
Oh-Heum Kwon and Ha-Joo Song
Division of Electronics, Computer, and Telecommunication Engineering
Pukyong National University
akhirn@pknu.ac.kr

Abstract—In localization algorithms following the “patch-and-stitch” strategy, the network is divided into small overlapping subregions. For each subregion, the algorithm builds a local structure called a patch which is actually an embedding of the nodes it spans in a relative coordinate system. Then, the patches are stitched together to form a single global map. In this class of algorithms, the stitching order makes a great influence on the performance of the algorithm. In this paper, we present a formal framework to deal with stitching orders. In our framework, each stitching scheme consists of a stitching policy and a potential function. The potential function is to predict how well a patch will be stitched if patches are stitched according to a given partial order. The stitching policy is a mechanism that determines the stitching order based on the predictions by the potential function. We present various stitching schemes and evaluate their performances though simulations.

Index Terms—wireless sensor network, localization algorithm, distributed algorithms, patch and stitch

I. INTRODUCTION

Sensor networks consist of a large number of densely deployed sensor nodes which gather local data and communicate with each other. The sensed data are often meaningful only if we know where the data are from. Therefore knowing the positions of sensor nodes is essential in wireless sensor networks. A straightforward method to achieve node localization is using the Global Positioning System (GPS). The use of GPS is, however, a very expensive solution in terms of cost, size, and power consumption. More importantly, since GPS requires line-of-sight between the receiver and the satellites, it may not work well indoors, underground, or in the presence of obstructions.

These limitations of GPS have motivated the search for alternative methods that rely on the measurements of distances between nodes. The distances could be measured by distance estimates (ToA), time-difference of arrival (TDoA), or received signal strength (RSSI) [1]. The ToA and TDoA make use of signal propagation time for determining the distances. The RSSI converts the signal strength measurements to distance estimates.

A large number of localization algorithms have been proposed [1]. Some algorithms assume the presence of anchor nodes that know their exact positions in advance [8], [12]. This knowledge could be manually initialized or acquired through some additional hardware like GPS. Although the presence of anchor nodes can greatly simplify the task of localization, neither using GPS nor manual initialization is possible without cost.

Meanwhile, anchor-free localization algorithms do not need anchor nodes [5], [7], [11], [13], [14]. Since no coordinate system is given to the network, coordinates are defined in an arbitrary relative coordinate system and the resulting coordinate assignments have translation and orientation degrees of freedom.

The incremental algorithm is the simplest type of anchor-free localization algorithm [14]. The incremental algorithm usually selects a small set of seed nodes and assigns coordinates to them. Then, it repeatedly adds appropriate node to this set. This can be done if a node attempting to join the set can estimate distances to three or more nodes in the set. A drawback of this type of algorithm is that it is prone to error accumulation, resulting in poor overall coordinate assignments [11].

Priyantha et al [11] proposed a two-phase anchor-free localization algorithm called AFL. The first phase produces an initial coordinate assignment based on the connectivity between nodes. The second phase uses a mass-spring based optimization to correct localization errors. A problem of the AFL algorithm is that the first phase assumes that the nodes are deployed in a rectangle-shaped area. It produces poor initial coordinate assignments if the topology of the network is anisotropic, which can be caused by either the irregular shape of the area or by obstacles within the area.

Moore et al [7] proposed an algorithm called “robust quadrilateral”. A robust quadrilateral is a fully-connected set of four nodes satisfying certain conditions on the distances between them. Each node finds the robust quadrilaterals it belongs. Then two robust quadrilaterals are merged if they share three nodes. This merging process continues until all quadrilaterals are merged.

Shang et al. [13] proposed “patched MDS-MAP” algorithm. In this algorithm, each node builds a “local map” using MDS (Multidimensional Scaling) method. Each local map spans a node and all of its neighbors. Then the local maps are merged together to form a single global map. Ji et al. [5] also proposed basically similar algorithm using MDS method.

Although those algorithms mentioned above have their own characteristics, it is possible to understand them, except
the AFL algorithm, in a framework called “patch-and-stitch” paradigm: First, each node builds a local structure called a patch. Each patch is an embedding of the nodes it spans into a relative coordinate system. Next, the patches are stitched together to form a single global map.

Even the incremental algorithm can be understood as an extreme case of this paradigm in which each patch consists of a single node and the stitching is accomplished by iterative applications of multilateration.

In the cases where each patch spans multiple nodes, the absolute orientation method is commonly used for stitching patches [4]. This method is applicable if two patches share three or more common nodes. In two dimensional space, two patches can be combined without ambiguity if they share three common nodes.

In stitching patches, the most simple and common scenario is the incremental stitching [5], [13]: First, a seed map is constructed. Initially, the seed map is a mere patch that spans at least three nodes. Next, each patch is stitched into the seed map one by one. In other words, stitching is always done between the seed map and a patch. This continues until all patches are stitched to the seed map.

It is easy to understand why the incremental stitching is preferred in general. If we allow arbitrary pairs of patches to be stitched concurrently, we may be confronted by two large patches each of which spans almost a half of the entire network. Stitching these two will need an almost centralized computation, which may not be feasible for large scale sensor networks.

In the incremental stitching, patches are stitched to the seed map according to a stitching scheme. We mean by the stitching scheme the rules that tell under what conditions a patch is allowed to be stitched into the seed map. Consequently, this scheme determines the (partial) order by which the patches are stitched. According to our simulations, this order makes a great influence on the performance of the localization algorithm. As far as we know, no previous work fully addressed this issue.

In this paper, we present a formal framework of stitching schemes which covers a variety of intuitive heuristics and evaluate several representative ones by simulations. In our formal framework, each scheme consists of two parts: a potential function and a stitching policy. The potential function predicts how well a patch will be positioned (i.e., stitched) if patches are stitched by a given partial order. The stitching policy is a mechanism that determines the stitching order based on the given potential function. Especially, we examine two categories of the stitching policies:

1) Single-Pass Greedy Policy: At each step, each patch is evaluated by the potential function. Then the best patch is selected and stitched to the seed map.

2) Two-Pass Policy: The stitching schedule is pre-computed in a preprocessing step. Then the patches are stitched into the seed map according to the pre-computed schedule.

This paper is organized as follows. In Section 2, we introduce some notations and describe some common tools which are widely used in localization algorithms. In Section 3, we present the formal framework of stitching scheme and propose several potential functions and stitching policies. Section 4 gives the results of our experimental analysis and Section 5 gives our concluding remarks.

II. PRELIMINARIES

A sensor network can be represented as a graph $G = (V, E)$ where each node $v \in V$ represents a sensor node and each edge $(u, v) \in E$ represents the fact that node $u$ and $v$ can communicate directly. Associated with each edge $(u, v)$ is a distance estimate $d(u, v)$ between two. Let $N(v)$ be the set of neighbors of $v$ in $G$ and let $N^+(v) = N(v) \cup \{v\}$.

We assume that each patch is constructed and owned by a node. The patch $P_v$, which is constructed and owned by node $v$, is indeed a tuple $(M_v, p_v)$ where $M_v \subseteq N^+(v)$ is the set of members of $P_v$ and $p_v : M_v \rightarrow \mathbb{R}^2$ is an assignment of coordinates to the members in a relative coordinate system.

In this paper, we concern ourselves with three types of patches: singleton, partial 1-hop, and full 1-hop patches. A singleton patch is just a node. The incremental algorithm uses patches of singleton type. A partial 1-hop patch consists of an owner node and a subset of neighbors of the owner. Robust quadrilaterals are examples of partial 1-hop patches. If we construct patch $P_v$ by iterative applications of multilateration within $N^+(v)$, then the patch will fall into this type. A full 1-hop patch always consists of an owner and all neighbors of the owner. The patched-MDS algorithm uses patches of this type.

Let us briefly explain the common tools which are often used in the patch-and-stitch localization algorithms. The multilateration is the method that determines coordinate of a node, given coordinates of three or more neighbors and distances from them. Let $x_i, i = 1, \ldots, k, k \geq 3$, be the coordinates of neighbors of a node $v$ and let $d_i, i = 1, \ldots, k$, be the estimated distances from the neighbors. Then the coordinate $x \in \mathbb{R}^2$ of $v$ is determined so that it minimizes the following error function:

$$\sum_{i=1}^{k} ((|x_i - x| - d_i)^2$$

The MDS is a technique widely used for analyzing dissimilarity of data and discovering the spatial structure in the data [2]. The input to the MDS is a distance matrix which specifies distances between every pair of nodes and the output is the set of coordinates that approximately coincide with the distance matrix. Applying MDS method to localization problem, we often need to specify the distances between non-adjacent nodes. We follow the approach taken in earlier work [5], [13] in which the shortest path lengths are used in places of the distances between non-adjacent nodes.

The absolute orientation method is the most commonly used stitching technique. Suppose that two patches have three or more nodes in common. Stitching two patches is achieved by translating, rotating, and/or reflecting one patch so that the coordinates of the common nodes in both patches coincide as closely as possible, as shown in Figure 1. This transformation can be formulated as a mathematical optimization problem that can be solved in linear time [4].
coordinate system. To construct the seed map, we select a
embedding of three or more nodes in an arbitrary relative
with the construction of the seed map. The seed map is an
multiple times. For the simplicity, our algorithm takes the first
algorithm usually assigns coordinate in the seed map to a node
a node often belongs to multiple patches. Therefore, this
must notify these coordinates to the members through either
the patch. Beside the case of singleton patches, the owner
collected sufficient information to apply the stitching technique
and also met the additional conditions enforced by the stitching
were stitched too early. In this sense, one purpose of all
stitching schemes is to postpone the stitching to an appropriate
point of time.

One principle underlies all schemes we present in this
paper. We can see the stitching process as successive decision
makings with uncertainties. At each step, the coordinates of
nodes in the seed map play the role of premises upon which
a new decision (that is, determining the coordinates of nodes
of a new patch) is made. The patches waiting to be stitched
can be considered as questions waiting to be answered. If one
of them is chosen and answered, the answer becomes a part
of the premises for the next decisions. In this scenario, it is
very natural and intuitive to begin with answering the question
of which the answer is expected to be the most reliable. By
doing so, we can keep the premises as reliable as possible.

To describe the stitching schemes formally, we need some
notations. For any two patches $P_i$ and $P_j$, let $co(P_i, P_j)$,
called the correlation of two, be the set of available stitching
information helpful in stitching $P_i$ into $P_j$. The exact definition
of the correlation depends on the stitching technique. For
instances, it can be the coordinates of the common nodes if the
absolute orientation method is used, or it can be the distances
of the edges connecting two patches if $P_i$ is a singleton patch
and the multilateration is used for stitching. We use the same
notation to represent the correlation between a patch and the
seed map since the seed map is a mere large patch.

A stitching order can be abstracted as a rooted DAG
(Directed Acyclic Graph) $D = (s, V, A)$ where the root $s \in V$
is the seed node. It is a rooted DAG in the sense that $s$
is an ancestor of all other nodes. An edge $(i, j)$ is in $A$
if (1) patch $P_i$ is stitched into the seed map prior to patch
$P_j$, (2) $co(P_i, P_j) \neq \emptyset$, and (3) the correlation
$co(P_i, P_j)$ is actually used in stitching $P_i$ into the seed map.
The condition (3) leaves the room for stitching algorithm to ignore some
stitching information even if it is available.

We can regard the stitching process as a process of finding a
DAG and our interest is finding a good DAG. So we are going
to establish a model which predicts how well a patch $P_o$ will
be positioned (i.e., stitched) if patches are stitched according to
the order specified by a given DAG $D$. We assume it depends
on three factors:

1) It depends on how well the patches owned by the
predecessors of $v$ in the DAG $D$ are positioned.
2) It depends on the quality and the quantity of the corre-
lation between the predecessors’s patches and the target
patch $P_o$. 

III. STITCHING SCHEMES

A. Outline of Localization Algorithm

Once all patches are constructed, the stitching phase begins
with the construction of the seed map. The seed map is an
embedding of three or more nodes in an arbitrary relative
coordinate system. To construct the seed map, we select a
seed node. If the patch owned by the seed node is as large as we
need, we just take the patch as the seed map. If not, we have to
construct the seed map separately. This can be done by letting
the seed node to gather local distance information and to
construct a seed map via some techniques like multilateration
or MDS.

The selection of seed node is important. In practice, we
may have to choose the seed node carefully. The choice can
be done based on some criterion like node degrees, distances
to the farthest node, and so on. But, in our simulations, we
just pick a random node because our purpose is to compare
stitching schemes.

The stitching phase proceeds as follows. Each node belong
to the seed map broadcasts a message to its neighbors which
tells the id and the coordinate of the sender in the seed map.
Each owner of a patch collects stitching information from
these messages. It depends on the stitching technique to be
used what the stitching information means. If the owner has
collected sufficient information to apply the stitching technique
and also met the additional conditions enforced by the stitching
scheme, then it performs stitching of its patch into the seed
map. This stitching assigns coordinates to the members of
the patch. Beside the case of singleton patches, the owner
must notify these coordinates to the members through either
a broadcast message or successive unicast messages. Upon
receipt of these messages, the recipients recognize themselves
as members of the seed map and repeat this procedure.

When either full 1-hop or partial 1-hop patches are used,
a node often belongs to multiple patches. Therefore, this
algorithm usually assigns coordinate in the seed map to a node
multiple times. For the simplicity, our algorithm takes the first
one and just ignores all the late-comers.

B. Stitching Policies

For each stitching technique, there exist necessary condi-
tions for the technique to be applied. We call these conditions
the triggering conditions. For examples, the multilateration
method requires for the singleton-patch to have at least three
neighbors in the seed map, while the absolute orientation
method requires at least three common nodes with the seed
map. We call it the immediate stitching scheme if each patch is
allowed to be stitched as soon as the triggering conditions have
met. This scheme is simple but undesirable since patches will
tend to be stitched too early. In this sense, one purpose of all
stitching schemes is to postpone the stitching to an appropriate
point of time.

Fig. 1. Stitching two patches using the absolute orientation method
3) It also depends on the quality (i.e., preciseness) of the target patch $P_v$.

If we quantify those factors by some means, we are able to represent the prediction by a recursive function $f_D$ of the following form:

$$f_D(v) = g(P_D(v), C_D(v), Q(v))$$

where $P_D(v) = \{f_D(u_1), f_D(u_2), \ldots, f_D(u_k)\}$,

$$C_D(v) = \{co(v, u_1), co(v, u_2), \ldots, co(v, u_k)\},$$

$$Q(v) = \text{the quality of the target patch } P_v,$$

where $u_1, u_2, \ldots, u_k$ are the predecessors of $v$ in $D$. We will call this function $f_D$ the potential function. This is the description of the potential function in its most general setting. Each potential function we actually examine in this paper takes into account only a subset of the factors listed. Usually, it is difficult to judge how accurate the correlation information is. For an example, judging the accuracies of distance estimates can also be applied to the incremental algorithm in which all patches are singleton and the multilateration is used for stitching technique. In this case, we interpret $co(v, u)$ the set of a single edge connecting $v$ and $u$.

Meanwhile, the MAXMIN strategy proposed in [10] uses the following potential function.

$$f_D^{\text{mm}}(v) = \begin{cases} \infty, & \text{if } v = s; \\ \min \{\min_i f_D^{\text{mm}}(u_i), f_D^{\text{co}}(v)\}, & \text{otherwise.} \end{cases}$$

Compared to $f_D^{\text{mm}}$, this function takes into account the predecessors’s correlations recursively. This function insists that, even if a patch has a large number of common nodes with the seed map, we could not expect a good result if its predecessors had been stitched based on few common nodes with the seed map.

As an another example, it is a very natural idea to stitch patches according to the increasing order of their hop-distances from the seed node. This heuristic is related to the following potential function:

$$f_D^{\text{hop}}(v) = \max_{i=1,\ldots,k} f_D^{\text{hop}}(u_i) + 1$$

Precisely speaking, the heuristic performs stitchings according to the DAG $D$ that minimizes $f_D^{\text{hop}}(v)$ for every node $v$.

It is possible to combine multiple functions. One way to combine functions is to multivariate the potential function. For an example,

$$f_D^{\text{hm}}(v) = (f_D^{\text{hop}}(v), f_D^{\text{co}}(v)).$$

By assuming lexicographic ordering of the function values, we will have a kind of hop-distance-major stitching order.

The functions listed so far do not count on the quality $Q(v)$ of the target patch. It is possible to incorporate it by assuming $0 \leq Q(v) \leq 1$ and multiplying it to the functions mentioned so far. The quality $Q(v)$ can be appropriately evaluated by comparing measured distances and the computed distances between members of patches. Of course, they will be no longer semi-topological functions. Actually, we have performed simulations to study the behaviors of those functions. Although they achieve consistent performance improvements, the differences are not significant. So, we will not mention this factor $Q(v)$ in this version of our paper anymore.

In this paper, we consider several potential functions. For some of them, the smaller function value means the better potential, while, for others, the larger the better. This makes our descriptions of the policies somewhat tricky. So we use the notation $f_D(u) > f_D(v)$ (or $f_D(u) \geq f_D(v)$) to indicate that the potential function value $f_D(u)$ is better than (or equal to) $f_D(v)$.

D. Single-Pass Greedy Policy:

Given an exact definition of the potential function, this policy tries to find the patch with the best potential at each step of the stitching phase and gives it the right to be stitched. We call this SPG-G (Single-Pass Greedy-Global) policy.

Precisely speaking, suppose that patches $\{P_1, \ldots, P_t\}$ have already been stitched to the seed map. Let $D'$ be the partial DAG representing the order by which those patches are
stitched and let $D' + v$ denote the DAG which is obtained by adding node $v$ and edges $(u, v)$ for each node $u \in \{1, \ldots, i\}$ with $\text{co}(v, u) \neq \emptyset$ into $D'$. The SPG-G policy selects the patch $v \notin \{1, \ldots, i\}$ which maximizes the potential function $f_{D'}(v)$.

We think this is a very powerful method, but doubtful if it can be implemented in a distributed way with reasonable overheads. A straightforward implementation will require an execution of the distributed leader election protocol at each step of the stitching phase, which needs $O(N \log N)$ messages [3] and, consequently, require $O(N^2 \log N)$ overall message complexity.

A compromise can be achieved by restricting the scope of comparison, that is, selecting the local best patches rather than the global best patch. A patch is selected if no neighboring patch has better potential function value than itself. This lead us the local leader election problem which can be solved by letting each node broadcast a message delivering its potential value whenever its potential value is updated. With potential functions $f^\text{mm}$ or $f^\text{hop}$, each node will broadcast the message at most $O(N)$ times. Therefore, the total number of messages is $O(N^2)$. We call this SPG-L (Single-Pass Greedy-Local) policy.

E. Two-Pass Policy:

This approach separates the construction of the DAG from the actual stitching process. In the preprocessing phase, we construct a DAG in a distributed way without actually performing stitching. In the second phase, patches are stitched into the seed map according to the order specified by the DAG, that is, a patch is stitched only if all its predecessors in the DAG have already been stitched.

**Definition 3.2:** Given a semi-topological potential function $f_D$, a set of patch memberships, and an initial seed map, we say that a DAG $D$ is optimal if, for any node $v$, $f_D(v) \geq f_{D'}(v)$ for any other DAG $D'$.

This definition of optimality is the strongest in the sense that if this optimality is achieved, it also optimizes the other typical forms of optimality like the max-sum, the max-min, and so on. But one problem is that it is not always possible to achieve this optimality by a DAG. It depends on the potential function whether there exists an optimal DAG or not.

**Definition 3.3:** We say that a semi-topological potential function $f_D$ is monotone if $f_D(u) \geq f_D(v)$ for any predecessor $u$ of $v$ in $D$.

To illustrate the meaning of the monotone potential function, we again count on the metaphor of successive decision makings. Definition 3.3 says that no conclusion can be more reliable than the proposition that is a part of the premises. Of course, it is just a hypothesis, not a fact. But it is also true that some reasonable potential functions fall into this category. For examples, functions $f^\text{hop}$, $f^\text{mm}$, and $f^\text{hm}$ are all monotone.

In this paper, we study the performance of the two-pass policy under several monotone potential functions including $f^\text{hop}$, $f^\text{mm}$, and $f^\text{hm}$. Figure 2 illustrates the differences between the hop-distance-related potential functions. In Figure 2, $\{v_1, \ldots, v_3\}$ are nodes with hop-distance one and $\{v_4, \ldots, v_6\}$ are nodes with hop-distance two from the seed node $s$. Figure 2 (a) is an optimal DAG with respect to potential function $f^\text{hop}$. Indeed, this DAG fails to come up with our hope with the hop-distance ordering. A node is allowed to stitch its patch as soon as one of its neighbors with the shorter hop-distance has been stitched.

Figure 2 (b) is optimal with respect to $f^\text{hm}(v)$. Since function $f^\text{hm}(v)$ aims to maximize the correlation with the seed map while preserving $f^\text{hop}$ the minimum, it has an effect of adding as many precedence constraints between nodes of adjacent levels in DAG as possible. Therefore, a node is allowed to stitch its patch only if all its neighbors with the shorter hop-distance have been stitched.

What potential function do we need if we want to add precedence constraints between nodes of the same level, as shown in Figure 2 (c)? In Figure 2 (c), nodes at the same level are also partially ordered so that their correlations with the seed map are maximized while preserving the acyclic property of the DAG. The following potential function is obviously monotone and achieves what we want.

$$f^\text{hm2}(v) = \begin{cases} f^\text{hop}(u_1) + 1, f^\text{hm}(v) \\ \max_{k=1}^{r} f^\text{hop}(u_i), f^\text{hm}(v) \end{cases} \quad \text{otherwise.}$$

An optimal DAG can be computed using a variant of the Bellman-Ford algorithm which is originally for finding shortest paths in a distributed way. In this algorithm, each node $v$ maintains its potential $f(v)$ and the set of predecessors $\text{pred}(v)$. Initially, $f(v)$ is set to $\infty$ if $v$ is the seed node; zero
otherwise. The set \( \text{pred}(v) \) is set to an empty set. Seed node broadcasts an initial message to its neighbors. This message contains the id and the potential of the seed node.

Upon receiving a message \( (u, f(u)) \) with \( f(u) \succ f(v) \), node \( v \) tries to update its potential \( f(v) \) based on the available potential values of its neighbors and also the correlation information with them. If an update occurs, node \( v \) announces its new potential to its neighbors. This process is repeated until no more update takes place.

### Algorithm 1 Constructing DAG (Node \( u \))

1. **if** this node \( u \) is the seed node **then**
2. \( f(u) \leftarrow \infty; \)
3. \( \text{message} \leftarrow (u, f(u)); \)
4. **BroadcastToNeighbors**(message);
5. **else**
6. \( f(u) \leftarrow 0; \)
7. **end if**
8. \( \text{pred}(u) \leftarrow \emptyset; \)
9. **loop**
10. \( \text{message} = \text{ListenToMessage}(); \)
11. // Exit the procedure if no message arrives
12. // for a specified period of time
13. **if** \( f(v) \succ f(u) \) **then**
14. \( \text{updated} = \text{UpdatePotential}(v, f(v)); \)
15. **if** updated **then**
16. \( \text{message} \leftarrow (u, f(u)); \)
17. **BroadcastToNeighbors**(message);
18. **end if**
19. **end loop**

Lemma 3.4: Algorithm 1 produces an optimal DAG if semi-topological function \( f_D \) is monotone.

**Proof:** Since the potential function is monotone, it is obvious that Algorithm 1 produces a DAG. Let \( D \) denote the DAG produced by this algorithm. Suppose, as a contradiction, there exists another DAG \( D' \) with \( f_D(v') \succ f_D(v) \) for a node \( v \). Let us call node \( v \) the current node. We trace upward DAG \( D' \) starting at node \( v \) by repeating the following move: If there exists any predecessor \( u \) of the current node in \( D' \) with \( f_{D'}(u) \succ f_D(u) \), then \( u \) becomes the current node. This movement will end at a node \( v' \) such that \( f_{D'}(v') \succ f_D(v') \) and \( f_{D'}(u) \preceq f_D(u) \) for any predecessor \( u \) of \( v' \) in \( D' \). Since \( f_D(v') \prec f_{D'}(v') \) and \( f_{D'}(v') \preceq f_D(u) \) and \( f_D(u) \preceq f_{D'}(u) \), we have \( f_D(v') \prec f_D(u) \) for any predecessor \( u \) of \( v' \) in \( D' \), which implies that \( u \) is not a descendant of \( v' \) in \( D \). Now that each predecessor \( u \) of \( v' \) in \( D' \) has no reasons for being disqualified to be a predecessor of \( v' \) in \( D \) and also \( f_{D'}(u) \preceq f_D(u) \), there is no reason for being \( f_{D'}(v') \succ f_D(v') \), which is a contradiction. 

Consider the message complexity of Algorithm 1. Each node broadcasts a message whenever its potential is updated. Therefore, the number of messages depends on the granularity of the potential value. With either \( f^{\text{hop}}_D \) or \( f^{\text{mm}}_D \), the potential value is an integer between 0 and \( N \). So each node sends at most \( N \) messages; the total number of messages is at most \( N^2 \). The function \( f^{\text{mm}}_D \) and \( f^{\text{mm}}_D \) has \( N^2 \) distinct function values. However, we can reduce the message complexity by computing and fixing the first component \( f^{\text{hop}}_D \) first and then computing the second component. By doing so, it suffices to send at most \( O(N) \) messages per node. Table I summarizes the message complexities of the stitching schemes. The messages required for constructing patches and also for executing localization algorithm described in Section III-A are not included.

Finally, there is a significant benefit of two-pass policy. If the potential function \( f_D \) is pure-topological, the Algorithm 1 can be executed before constructing the patches. Indeed, all functions mentioned so far are pure-topological if either full 1-hop or singleton patches are used. This fact has some importance for the case where full 1-hop patched are used. If a node has no successor in the DAG, then the node does not need to perform stitching and even does not need to construct a patch. According to our simulations, about 40% ~ 60% of nodes are terminal nodes in the DAG and, therefore, do not need to construct patches.

### IV. Experimental Results

To study the behavior of the stitching schemes, we use an example network configuration shown in Figure 3. We regulate the average node degree by varying the number of nodes enclosed in the area. The number of nodes is between 150 and 400. We assume that every node has the same communication range \( r \) and two nodes succeed to communicate with probability \( p = 1 - \frac{d}{r^2} \) if \( d \leq r \) where \( d \) is the distance between them. Although this assumption may not be a realistic description in physical environments, it is a simple and valid starting point for simulation purposes. We also consider the

![Fig. 3. Example sensor network configuration (260 nodes).](image-url)
errors of distance measurements. We perform simulations with the maximum measurement error from 5% to 40% of the actual distances. The distance error is taken independently upon each edge and uniformly random over the offset limit.

We compare the positioning capabilities of the algorithms. Since anchor-free localization algorithms produce coordinates with translational and rotational degrees of freedom, it is meaningless to compare the actual and the computed coordinates directly. So we first transform the computed coordinates so that they are defined in the same coordinate system. This transformation is exactly what the absolute orientation method does. In other word, we translate, rotate, and/or reflect the computed coordinates so that the sum of the squared distances between the actual coordinates and the transformed computed coordinates is minimized. And then, we measure the performance with the mean normalized distances between the actual coordinates and the transformed coordinates:

\[
\text{error} = \frac{1}{N'} \sum_{v} \frac{||\text{crd}_{\text{actual}}(v) - \text{crd}_{\text{comp}}(v)||}{r},
\]

where \(\text{crd}_{\text{actual}}(v)\) is the actual coordinate, \(\text{crd}_{\text{comp}}(v)\) is the transformed computed coordinate of node \(v\), and \(N'\) is the total number of localized nodes. The summation is done for all localized nodes. We mean by the localized nodes the nodes in the final seed map.

In this paper, we simulate several stitching schemes. The schemes are named according to the following form:

[type of patches | patch construction technique | stitching technique | stitching policy | potential function]

Although a lot of schemes can be made, we focus on just two classes of schemes: [full 1-hop | MDS | absolute orientation [*] [*] and [singleton | – | multilateration [*] [*].

Figure 4 shows the percentages of the localized nodes. For full 1-hop patches, almost all nodes are localized if the average node degree is about 8 or more. On the other hand, For singleton patches, the average node degree must be about 16 or more for the percentage to be close to 100%.

Figure 5 shows the localization errors with varying the average node degrees. The maximum distance measurement error is fixed to 10% of the actual distances. Figure 5 (a) shows the results when full 1-hop patches are used. Overall, there exists a huge performance gap between the immediate stitching policy and the others. The SPG-G policy performs best if it is associated with either \(f_{D}^{nor}\) or \(f_{D}^{co}\). Following them are the two-pass policies with potential functions \(f_{D}^{mm}\) and \(f_{D}^{mm2}\). The SPG-L with potential function \(f_{D}^{nor}\) is fairly comparable to the two-pass policies.

Figure 5 (b) shows the results when singleton patches are used. In this case, the SGP-G with \(f_{D}^{co}\) performs best. Following it are the two-pass policy with \(f_{D}^{mm}\), the SPG-G with \(f_{D}^{nor}\), and the SPG-L with \(f_{D}^{co}\). It is remarkable that the stitching order purely based on the hop-distances from the seed node is far worse than expected. Some curves in Figure 5 (b) are increasing as the average node degree increases. That is because the percentage of localized nodes is also increasing as the average node degree increases. Figure 6 shows the localization errors as the distance measurement error bound varies from 5% to 40%. The average node degree is fixed to about 17.

### Table I

<table>
<thead>
<tr>
<th>Type of Patches</th>
<th>Stitching Scheme</th>
<th>Potential Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Immediate</td>
<td>no message</td>
</tr>
<tr>
<td>Full 1-hop</td>
<td>SPG-L</td>
<td>(O(N^2))</td>
</tr>
<tr>
<td></td>
<td>SPG-G</td>
<td>(O(N^2 \log N))</td>
</tr>
<tr>
<td></td>
<td>Two-Pass</td>
<td>(O(N^2))</td>
</tr>
<tr>
<td>Singleton</td>
<td>Immediate</td>
<td>no message</td>
</tr>
<tr>
<td></td>
<td>SPG-L</td>
<td>(O(N^2))</td>
</tr>
<tr>
<td></td>
<td>SPG-G</td>
<td>(O(N^2 \log N))</td>
</tr>
<tr>
<td></td>
<td>Two-Pass</td>
<td>(O(N^2))</td>
</tr>
</tbody>
</table>

*Fig. 4. Percentages of localized nodes*
V. CONCLUDING REMARKS

In this paper, we present a formal framework for deciding stitching orders in patch-and-stitch localization algorithms which consists of a potential function and a stitching policy. We propose several reasonable examples of stitching schemes and evaluate their performances through simulations. We have got several conclusions. First, although SPG-G policy always performs best, the simulation results show that the two-pass policy and SPG-L can be more viable solutions if they are associated with appropriate potential functions. Second, the immediate stitching policy is almost not worthy of consideration. Third, in general, the hop-distances from the seed map is less important than the amount of correlations.

REFERENCES

Fig. 6. Localization errors with varying distance measurement error rate