

# A BP Neural Network Improvement to Hop-counting for Localization in Wireless Sensor Networks

Yurong Xu<sup>1,2</sup>, James Ford<sup>1,2</sup>, Eric Becker<sup>1</sup>, Vangelis Karkaletsis<sup>3</sup>, and Fillia Makedon<sup>1,2</sup>

<sup>1</sup>Heracleia Human Centered Computing Lab, Dept. of Computer Science and Engineering, Univ. of Texas at Arlington, Arlington, TX 76013

<sup>2</sup>DEVLAB, Computer Science Department, Dartmouth College, Hanover NH 03755, USA

<sup>3</sup>Institution of Informatics and Telecommunications, NCSR “Demokritos”, Greece  
{yurong, jford}@cs.dartmouth.edu, becker@uta.edu, vangelis@iit.demokritos.gr, makedon@uta.edu

**Abstract.** GDL is a distributed localization technique designed to allow nodes in a network to determine their relative locations without special hardware or configuration. GDL and other approaches based on hop-counting only require the connectivity of the network; however, previous hop-counting methods have suffered from low accuracy. In this paper, we propose to incorporate neural networks into the standard hop counting-based approach to improve the accuracy of localization over the existing method. We use one neural network at each node to compute its hop-coordinates, after a training procedure. Using simulation, we show that incorporating our technique improves the accuracy of localization results generated by GDL by 15%. The application of this method is not limited to the GDL algorithm; it can also be used in other hop-counting based localization algorithms, such as MDS-MAP. It is also designed to be suitable for implementation in current embedded systems, such as standard motes like Moteiv’s Invent.

**Key words:** Neural Networks, Embedded Computation, Measurement, Hop-counting, Localization

## 1 Introduction

Wireless Sensor Networks (WSNs), an emerging new type of ad-hoc network, integrate sensing, processing and wireless communication in a distributed system. WSNs have numerous applications, such as surveillance, healthcare, industry automation, and military uses [1].

Many location-aware applications of WSNs require the location of nodes to be known. In addition, different protocols in WSNs require that each node be aware of its location as accurately as possible. One such protocol is geographic routing, which is important in many WSNs [21]. Many physical features in WSN have

been utilized by different localization algorithms to generate the location of nodes in a WSN: Angle Of Arrival (AOA) [12], Time Of Arrival (TOA) [6], Time Difference Of Arrival (TDOA) [15], Receive Signal Strength Indicator (RSSI) [24], Radio Interferometry (RI) [8], hop information, and connectivity. Usually, sensor nodes must deploy additional hardware for AOA, TOA and TDOA, or special antennas for AOA. As to RSSI, it depends on a rough relationship between the distance and the signal strength, but this relationship is affected unpredictably by the antenna, battery, electric components inside of the node and the outside environment [24]. RI requires accurate synchronization. In contrast with other physical features which we can use in localization algorithms, hop information and connectivity are much more stable and can be achieved in most currently deployed WSNs without any additional hardware.

Extensive simulation experiments on different localization algorithms [2, 18, 16, 19] based on only hop-counting or connectivity have shown that there is usually a sizeable error in the location generated by the algorithms in comparison with actual locations, even after some refinement processes such as the ones in [18] and [19]. While approximate localization is acceptable for a number of purposes, including routing, increasing the accuracy of localization would be beneficial for many applications.

This paper aims to identify the problems leading to low accuracy in hop-counting, and focuses on how to correct inaccuracies that occur in hop-counting with the help of a neural network approach. For each node in the network, we employ a back-propagation (BP) neural network to let nodes learn and remember their local network structure; using this information, nodes can then distinguish different neighbors one hop away and improve the measurement accuracy of hop-counting. We called this method *neural network based hop coordinates (NN hop coordinates)* [23]. It not only counts the number of hops, but also offsets this total based on the local network structure.

The method can be used in any localization algorithms based on hop-counting. When combined with three well-known localization algorithms of this type—MDS-MAP [19], MDS-MAP(P) [18] and GDL [22]— it achieves an 86%, 25% and 15% improvement, respectively, in our simulation, comparing with results from the same algorithms without NN-based hop coordinates.

## 2 Related Work

Node localization in WSNs has been an active research field for some time, and many localization algorithms have been proposed. Based on the different physical features of WSNs, we can roughly separate these into four categories: (1) some algorithms, such as [2, 3, 7, 10, 11, 13, 17–19], merely utilize the connectivity or hop count information in a WSN; (2) other algorithms, such as [12], use AOA in addition to hop-counting; (3) some, such as [4], uses TOA information; (4) and some (e.g. [9]) use TDOA. Hop information and connectivity are available in currently implemented WSNs, while AOA requires additional devices such as ultrasound receivers [15] or directional antennas, and TOA and TDOA require

nodes to have a second, different propagation speed communication device, such as an ultrasound transceiver [15], in addition to RF transceivers to measure the delivery time. Recently, Spotlight, a light-based localization system that relies on additional hardware, was proposed [20], introducing a fifth category.

A number of methods exist (e.g. [20]) that do not need additional hardware but do require “anchor” nodes (nodes whose positions are known in advance). In this paper, we will only consider localization algorithms that use only hop information and connectivity, and that thus need no additional hardware and no special nodes such as anchor nodes. This will exclude algorithms such as BVR [5], but still leaves us with several algorithms. DV-Hop and DV-distance by Niculescu and Nath [14], and Hop-TERRAIN by Savarese et al. [16] use hop-counting to calculate the location of nodes. However, these methods are outperformed by the following MDS-MAP series of algorithms and by GDL [22].

MDS-MAP [19] uses a multidimensional scaling (MDS) calculation to generate a localization map based on connectivity in WSNs. Shang et al. later presented a second algorithm called MDS-MAP(P) [18], which extends MDS-MAP into a distributed algorithm. MDS-MAP(P) first computes local maps for each node with MDS using local shortest paths, then merges local maps into a global map.

GDL, introduced in [23], provides another variation on hop-counting, which averages the counted hops from neighbor nodes of each node to improve the accuracy of measurement. A problem for GDL is that such a method doesn’t make any allowance for the possibility of an irregular distribution of neighbors around a node. One extreme example is a very sparse network, such as a line of nodes variably spaced to be more than one half of a hop radius apart, but within the hop distance: since each node only has two neighbors with different hop numbers in this case, the averaging calculations used by GDL are not beneficial.

Though the main idea derives from hop-counting, our technique can not only be applied directly in the localization algorithms based on hop-counting, but can also be applied in localization algorithms which need connectivity, such as DV-hop, the MDS-MAP algorithms, and GDL. In our simulation experiments in this work, we apply hop coordinates in the MDS-MAP and GDL algorithms.

### 3 Problems

In this section, we will talk about the reason of why hop-counting technique is not accurate.

#### 3.1 Hop-Counting Technique

Hop-counting needs no special hardware to calculate localization information. It is based on having a pre-appointed but arbitrary bootstrap node send out a hop-counting message with a variable  $hops = 0$  inside; by using that message, each node determines its hop distance from the bootstrap node and forwards that message after accumulating the variable  $hops$ .

Suppose that an arbitrary node  $a$  is calculating this hop distance, and node  $b$  is one of the neighbors of node  $a$ . Then, the basic hop-counting procedure for node  $a$  can be shown as follows in Procedure 1:

---

**Algorithm 1** Compute Hops in node  $a$ 


---

```

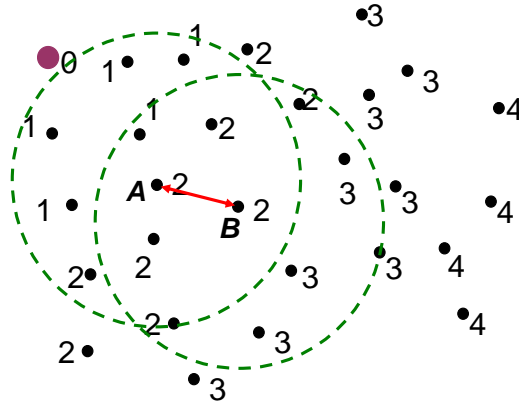
1: INPUT: message ( $hop_b$ ) from node  $b \in N_a$ 
2:  $hop_a = \infty$ 
3: for message ( $hop_b$ ) from any  $B \in N_a$  and not TIMEOUT do
4:   if  $hop_b < hop_a$  then
5:      $hop_a = hop_b + 1$ 
6:     forward (message( $hop_a$ )) to MAC
7:   else
8:     drop( message( $hop_b$ ) )
9:   end if
10: end for
11: RETURN  $hop_a$ 

```

---

Here,  $a$  is a node,  $hop_a$  and  $hop_b$  represent the minimum number of hops to reach node  $a$ ,  $b$  counting from some bootstrap node ( $x$ ).  $N_a$  is a set of nodes which can be reached by node  $a$  as neighbors in  $k$  hops, and  $|N_a|$  is the number of nodes in  $N_a$ . MAC means MAC layer.

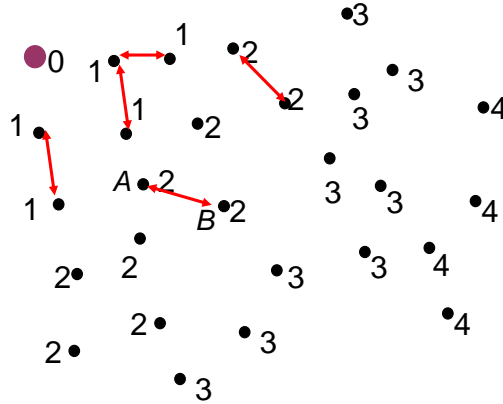
### 3.2 Extending the Hop-Counting Technique



**Fig. 1.** Nodes  $a$  and node  $b$  share same number of hops to the bootstrap node (“hop count”) in a network, despite being significantly far apart. Here, the dots represent nodes in the network, the oversized node is the bootstrap node, and the number near each node is the hop count for that node.

The basic idea behind hop-counting is that there is a special bootstrap node in the network, which sends out a message to flood the network, and all other nodes in the network will use this message to count the number of hops to that bootstrap node. This is an efficient way to measure the distance between any two arbitrary nodes in wireless sensor networks, but a problem is that it only distinguishes nodes based on the number of hops between them, so that many nodes share the same distance from the bootstrap node in terms of their number of hops. One example is shown in Fig. 1.

From Fig. 1, we can see that nodes  $a$  and  $b$  are identified as having the same hop number based on the number of distinct hops it takes to reach them from the bootstrap node; in actuality, nodes  $a$  and  $b$  have different physical distances to the bootstrap node. Unfortunately, this problem occurs frequently in a typical network, as illustrated in Fig. 2.



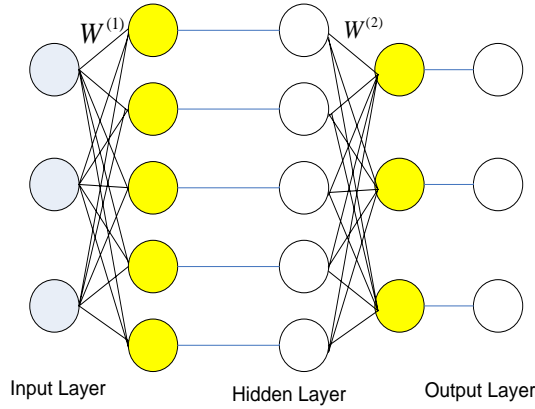
**Fig. 2.** Different nodes share same hop count. After hop-counting, there are many different nodes sharing the same hop counts, some examples of which are indicated by arrows.

In our previous work on GDL, we used an averaging technique to improve the hop-counting technique; here, we introduce a different approach based on BP neural networks.

### 4 Neural Networks

In this paper, we use a standard three-layer neural network with training *via* a back-propagation algorithm. There are three layers, comprising the input layer, output layer, and hidden layer, as shown in Fig. 3.

Given input nodes  $x_k$ , hidden nodes  $h_k$  and output nodes  $y_k$ , we can calculate the values of the latter two of these based on the earlier ones plus sets of weights,  $w^{(1)}$  and  $w^{(2)}$ , between layers:



**Fig. 3.** Neural Networks with Hidden Layer. Inputs (leftmost column) drive the input layer, which in turn causes propagation of information through weighted links ( $W^{(1)}$ ), to the hidden layer; this layer then influences an output layer with additional weighted links ( $W^{(2)}$ ), ultimately driving a set of outputs (rightmost column of nodes)

$$h_k = F^{(1)}[\sum_j w_{ij}^{(1)} \bullet x_i]$$

$$y_k = F^{(2)}[\sum_j w_{jk}^{(2)} \bullet h_k]$$

Given a training set  $\{x_n\}$  and desired target outputs  $\{t_n\}$ , to find  $w^{(1)}$  and  $w^{(2)}$ , we minimize the network error, defined by

$$E = \sum_n |y(x_n) - t_n|^2$$

We use a standard back propagation algorithm to train this neural network. Back propagation uses supervised learning in which the network is trained using data for which inputs as well as outputs are known. This algorithm is a generalization of the least mean squares approach, and modifies network weights to minimize the mean squared error between the desired and actual outputs of the network. The calculation of  $w^{(2)}$  proceeds as follows:

$$\frac{\partial E}{\partial w_{jk}^{(2)}} = \frac{\partial E}{\partial y} \bullet \frac{\partial y}{\partial \sum} \bullet \frac{\partial(\sum_j w_{jk}^{(2)} \cdot h_j)}{\partial w_{jk}^{(2)}} = 2(|y - t|) \cdot F^{(2)}[\sum] \cdot h_j w_{jk}^{(2)} = w_{jk}^{(2)} - a \bullet \frac{\partial E}{\partial w_{jk}^{(2)}}$$

$$\frac{\partial E}{\partial w_{jk}^{(2)}} = \frac{\partial E}{\partial y} \bullet \frac{\partial y}{\partial \sum} \bullet \frac{\partial(\sum_j w_{jk}^{(2)} \cdot h_j)}{\partial w_{jk}^{(2)}} = 2(|y - t|) \cdot F^{(2)}[\sum] \cdot h_j w_{jk}^{(2)} = w_{jk}^{(2)} - a \bullet \frac{\partial E}{\partial w_{jk}^{(2)}}$$

Here,  $a$  is the learning rate.

A similar procedure is used to calculate  $w^{(1)}$ . Once trained, the network weights are identified as fixed values to be used to compute output values for new input samples.

## 5 NN-based Hop Coordinates

### 5.1 Definition of NN-based Hop Coordinate

Unlike most other localization algorithms, which only count number of hops or utilize connectivity between nodes, the proposed *NN-based Hop coordinates*

introduced in this paper not only counts the number of hops from some bootstrap node, but also offsets these counts with local network information pertaining to that node.

**Definition 1:** A *NN-based Hop Coordinate* is constructed from two parts: *number of hops* and *offset*. The first part is a positive integer which equals the number of hops in a minimum hop route from some bootstrap node to a given node. The second part can be seen as a decimal fraction generated from local network information, shown as:

$$\text{hopcoordinates} = \langle \text{hop}, \text{offset} \rangle \text{ (here } a \text{ is any arbitrary node)}$$

We assume that *hop* is equal to or larger than 0 and that  $0 \leq \text{offset} < 1$ .

In the following,  $N_a$  is a set of nodes which can be reached by node  $a$  in  $k$  hops, and  $|N_a|$  is the number of nodes in  $N_a$ .

## 5.2 Constructing the Neural Network

Before we describe our approach, let us first reexamine the example of hop-counting problems shown in Fig. 1. From that picture we can see that even though nodes  $a$  and  $b$  share the same hop count, they do not share the same neighbors, which are shown inside dashed circles.

For any given node, which we can call node  $a$ , we assume  $N_a$  is the set of neighbor nodes (nodes within one hop). We can prove that there is no node  $b$ , such that it is both a neighbor of node  $a$  and has a hop count that differs by more than one as follows:

**Theorem 1:** For a given node  $a$ , if  $N_a$  is the set of neighbor nodes for node  $a$ , there is no node  $b$  which belongs to  $N_a$  and satisfies  $|\text{hop}_b - \text{hop}_a| > 1$ .

**Proof:** suppose there is a node  $b$  that belongs to  $N_a$  and  $|\text{hop}_b - \text{hop}_a| > 1$ . Without loss of generality, assume that  $\text{hop}_b - \text{hop}_a > 1$ . Then, because node  $b$  is node  $a$ 's neighbor, node  $b$  must have received a hop-counting message from node  $a$ , based on the hop-counting algorithm Procedure 1, and since current  $\text{hop}_b > \text{received hop}_a + 1$ , we know  $\text{hop}_b$  must have been updated to  $\text{hop}_a + 1$ . This contradicts the assumption that  $|\text{hop}_b - \text{hop}_a| > 1$ . For the  $\text{hop}_a - \text{hop}_b > 1$  case, the proof is similar. QED.

So, based on **Theorem 1**, for a node  $a$ , there are only three types of nodes in  $N_a$ :

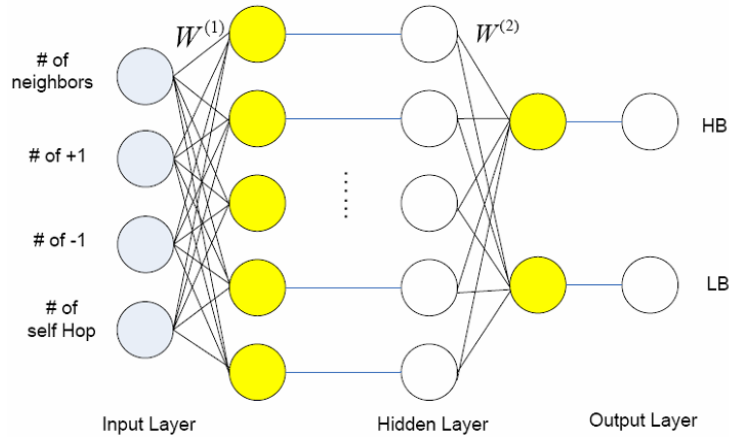
Nodes with a hop count equal to  $(\text{hop}_a + 1)$ , which we designate as  $N_a^{+1}$ .

Nodes with a hop count equal to  $(\text{hop}_a - 1)$ , which we designate as  $N_a^{-1}$ .

Nodes with a hop count equal to  $(\text{hop}_a)$ , which we designate as  $N_a^0$ .

We use the number of nodes of each type, i.e.  $|N_a^{-1}|$ ,  $|N_a^{+1}|$ , and  $|N_a|$ , together with  $\text{hop}_a$  as inputs for the neural network in node  $a$ .

The previous presentation of the NN hop coordinates method assumes we can make use of floating point arithmetic; however, a standard WSN node usually only has an 8 bit or 16 bit microprocessor without floating point support [1]. We thus need to use integer values to represent  $\text{offset}_a$  and  $\text{hop}_a$  (here  $a$  is any arbitrary node). In practice, we can conveniently use one byte each to represent  $\text{hop}_a$  and  $\text{offset}_a$ , allowing values from 0 to 255 which we can interpret as described later in the paper. We designate these bytes as the high byte (HB) for



**Fig. 4.** Neural Networks for NN-based Hop Coordinates. Inputs (leftmost column) drive an input layer, which in turn causes propagation of information through weighted links ( $W_{(1)}$ ), to a hidden layer; this layer then influences an output layer with additional weighted links ( $W_{(2)}$ ), ultimately driving a set of outputs (rightmost column of nodes)

the hop count and low byte (LB) for the offset. If an application needs more resolution for hop-coordinates we can use two bytes, or a double word, for each.

## 6 Simulation

In this section, we will talk about how to setup a simulation environment as well as simulation results.

### 6.1 Simulation Process

There are three steps in our simulation.

*Step 1:* Create training data set from NS-2

A 6-element tuple is used to represent a training data set for arbitrary node  $a$ :  $(|N_a|, |N_a^{-1}|, |N_a^{+1}|, hop_a, LB, HB)$ . The elements represent, respectively, the number of neighbors, number of neighbors with  $(hop_a - 1)$ , number of neighbors with  $(hop_a + 1)$ ,  $hop_a$ , and the LB and HB values described in Section 4.2. In our experiments, these HB and LB are computed as the result of dividing the physical distance from node  $a$  to the bootstrap node by the transmission range  $R$ . We trained with around 64 samples for each node as a training set.

*Step 2:* Process to Compute Hop Coordinates with Neural Network

After we finish training for the neural network in each node, we fix the internal layer in each node and use it to compute the NN-based hop-coordinates for the node by inputting a 4-tuple  $(|N_a|, |N_a^{-1}|, |N_a^{+1}|, hop_a)$  to compute (LB, HB).

*Step 3: Compute Predicted Maps using Localization Algorithms*

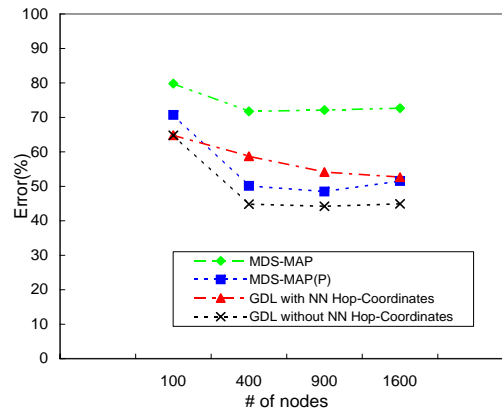
As described in the related work summary above, the GDL and MDS-MAP series of localization algorithms outperform other localization algorithm, so in our simulation we use GDL and MDS-MAP series algorithms to evaluate our technique.

By collecting the NN-based hop-coordinates from neighbors for a node, that node can compute shortest paths between all pairs of nodes in the neighbors using Dijkstra's algorithm. Then, we apply MDS-MAP [19], MDS-MAP(P) [18] and GDL [22] to the above data to compute the relative maps. Given four fixed anchor nodes, we transform these relative maps to absolute maps.

Thus, by comparing with the physical position of the each node in the network, we can obtain and compare the accuracy measurements with and without NN-based hop-coordinates.

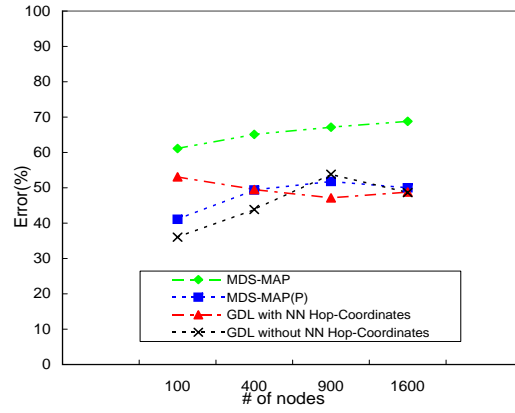
## 6.2 Simulation Results

We obtained results of the accuracy of localization by the MDS-MAP and GDL algorithms, with and without NN-based hop coordinates, on networks with  $n = 100, 400, 900,$  and  $1600$  nodes, and  $r = 2, 4, 6, 8, 10,$  and  $12$  meters, as shown in Fig. 5 for  $r = 2$  through Fig. 10 for  $r = 12$ .

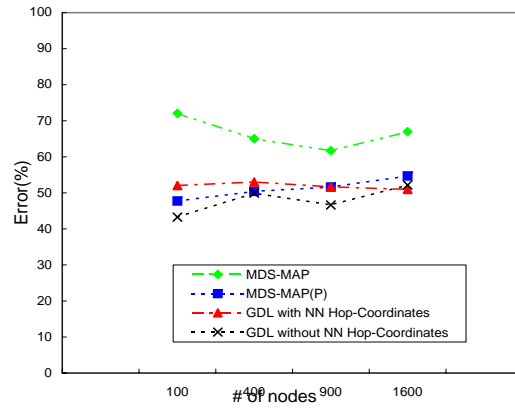


**Fig. 5.** Comparison of the accuracy of localization of MDS-MAP, MDS-MAP(P) and GDL algorithms with or without the NN hop coordinates technique, on networks with  $n = 100, 400, 900, 1600$  nodes, when  $r = 2$ . Notes: X-axis: # of nodes, Y-axis: the error of localization in percentage.

From Fig. 5, 6 and 7 we can see that when  $r = 2, 4, 6$ , the accuracy in localization with NN-based hop coordinates is higher than MDS-MAP, but close to MDS-MAP(P) and GDL. Since MDS-MAP only utilizes connectivity information, NN-based hop-coordinates can help to improve the localization accuracy. In

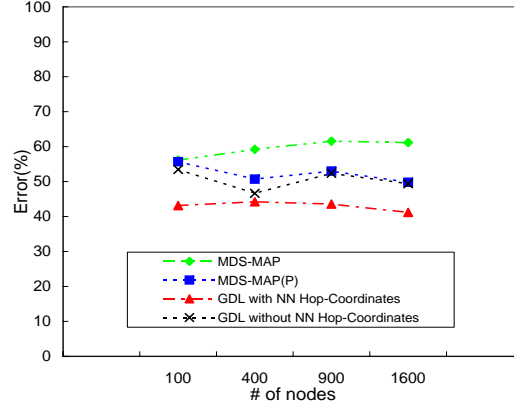


**Fig. 6.** Comparison of the accuracy of localization of MDS-MAP, MDS-MAP(P) and GDL algorithms with or without the NN hop coordinates technique, when  $r = 4$ .



**Fig. 7.** Comparison of the accuracy of localization of MDS-MAP, MDS-MAP(P) and GDL algorithms with or without the NN hop coordinates technique, when  $r = 6$ .

GDL and MDS-MAP(P), in contrast, the fact that both incorporate techniques to improve accuracy, and that the network density is high, means that NN-based hop-coordinates provide little additional benefit.



**Fig. 8.** Comparison of the accuracy of localization of MDS-MAP, MDS-MAP(P) and GDL algorithms with or without the NN hop coordinates technique, on networks with  $n = 100, 400, 900, 1600$  nodes, and  $r = 8$ .

From Fig. 8, 9, 10, we can see that when  $r > 6$  meters, improvements of accuracy in coordinates are significant in MDS-MAP, MDS-MAP(P) and GDL. As a result of the NN training process, NN-based hop-coordinates provide a larger accuracy improvement than the techniques used in the MDS-MAP and GDL algorithms.

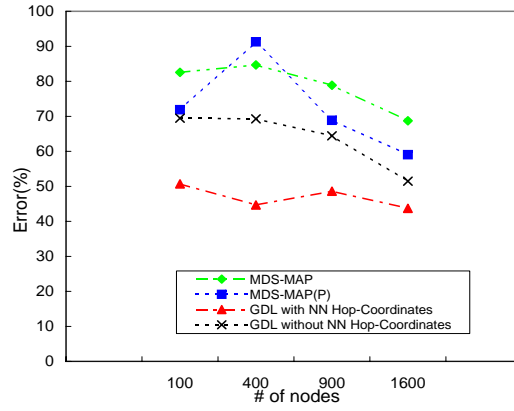
Then, we compute the overall average accuracy based on Fig. 5 to Fig. 10, corresponding to  $r = 2, 4, 6, 8, 10, 12$  meters. The results are shown in Fig. 11. From it, we can see that in the MDS-MAP, MDS-MAP(P) and GDL algorithms, the localization accuracy with our technique is about 86% and 26% and 15% improved compared with the GDL algorithms using only hop-counting, when  $2 \leq r \leq 12$ .

### 6.3 Costs

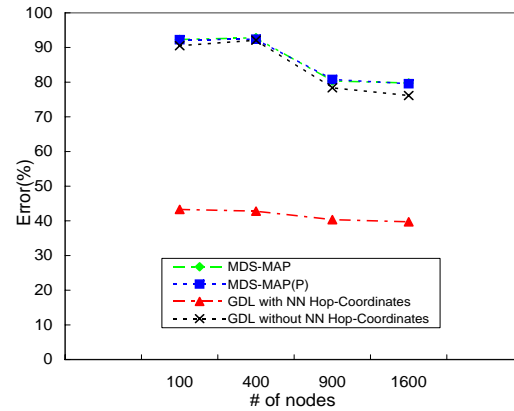
As a setup phase, our technique requires a significantly long training time, but afterwards the technique has a low computational cost: in essence, it requires the multiplication of two small matrices. The approach described here requires very little memory to store the neural network parameters in each node.

## 7 Summary and Conclusions

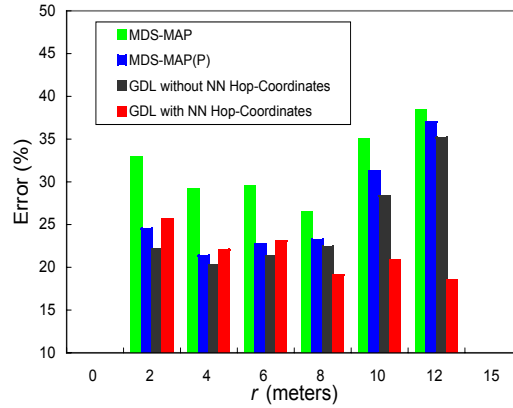
In this paper, we presented a new localization technique, Neural Network based Hop-Coordinates, which when incorporated into existing connectivity-based al-



**Fig. 9.** Comparison of the accuracy of localization of MDS-MAP, MDS-MAP(P) and GDL algorithms with or without the NN hop coordinates technique, when  $r = 10$ .



**Fig. 10.** Comparison of the accuracy of localization of MDS-MAP, MDS-MAP(P) and GDL algorithms with or without the NN hop coordinates technique, when  $r = 12$ .



**Fig. 11.** comparison of overall average accuracy of localization with or without hop coordinates

gorithms improves their accuracy of localization. In simulations, our technique on average improved the accuracy of locations generated by MDS-MAP(P) by 26%, and those generated by GDL 15%, in comparison with the results without applying our technique. We also achieve good results in comparison to GDL in combination with another variation of hop coordinates based on using an average of the neighborhood hop count as the hop coordinates of each node [22]. Our technique works well in networks with varying numbers of nodes, especially when there are sparse nodes, which is typically the worst case for GDL otherwise.

Through a simulation, we show how our technique can work with different algorithms, particularly the MDS-MAP series of algorithms and GDL, which require only connectivity. Our technique can also be applied to other localization algorithms, such as DV-hop and DV-distance, Hop-TERRAIN, which rely on hop-counting techniques, without modification.

In cost, this method has a low computational cost after its training process, and can be implemented in microprocessors without a floating-point unit (FPU).

**Acknowledgments.** We acknowledge the support from the National Science Foundation under award Number ITR 0312629 and IDM 0308229.

## References

1. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, 2002.
2. R. Bischoff and R. Wattenhofer. Analyzing connectivity-based multi-hop ad-hoc positioning. *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 165–174, 2004.
3. W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 896–901, 1996.

4. S. Čapkun, M. Hamdi, and J. Hubaux. GPS-free Positioning in Mobile Ad Hoc Networks. *Cluster Computing*, 5(2):157–167, 2002.
5. R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensor networks. *Proceedings of NSDI 2005*, pages 329–342, 2005.
6. L. Girod, V. Bychkovskiy, J. Elson, and D. Estrin. Locating tiny sensors in time and space: A case study. In *Proceedings of the International Conference on Computer Design (ICCD 2002)*, 2002.
7. T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 81–95, 2003.
8. M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, Á. Lédeczi, G. Balogh, and K. Molnár. Radio interferometric geolocation. *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 1–12, 2005.
9. D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, 2004.
10. R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. *Proc. of Information Processing in Sensor Networks (IPSN)*, 2003.
11. D. Niculescu and B. Nath. Ad-Hoc Positioning Systems (APS). *Proceedings of IEEE GLOBECOM*, 1:25–29, 2001.
12. D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 3, 2003.
13. D. Niculescu and B. Nath. DV Based Positioning in Ad Hoc Networks. *Telecommunication Systems*, 22(1):267–280, 2003.
14. D. Niculescu and B. Nath. Error characteristics of ad hoc positioning systems. *Proc. of Intl. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.
15. N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43, 2000.
16. C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. *USENIX Technical Annual Conference*, pages 317–328, 2002.
17. Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz. Localization from connectivity in sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 15(11):961–974, 2004.
18. Y. Shang and W. Ruml. Improved MDS-based localization. *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 4, 2004.
19. Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 201–212, 2003.
20. R. Stoleru, T. He, J. Stankovic, and D. Luebke. A high-accuracy, low-cost localization system for wireless sensor networks. *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 13–26, 2005.
21. M. Vieira, C. Coelho Jr, D. da Silva Jr, and J. da Mata. Survey on wireless sensor network devices. *IEEE Emerging Technologies and Factory Automation*, pages 537–544, 2003.

22. Y. Xu, J. Ford, and F. Makedon. GDL: A Geographic Distributed Localization Algorithm for Wireless Sensor Networks. *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings. 15th International Conference on*, 2006.
23. Y. Xu, J. Ford, and F. S. Makedon. A Variation on Hop-counting for Geographic Routing. *Embedded Networked Sensors, 2006. EmNetS-III. The third IEEE Workshop on*, 2006.
24. G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic. Impact of radio irregularity on wireless sensor networks. *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138, 2004.