}^3_{\mathrm{lvl}}$
        $\mathcal{N}_{\mathrm{lvl}+1} \leftarrow$ selected nodes from $\mathcal{N}^3_{\mathrm{lvl}}$
        store hops in $\mathcal{N}^3_{\mathrm{lvl}}$ for neighbors $\mathcal{N}_{\mathrm{lvl}+1}$
        derive $d_{\mathrm{lvl}+1}$ and $\omega_{\mathrm{lvl}+1}$ from local solution
        $\mathrm{lvl} \leftarrow \mathrm{lvl} + 1$
    else
        active $\leftarrow$ false
    end if
end while
wait for first localization
send localizations to (inactive) nodes in $\mathcal{N}^3_0$ to $\mathcal{N}^3_{\mathrm{lvl}-1}$
```

$k \in \mathcal{O}(\log D)$, neighbors in G_i can be up to 3^i hops apart in G (at most D hops). Thus, it is not only impossible for a node v to store complete routes to neighbors on higher levels, but we would also need to pass this information along with the messages, which is prohibitive, too.

A quite straightforward way to solve this problem would be to let the nodes on routes know how to route future messages: Given that messages can be routed between *neighbors* in G_i, which is trivially true in G_0, every edge in G_{i+1} is composed of at most 3 hops in G_i. Thus, sending an additional message containing these 3 hops for every edge in G_{i+1} is enough to inform all nodes about this edge. By the selection of active nodes the construction of the overlay connections, each node usually is part of at most a constant number of edges on a certain level. Hence, additional memory of $\mathcal{O}(\log D)$ per node is sufficent for this task.

Another way to route messages is to exploit the fact that every node has to store local solutions for every level in which it was active. Without storing any additional routing information, these local neighborhoods are sufficient to route messages in G_i with at most $2i + 1$ IDs enclosed: Given a message to be sent from a node $s \in G_i$ to one of its neighbors $t \in G_i$, s knows a path

$$\underbrace{s - v_{i-1} - w_{i-1} - t}_{\text{path in } G_{i-1}}$$

to t in G_{i-1}. The node s also knows such a path to v'_1 in G_{i-2} and can add this information to the route:

$$\underbrace{s - v_{i-2} - w_{i-2} - v_{i-1}}_{\text{path in } G_{i-2}} - w_{i-1} - t$$

Repeatedly applying this scheme, s can send the messages with waypoints $v_0 - w_0 - v_1 - w_1 - \cdots - v_{i-1} - w_{i-1} - t$. While the message is routed, passed waypoints can be removed

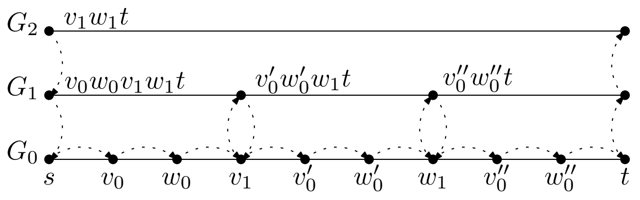


Fig. 9. Routing a message between two nodes s, t in G_2 with waypoints.

and additional waypoints are added as above. The number of waypoints then never exceeds $2i+1$. An example is depicted in Figure 9: The node s initially knows the target node t and adds the hops known in G_1 and G_0 to $v_0w_0v_1w_1t$. The message passes v_0 and w_0 and the next node from G_1, v_1 adds the waypoints v'_0 and w'_0, which are necessary to reach w_1 and so on.

In any case, every node needs to store local solutions for every level in which it was active, and as long as local solutions do not grow larger than a constant bound, memory consumption for every node is in $\mathcal{O}(\log D)$.

V. IMPLEMENTATION AND SIMULATION RESULTS

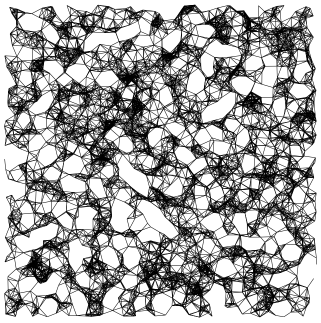
We implemented the proposed algorithm in its centralized variant to run a variety of simulations on networks of different shapes, densities, sizes and error characteristics. In our current implementation, we use a very simple force-directed layout algorithm to solve local subproblems and used $k = 10$ and $c = 5\%$ for the selection of a dominating set. For all tests, we evaluated a variation of the *global stress RMS (GSR),* i. e., the quadratic mean of the pairwise relative distance violation between the original distances between nodes in the graph d_{ij} and the distances in the reconstruction $\hat{d}_{ij}$,

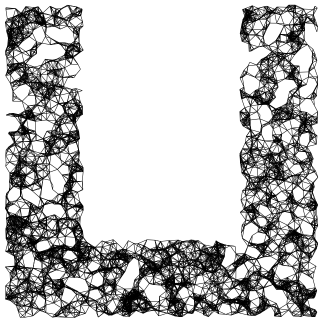
$$GSR := \sqrt{\frac{2}{n(n-1)} \sum_{i<j} \left(\frac{d_{ij} - \hat{d}_{ij}}{d_{ij}} \right)^2} \; ,$$

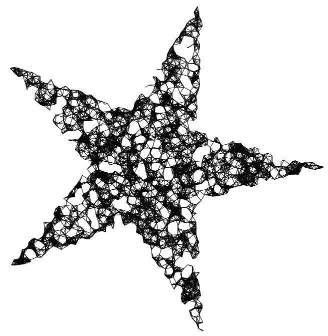
which is very similar to the *global energy ratio* (GER) as defined by Priyantha et al. [14], but better suited for comparing different network sizes. However, both indices equally measure both large- and small-scale errors, i. e., small values indicate that first, there is not much stress in local neighborhoods, and second, the global picture of the network is recovered correctly.

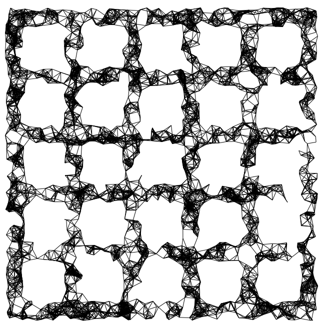
Deployment Regions and Radio Range: We evaluated the implementation on six different deployment regions. Region types were squared, u-shaped, star-shaped, grid-like, donut-shaped and one of an irregular pattern (see Figure 10). In any case, the nodes were randomly distributed over the area; their radio range was then uniformly chosen to generate a specified average node degree, typically between 8 and 12.

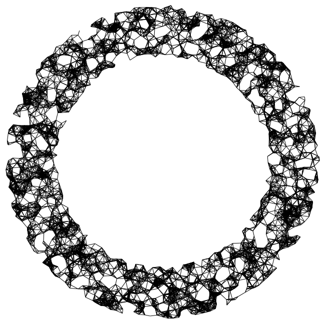
Errors: To evaluate the robustness of our algorithm, we introduced noise for both distances and directions, i. e., the input distances were normally distributed around the original values with a standard deviation of up to 30%, input directions with a standard deviation of up to 30°. Note that this error


(a) square


(b) u-shape


(c) star-shape


(d) grid-like


(e) donut

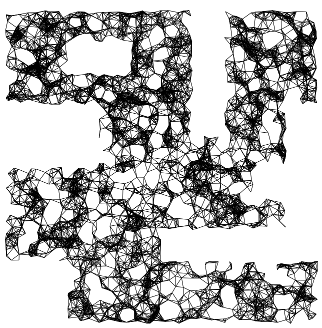
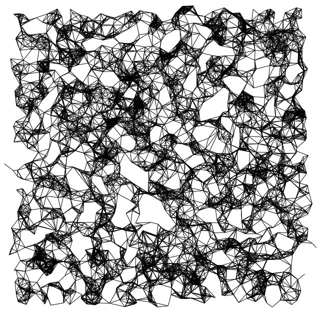
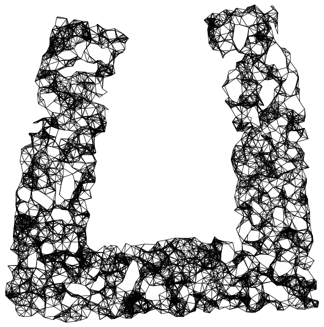
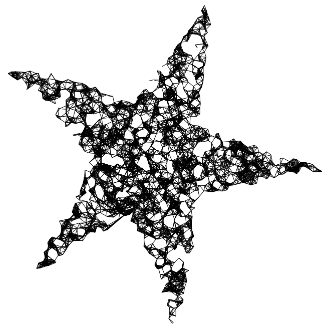
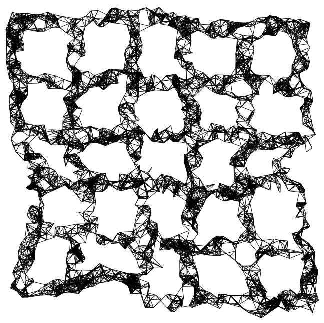

(f) irregular

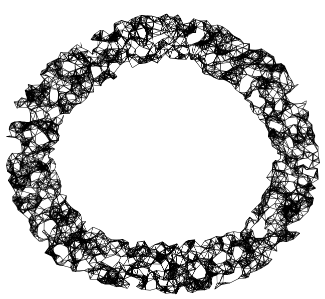
Fig. 10. Deployment areas; in each region, the same number of nodes – here: 2500 – are placed uniformly at random. Communication radii are uniformly chosen to achieve a specified average node degree (here: 12).


(a) GSR=0.053 (0.058)


(b) GSR=0.097 (0.078)


(c) GSR=0.075 (0.068)


(d) GSR=0.104 (0.132)


(e) GSR=0.089 (0.071)

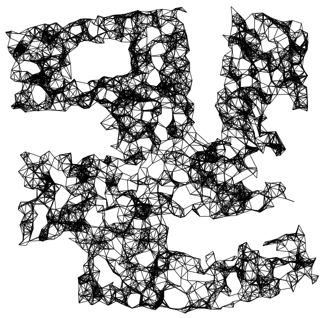

(f) GSR=0.083 (0.079)

Fig. 11. Exemplary localizations with standard deviation of distance error 15% and angular error 15° (average GSR values in parentheses). Both local and global structure are recovered correctly.

model does not give a bound for possible errors. As in real-life scenarios, a big challenge for the algorithm is to deal with some measurements being almost completely wrong.

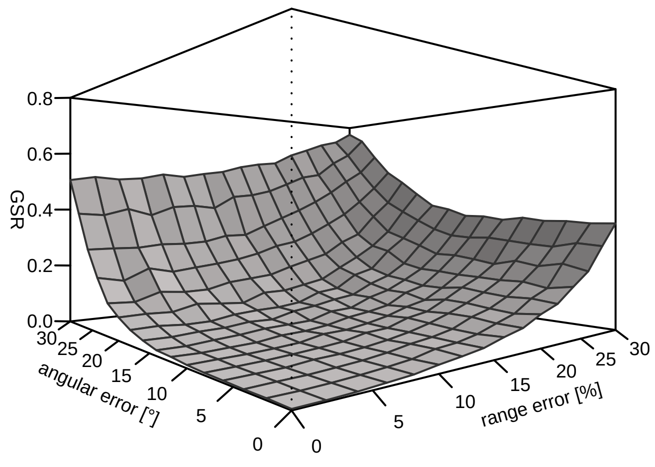
Figure 11 shows exemplary localizations of (comparably small) networks of 2500 nodes with errors of standard deviations approximately 15% and 15° respectively. The results show how close to reality localizations with typical GSR values (0.05 to 0.1, depending on the deployment area) are. In Figure 12, results for different error characteristics are depicted for networks with 5000 nodes. For these tests, we chose an average degree of 12, matching with the simulations from [21].

For each combination, experiments were repeated until statistical significance has been attained with an α-level of 0.95 and a confidence interval of length 0.02 for GSR[1]. We get very similar results for all scenarios: The algorithm is able to recover node positions very accurately for errors even with a standard deviation of 15% and 15° and above. For much larger errors, overall quality decreases due to failures of the local algorithm. Not surprisingly, best results are achieved for the typical benchmark scenarios of squared and star-shaped deployment regions, but our algorithm also performed very well on the deployment regions that seemed to be much harder to localize before.

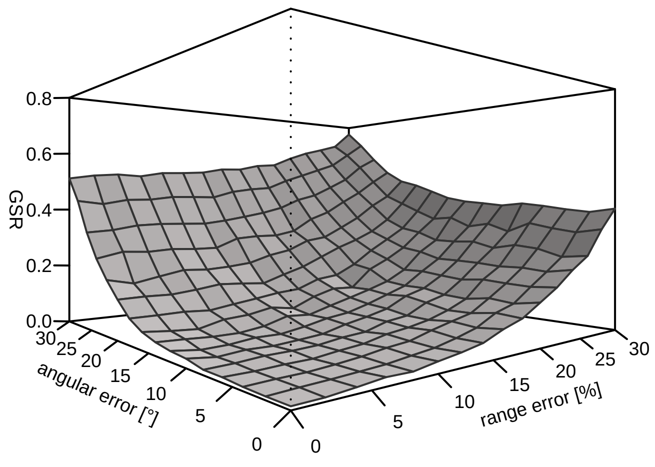
[1]at most 250 runs per setup

Additionally, we ran our algorithm on different network sizes of up to 128000 nodes. As expected, the runtime increased only linearly with the number of nodes (see Figure 13(a)), while the quality is independent of the network's size (see Figure 13(b)). Larger errors resulted in slightly higher runtimes only as an artefact of the slower convergence of the optimization algorithm. This completely complies with the demand for scalable localization algorithms.

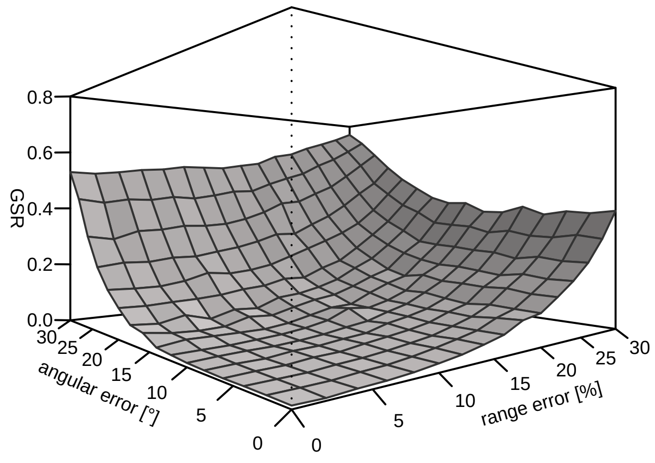
An evaluation of the algorithm with varying average node



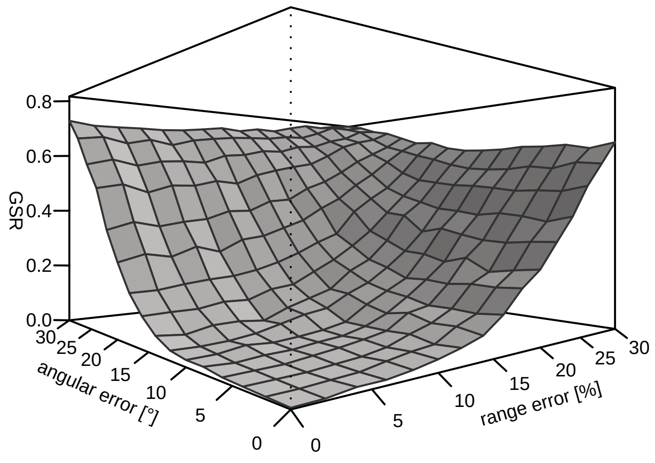
(a) square



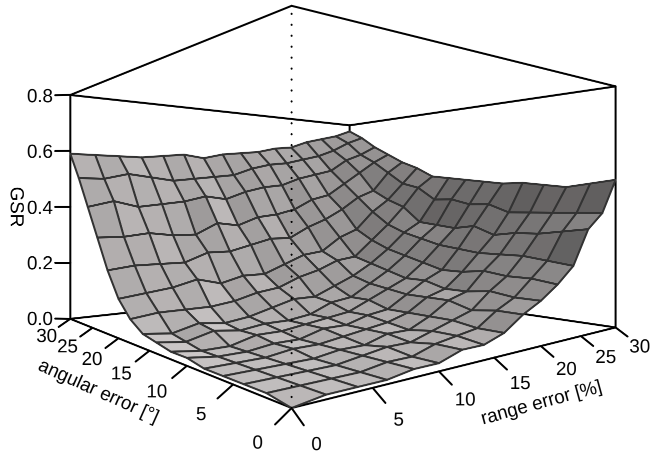
(b) u-shape



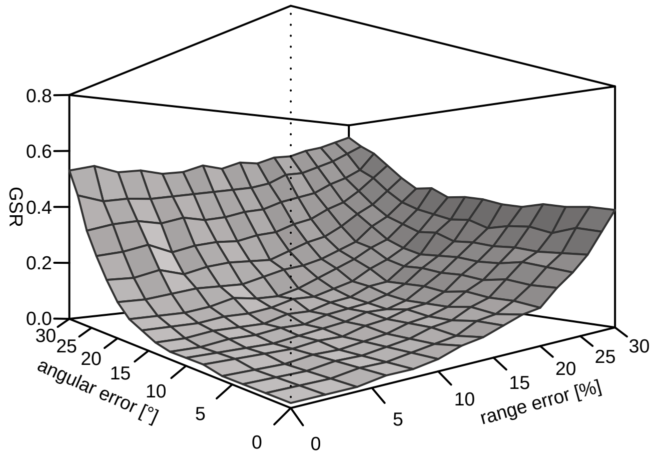
(c) star-shape



(d) grid-like

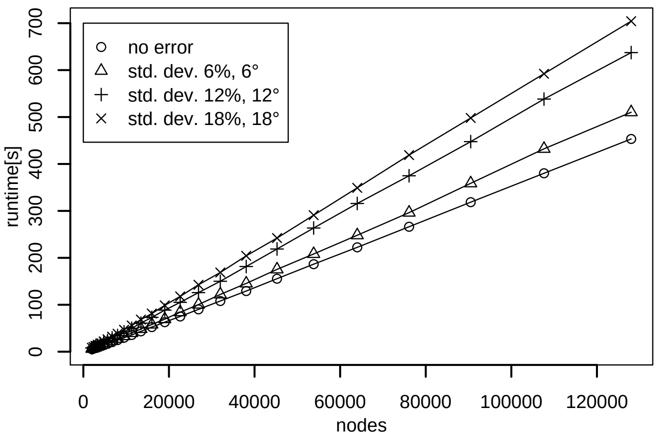


(e) donut

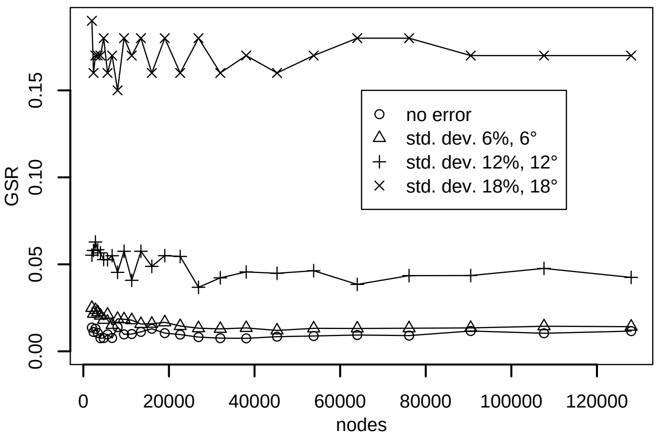


(f) irregular

Fig. 12. GSR norm on localizations for errors ranging from a standard deviation of 0% to 30% for distances and 0° to 30° for directions, respectively.



(a) Runtime



(b) GSR

Fig. 13. Runtime and localization quality for varying network sizes of up to 128000 nodes (centralized algorithm applied on u-shaped deployment area). Runtime is depicted for our Java-implementation running on one single core of a dual-core AMD Opteron(tm) processor 2218.

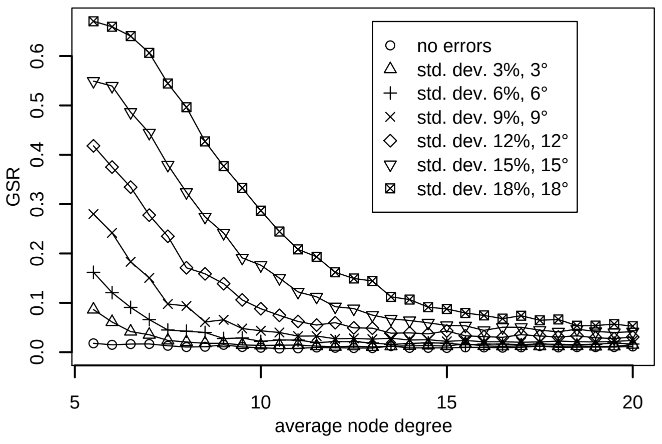


Fig. 14. Quality of reconstruction under varying average node degrees for 5000 nodes with different errors.

degree is depicted in Figure 14. It shows how a higher average node degree allows to compensate for more severe errors while for lower errors, a less connected graph would be sufficient.

During all our experiments[2], there was no subproblem, i. e., 3-hop neighborhood, on any level that contained more than 140 nodes – except for tests with high average degree, where 3-neighborhoods in the input graph became larger. On the average, subproblems contained around 60–70 nodes. For our implementation, we experienced that even for 128000 nodes, 10–12 levels were sufficient.

VI. Conclusion and Future Work

In this work, we addressed the problem of sensor network localization with both distance and *relative* direction measurements in the presence of errors. We have shown that the presence of arbitrarily small errors is enough to make this problem hard, generalizing the work of Basu et al. [21], waiving the assumption that nodes have global knowledge about their orientation.

We presented a novel, distributed algorithm scheme which combines robustness to fairly large errors with scalability and independence of the deployment region. The experimental evaluation yielded that this algorithm scheme recovered node positions very accurately for severe errors even with a quite naïve algorithm applied to the local subproblems. Here, next steps will be to combine the proposed scheme with more sophisticated algorithms to solve local problems, as it seems that positioning with large errors mostly fails due to the inability to reduce errors on early levels. Furthermore, different algorithms could make directional information dispensable, which is currently only needed for an initial layout of the force-directed layout algorithm.

References

[1] D. E. Culler, D. Estrin, and M. B. Srivastava, “Guest Editors’ Introduction: Overview of Sensor Networks,” *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[2] A. Savvides, M. B. Srivastava, L. Girod, and D. Estrin, “Localization in Sensor Networks,” *Wireless Sensor Networks*, pp. 327–349, 2004.

[3] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur, “A Theory of Network Localization,” *IEEE Transaction on Mobile Computing*, vol. 5, no. 12, pp. 1663–1678, December 2006.

[4] P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks*. Wiley, September 2005.

[5] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, “Routing with Guaranteed Delivery in Ad Hoc Wireless Networks,” *Wireless Networks*, vol. 7, no. 6, pp. 609–616, 2001.

[6] B. Karp and H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM’00)*, 2000, pp. 243–254.

[7] C. Savarese, J. Rabaey, and J. Beutel, “Locationing in Distributed Ad-Hoc Wireless Sensor Networks,” in *Proceedings of the 26th International Conference on Acoustics, Speech, and Signal Processing (ICASSP’01)*, 2001.

[8] A. Savvides, C.-C. Han, and M. B. Srivastava, “Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors,” in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM’01)*. ACM Press, 2001.

[2]Overall, we performed more than 160000 tests.

[9] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from Mere Connectivity," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'03)*. ACM Press, 2003.

[10] K. K. Chintalapudi, A. Dhariwal, and R. G. andGaurav Sukhatme, "Ad-Hoc Localization Using Ranging and Sectoring," in *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, 2004.

[11] X. Ji and H. Zha, "Sensor Positioning in Wireless Ad-hoc Sensor Networks Using Multidimensional Scaling," in *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, 2004.

[12] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust Distributed Network Localization with Noisy Range Measurements," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SENSYS'04)*. ACM Press, 2004, pp. 50–61.

[13] J. Bruck, J. Gao, and A. Jiang, "Localization and Routing in Sensor Networks by Local Angle Information," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'05)*. ACM Press, 2005, pp. 181–192.

[14] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-Free Distributed Localization in Sensor Networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SENSYS'03)*, 2003, pp. 340–341.

[15] C. Gotsman and Y. Koren, "Distributed Graph Layout for Sensor Networks," in *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)*. Springer-Verlag, 2004.

[16] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM'00)*, 2000, pp. 32–43.

[17] J. B. Saxe, "Embeddability of weighted graphs in k-space is strongly NP-hard." in *Proceedings of the 17th Allerton Conference in Communications, Control and Computing*, 1979, pp. 480–489.

[18] H. Breu and D. G. Kirkpatrick, "Unit Disk Graph Recognition is NP-Hard," *Computational Geometry: Theory and Applications*, vol. 9, no. 1–2, pp. 3–24, 1998.

[19] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Unit Disk Graph Approximation," in *Proceedings of the 8th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Dial-M'04)*, 2004.

[20] J. Aspnes, D. K. Goldenberg, and Y. R. Yang, "On the Computational Complexity of Sensor Network Localization," in *Proceedings of the 1st Intl. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'04)*. Springer-Verlag, 2004, pp. 32–44.

[21] A. Basu, J. Gao, J. S. B. Mitchell, and G. Sabhnani, "Distributed Localization Using Noisy Distance and Angle Information," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'06)*. ACM Press, 2006, pp. 262–273.

[22] M. Tubaishat and S. Madria, "Sensor Networks: An Overview," *IEEE Potentials*, vol. 22, no. 2, pp. 20–23, 2003.

[23] P. Biswas and Y. Ye, "Semidefinite Programming for Ad hoc Wireless Sensor Network Localization," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*. ACM Press, 2004, pp. 46–54.

[24] P. Gajer, M. Goodrich, and S. G. Kobourov, "A fast multi-dimensional algorithm for drawing large graphs," *Computational Geometry: Theory and Applications*, vol. 29, pp. 3–18, 2004.

[25] C. Erten, A. Efrat, D. Forrester, A. Iyer, and S. G. Kobourov, "Force-Directed Approaches to Sensor Localization," in *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX'06)*, 2006.