Localization of networks with presence and distance constraints based on 1-hop and 2-hop mass–spring optimization

Gabriele Oliva\textsuperscript{a,}\textsuperscript{*}, Roberto Setola\textsuperscript{a}, Stefano Panzieri\textsuperscript{b}, Federica Pascucci\textsuperscript{b}

\textsuperscript{a} University Campus Bio-Medico of Rome, Unit of Automatic, Via A. del Portillo 21, 00128 Rome, Italy
\textsuperscript{b} Dipartimento di Ingegneria, University Roma TRE, Via della Vasca Navale, 79, 00146, Roma, Italy

Received 30 September 2015; received in revised form 14 January 2016; accepted 5 February 2016

Available online 23 February 2016

Abstract

In this paper we consider the localization of a sensor network where the nodes are heterogeneous, in that some of them are able to measure the distance from their neighbors, while some others are just able to detect their presence, and we provide a post-processing algorithm that can be used to improve an initial estimate for the location of the nodes, based on a mass–spring optimization approach, taking into account presence and distance information, as well as one-hop and two-hop information.

© 2016 The Korean Institute of Communications Information Sciences. Production and Hosting by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Keywords: Wireless sensor networks; Range-free localization; Trilateration; Mass–spring optimization

1. Introduction

The localization problem in sensor networks is usually addressed by considering nodes able to compute inter-distances (see for instance [1–3]), or sensors that are able to detect the presence of nodes in the neighborhood (e.g., [4]). In [5] we propose a different perspective, by considering hybrid networks, i.e., networks composed of both types of nodes. When the available information is affected by noise, however, the estimated position for the nodes might be unsatisfactory, and there is a need to provide adequate post-processing algorithms to reduce the position error. Among the others, the mass–spring optimization algorithm [6] shows good results in terms of error reduction and complexity.

In this paper, we extend the mass–spring optimization algorithm to hybrid networks, in order to handle both distance and presence information. Assuming the network is a unit disk graph (i.e., that a pair of nodes communicate when their distance is less than a given communication radius), moreover, we further improve the algorithm by taking into account also negative information on the fact that 2-hop neighbors (i.e., nodes that are not neighbors, but have a neighbor in common) are not connected. A simulation campaign which shows the benefits of the proposed approach concludes the paper.

The outline of the paper is as follows: in Section 2 we present the problem setting, while in Section 3 we review the mass–spring optimization algorithm; in Section 4 we develop an extension of the mass–spring optimization algorithm that takes into account also presence information and 2-hop information, while in Section 5 we present our simulations; some conclusive remarks are collected in Section 6.

2. Problem setting

Let us consider a hybrid sensor network, where some nodes, namely presence nodes, are able to measure just the presence of their neighbors, while some other nodes, namely distance nodes, are also able to measure the distance from their neighbors. We assume the distance nodes are able to transmit their measured distances to their presence neighbors, so a distance information is available for two sensors $i$ and $j$ provided that at least one of them is a distance node. The hybrid sensor network can be represented by a graph $G = (\mathcal{V}, \mathcal{E}_d \cup \mathcal{E}_p)$ with $|\mathcal{V}| = n$ nodes. The edges $\mathcal{E}_d$ and $\mathcal{E}_p$ represent distance and presence.
A distance constraint is a constraint in the form \( \| p_i - p_j \| = d_{ij} \leq \rho \), while a presence constraint is a constraint in the form \( \| p_i - p_j \| \leq \rho \), where \( \rho \) is the communication radius and we assume that \( \rho \) is the same for all the agents. We assume that the graph \( G \) is a unit disk graph, i.e., a graph such that there is a link between two nodes \( v_i \) and \( v_j \) provided that \( \| p_i - p_j \| \leq \rho \). The above assumption implies that we can use also negative information to get rid, to some extent, of position ambiguity. For instance, suppose that a localized node is not in reach of a non localized node; we conclude that the circle of radius \( \rho \) centered at the localized node does not contain the node to be localized. We assume the measured distances \( d_{ij} \) are affected by noise, and we assume we already have an estimate for the position of the nodes. In particular, we assume the nodes have calculated their position resorting to the approximated algorithms provided in [5]. We want to provide a mechanism to improve the accuracy of the localization of the nodes in the sensor network.

3. Mass–Spring optimization

**Algorithm 1: Mass–Spring Optimization**

\[
\begin{align*}
& t \leftarrow 0; \\
& e(0) \leftarrow \infty; \\
& \hat{p}_i \leftarrow \text{initial estimate for } i = 1, \ldots, n; \\
& \hat{p}_i^* \leftarrow \hat{p}_i \text{ for } i = 1, \ldots, n; \\
& \text{exit-condition} \leftarrow 0; \\
& \text{while not exit-condition do} \\
& \quad \text{calculate } \vec{F}_i(t), \forall i = 1, \ldots, n; \\
& \quad \hat{p}_i^* \leftarrow \hat{p}_i^* + \frac{\vec{F}_i(t)}{2|N_i|}, \forall i = 1, \ldots, n; \\
& \quad \text{calculate } e_i(t), \forall i = 1, \ldots, n; \\
& \quad \text{calculate } e(t); \\
& \quad \text{if } e(t) < e(t-1) \text{ then} \\
& \quad \quad \hat{p}_i \leftarrow \hat{p}_i^*, \forall i = 1, \ldots, n; \\
& \quad \text{end if} \\
& \quad \text{if } e(t) < \eta \text{ then} \\
& \quad \quad \text{exit-condition} = 1; \\
& \quad \text{end if} \\
& t \leftarrow t + 1; \\
& \text{end while} \\
& \text{return } \hat{p}_i \text{ for } i = 1, \ldots, n;
\end{align*}
\]

In this section we briefly review the Mass–Spring Optimization technique [6], while we extend the framework in order to handle presence constraints and 2-hop information in the next section. The procedure described below is summarized in Algorithm 1. In [6] a mass–spring optimization algorithm is used to refine an initial estimate for the position of the nodes, assuming just distance constraints are available (i.e., \( E_p = \emptyset \)). Specifically, each link is treated as a spring whose natural length is the noisy measured distance \( \hat{d}_{ij}(0) = \hat{d}_{ij} + \delta_{ij} \) and the nodes \( v_i \) and \( v_j \) are initially estimated to be in the positions \( \hat{p}_i(0) \) and \( \hat{p}_j(0) \) that are the result of a localization procedure. The algorithm simulates a framework of springs and aims at reducing the energy associated to each node, in order to get close to a zero energy state, although in practice a local minimum is likely to be found [6]. Let \( \vec{w}_{ij}(t) \) be the unit vector in the direction from \( \hat{p}_i(t) \) to \( \hat{p}_j(t) \), at time instant \( t \). The force exerted by the single spring is given by

\[
\vec{F}_{ij}(t) = \vec{w}_{ij}(t)(\hat{d}_{ij}(t) - \hat{d}_{ij}(0))
\]

where \( \hat{d}_{ij}(t) \) is the distance calculated at step \( t \) as a result of the choice of \( \hat{p}_i(t) \) and \( \hat{p}_j(t) \).

The overall force for node \( i \) is given by

\[
\vec{F}_i(t) = \sum_{j=1}^{n} \vec{F}_{ij}(t)
\]

while the energy for node \( i \) is given by

\[
e_i(t) = \sum_{j=1}^{n} e_{ij}(t) = \sum_{j=1}^{n} (\hat{d}_{ij}(t) - \hat{d}_{ij}(0))^2.
\]

The total energy is calculated as

\[
e(t) = \sum_{i=1}^{n} e_i(t).
\]

At each step, the framework of springs is simulated in that each node moves its estimated position along the direction of \( \vec{F}_i(t) \); such movement has a magnitude \(|\vec{F}_i(t)|/(2|N_i|)\), along the direction of the force \( \vec{F}_i(t) \), where \( N_i \) is the number of 1-hop neihbors of node \( v_i \), i.e., the number of nodes that are connected to \( v_i \) in the graph \( G \). The above choice of the magnitude has been selected empirically in [6]. Notice that the movement is done if and only if the total energy is reduced.

The mass–spring algorithm amounts to a repetition of the above procedure, which is iterated until \( e(t) < \eta \), for a given threshold \( \eta \).

4. Mass–Spring optimization with presence and 2-hop information

4.1. Adding presence 1-hop information

Let us suppose an initial estimate \( \hat{p}_i(0) \) for the position of each node \( v_i \) in an hybrid sensor network \( \Sigma \) is available. Differently from the standard Mass–Spring Optimization approach, we need to develop a mechanism to use the presence information.

For any two nodes \( v_i, v_j \) such that \( (v_i, v_j) \in E \), we choose

\[
\vec{F}_{ij}^{one}(t) = \vec{w}_{ij}(t)F_{ij}^{one}(t)
\]

where

\[
F_{ij}^{one}(t) = \begin{cases} 
0, & \text{if } (v_i, v_j) \in E_p \text{ and } \hat{d}_{ij}(t) \leq \rho \\
\hat{d}_{ij}(t) - \rho, & \text{if } (v_i, v_j) \in E_p \text{ and } \hat{d}_{ij}(t) > \rho \\
\hat{d}_{ij}(t) - \hat{d}_{ij}(0), & \text{else}
\end{cases}
\]

The above choice implies that the spring behaves as a regular spring, unless the link \( (v_i, v_j) \in E_p \). In this case, in fact, the spring has a rest length equal to \( \rho \); however we assume the
spring exerts a null force when \( \hat{d}_{ij}(t) \) is smaller than \( \rho \), because the presence constraint is violated only if \( \hat{d}_{ij}(t) > \rho \). Therefore, the spring for a presence constraint has just an attraction effect when the nodes are too far, while the repulsion is not considered, because the nodes are not able to measure the distance between them in first place.

### 4.2. Adding 2-hop information

Let us develop a mechanism to include 2-hop information in the mass–spring optimization framework. Specifically, we consider additional force terms \( F_{ij}^{\text{two}}(t) \) for any node \( v_i \) and any of its 2-hop neighbors\(^1\) \( v_j \in \mathcal{N}_i^{\text{two}} \) such that

\[
\tilde{F}_{ij}^{\text{two}}(t) = \tilde{w}_{ij}(t) F_{ij}^{\text{two}}(t)
\]

where

\[
F_{ij}^{\text{two}}(t) = \begin{cases} 
0, & \text{if } \hat{d}_{ij}(t) \in [\rho, 2\rho] \\
\hat{d}_{ij}(t) - \rho, & \text{if } \hat{d}_{ij}(t) < \rho \\
\hat{d}_{ij}(t) - 2\rho, & \text{if } \hat{d}_{ij}(t) > 2\rho.
\end{cases}
\]

The above equation implies that the spring is activated only if the difference between the measured and calculated distances is below \( \rho \) (there is a repulsion in this case) or above \( 2\rho \) (there is an attraction in this case), while the spring has no effect in the range \([\rho, 2\rho]\); this happens because if the nodes are 2-hop neighbors over a unit disk graph, then their distance must be bigger that \( \rho \) (otherwise they would be neighbors) and smaller than \( 2\rho \) (otherwise they would not have a common neighbor).

As a result, we modify the standard mass–spring algorithm in that we consider

\[
\tilde{F}_i(t) = \sum_{j \in \mathcal{N}_i} \tilde{F}_{ij}^{\text{one}}(t) + \sum_{j \in \mathcal{N}_i^{\text{two}}} \tilde{F}_{ij}^{\text{two}}(t).
\]

The pseudocode of the proposed algorithm coincides with Algorithm 1, when \( \tilde{F}_i(t) \) is calculated according to Eq. (9).

### 5. Simulation results

In Fig. 1 we report an example of execution of the proposed mass–spring algorithm as a postprocessing of the localization algorithm developed in [5]. Specifically, in the figure we report the network topology (upper plot) and the mean and standard deviation over all the non-anchor nodes of the error

\[
\epsilon_i = \| p_i - p_i^* \|
\]

that is, the norm of the difference between the nodes’ actual position \( p_i \) and the estimate \( p_i^* \). We choose in all simulations a threshold \( \eta = 0.0001 \) for the termination of the algorithm. The results in red represent the case of a post-processing using 1-hop and 2-hop information, in blue 1-hop distance and presence information, in dark gray just distance 1-hop information. It can be noted that, the post processing that is based also on 2-hop information is quite effective in reducing the position error, and the results are sensibly better than the other approaches in terms of average error and in terms of reduction of the associated standard deviation. In Fig. 2 we report the result of a test campaign for \( m = 100 \) runs with the same parameter choice of the previous example, except that we vary the fraction of presence nodes. Specifically, we plot a global indicator of the error done, which we define as

\[
\epsilon = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \epsilon_i^2}.
\]

---

\(^1\) Two nodes are 2-hop neighbors if they are not directly connected by an edge of the graph but they have a neighbor in common; we denote the set of 2-hop neighbors of node \( v_i \) by \( \mathcal{N}_i^{\text{two}} \).
According to the figure, the median value for $\epsilon$ is around 0.04 for the initial estimate, regardless of the fraction of range-free nodes. The 1-hop cases (presence and distance) shows a median value for $\epsilon$ is around 0.01 for $\rho < 0.9$. As for the proposed approach based on 1-hop and 2-hop information, we have a median value for $\epsilon$ around 0.002 for $\rho < 0.9$; moreover, the 75th percentile for $\rho < 0.9$ is below 0.01, implying that the suggested algorithm behaves better than the median value of the 1-hop only case in the 75% of the instances considered.

6. Conclusions

In this paper we provide a post processing algorithm for hybrid sensor networks that uses both 1-hop information on distance and presence and 2-hop negative information. According to the simulations, the algorithm is quite effective in reducing the position error, and the relative contribution of the 2-hop information plays a pivotal role in error reduction. Future work will be aimed at providing a real-world implementation of the proposed algorithms and to identify efficient distributed ways to refine an initial rough estimate of the sensors’ positions. We will also inspect the robustness of the proposed approach with respect to accidental failures and malicious false data injections.

References