Title	A Modified Differential Evolution with Heuristics Algorithm for Non-convex Optimization on Sensor Network Localization
Author(s)	Qiao, D.P.; Pang, GKH
Citation	IEEE Transactions on Vehicular Technology, 2015
Issued Date	2015
URL	http://hdl.handle.net/10722/214171
Rights	Creative Commons: Attribution 3.0 Hong Kong License

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

A Modified Differential Evolution with Heuristics Algorithm for Non-convex Optimization on Sensor Network Localization

Dapeng Qiao, Grantham K.H. Pang, *Senior Member, IEEE*

***Abstract*—The sensor network localization based on connectivity can be modeled as a non-convex optimization problem. However, current models only consider the convex constraints i.e. connections among the nodes. The proposed method considers not only the connection constraints, but also the disconnection constraints, which are non-convex in nature. It is argued that the connectivity-based localization problem should be represented as an optimization problem with both convex and non-convex constraints. In this paper, an algorithm combining a modified differential evolution (DE) algorithm and heuristics is presented for the situation that the communication range value is unknown. The developed algorithm has a new crossover procedure, with refined procedures to produce a new generation of individuals/candidates. A "single node treatment" procedure is also designed for the search procedure to formulate a new set of coordinate locations to jump out from the local minimum. The final solution can reach the most suitable configuration of the unknown nodes (nodes without knowing their location) because all the information on the constraints has been used. Simulation results have shown that better solutions can be obtained when compared with other convex-constraint methods. The proposed method also gives better result than other general non-convex optimization methods.**

***Index Terms*—wireless sensor network, localization, connectivity, optimization, differential evolution, non-convex constraints.**

I. Introduction

Position estimation is necessary in many applications such as remote patient monitoring, package and person tracking, environment monitoring and wildlife habitat monitoring. In these systems, there could be hundreds of low-cost sensor nodes, which can take some simple measurements. Based on either the distances or the connectivity among the nodes, we would like to estimate the location of these nodes in the sensor network. It is necessary to accurately localize the sensors in order to measure data which is geographically meaningful.

For applications like automatic guidance, and wildlife habitat tracking, GPS-like devices are widely used. However, GPS devices are expensive and inefficient on power consumption [1]. Thus, in sensor networks with a large number of sensor nodes, attaching a GPS device to each node is not practical. In most cases, there are only a few nodes with known positions in the whole sensor network, while others are unknown. The only information between the known nodes (nodes knowing their location) and the unknown nodes (nodes without knowing their location) is the communication among them, which can imply the distance or connectivity between the nodes. Localization in a sensor network is to use any useful information for the best position estimation of the unknown nodes. As connectivity requires less hardware and is much cheaper to establish than distance measurement, connectivity-based localization is more attractive. When having obtained the connectivity information between any pair of nodes, a good algorithm to abstract useful information for localization and to serve accurate position estimation is the challenge. This paper concentrates on the localization algorithms based on connectivity.

The current solutions of the connectivity-based localization problem can fall into two categories. The first class of methods tries to find the number of direct connections between two nodes. In other words, the number of hops from one node to another needs to be calculated. Hence, they use the hop count to roughly represent the distance between two nodes. The centroid method [2], the Approximate Point In Triangulation (APIT) [3], the multidimensional scaling–MAP (MDS–MAP) (also known as MDS) [4], and Distance Vector–Hop (DV-Hop) [5] all belong to this category. The other class of methods models the connectivity-based localization problem as a constrained optimization problem. The connectivity information becomes the constraints that the optimization result must satisfy. For example, convex position estimation (CPE), also called semi-definite programming (SDP) [6], selects the convex constraints and formulates the problem as a convex optimization problem. This method has been used as a starting point for further searching [7] in distance-based localization. However, due to the lack of non-convex constraints, the solution obtained tends to have the estimated nodes crowd together, and so the nodes cannot "keep distance" from each other, which could give an overall erroneous result in practice. Besides, SDP cannot work in the case that communication range is not available. When the communication range is assumed unknown, the available algorithms included centriod localization, MDS and DV-hop, which are introduced next.

Centroid localization is probably the earliest and simplest approach. A proximity-based and coarse approach is proposed by Bulusu and Heidemann [2]. Every unknown node receives several nearby anchors' information. The location information

Dapeng Qiao is with the School of Automation, Beijing Institute of Technology. Beijing, China. (e-mail: dpqiao@eee.hku.hk).

Grantham K.H. Pang is with the Industrial Automation Research Laboratory of Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong (e-mail: gpang@eee.hku.hk).

This work was supported by State Key Laboratory of Intelligent Control and Decision of Complex Systems in China. It was also sponsored by the National Natural Science Foundation of China under Grant NO. 61433003 and School Foundation of Beijing Institute of Technology under Grant NO. 20130642011.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

of the anchors is used, and the estimated location of the unknown node is assumed to be the average of the location of all the nearby anchors.

The basic MDS method [4, 8] can estimate the positions of all the unknown nodes by using the distance information between any two nodes. An extension of MDS [4, 9, 10] for the connectivity-based localization problem has also been developed. First, a rough estimate of the relative node distance is obtained based on hop count information. One hop is one direct connection between two nodes. The hop count between any two nodes roughly represents the distance. Then, the relative positions are calculated by Singular Value Decomposition (SVD) [11] on the estimated distance information matrix. Finally, absolute positions of the unknown nodes are estimated based on the relative positions and the positions of the anchors. The computational complexity of this method is about $O(n^3)$ time for a sensor network of n nodes. MDS has also been modified for the connectivity-based localization problems based on the hop count information to replace the estimated distance between a pair of nodes [10].

Another well-known localization algorithm is DV–Hop [5, 12-14]. The idea of DV–Hop is to transform the distance to all anchors from hops to units of length measurement using the average size of a hop. DV–Hop was first proposed by Niculescu [14], and improved by many researchers. Anchors broadcast their location information to other anchors, and such information will be flooded with the hop count increment. Every anchor knows the hop count from any other anchor, and uses this information to estimate the average hop size. The distance between an anchor and an unknown node is computed by the hop size and the hop count between them. Finally, trilateration is used when the distances between an unknown node and at least three anchors are obtained by the above computation.

These connectivity based algorithms only consider the connections between nodes, and ignore the disconnections. Disconnection is actually important information on connectivity, which can provide a better solution with nodes "keeping distance" from each other. However, if disconnections information is used, the computational complexity of the localization algorithm will be greatly increased, which most researchers try to avoid. As far as we know, there is no other research which takes disconnections into consideration to calculate the sensor locations. The aim of this paper is to utilize the disconnection information and a new algorithm based on modified differential evolution algorithm has been developed to deal with all connectivity information of the network. This paper first gives a formal definition of the connectivity-based localization problem. The connectivity-based localization is formulated as a non-convex optimization problem with connections being modeled as convex constraints, and the disconnections being modeled as non-convex constraints. An algorithm based on the differential evolution and heuristics of the localization problem is proposed and compared with the available localization algorithms in the same situation such as centroid, MDS and DV-Hop. In addition, the proposed algorithm is compared with other non-convex optimization methods. In our previous work [15], an 'active-set algorithm' has been used to solve this non-convex optimization problem, and will be compared with the new algorithm in this paper. Furthermore, a widely used two-objective evolutionary algorithm called Pareto Archived Evolution Strategy (PAES) [16] was used as a benchmark solution to the problem and it is also compared with the new algorithm.

II. Problem Definition

The situation considered in this paper is that every node (including anchors) has identical communication range, which is unknown and needs to be estimated. The communication area of every node is modeled as a perfect disk, which means its antenna is omni-directional. The connection is established if and only if another node is within this disk. The known information includes some anchors' locations and the connectivity information between any two nodes. The following description and simulations of the algorithms are all based on this formulation.

A formal definition of the connectivity-based localization problem is given in this section. Let $G_{network} = (V, E)$ be a given network, where V denotes the nodes of the network and E denotes the edge of the network. Let V be partitioned into two sets: $V_a = \{1,...,m\}$ of anchors; $V_b = \{m+1,...,m+n\}$ of sensors (unknown nodes). E is also partitioned into two sets: $E_{ab} = \{(i,j) \in E : i \in V_a, j \in V_b\}$ which are the edges between a sensor and an anchor; $E_{bb} = \{(i,j) \in E : i, j \in V_b\}$ which are the edges between two sensors.

For each anchor $i \in V_a$, the position $a_i \in \Re^2$ is assumed to be known. For each sensor $i \in V_b$, the position $b_i \in \Re^2$ is assumed to be unknown. Let $C_{ab} = \{(i,j,k) : i \in V_a, j \in V_b, k \in \{0,1\}\}$ be the connectivity information between a sensor and an anchor. Also let $C_{bb} = \{(i,j,k) : i, j \in V_b, k \in \{0,1\}\}$ be that between two sensors. The value k in C_{ab} or C_{bb} is binary (either 0 or 1): k =0 if there is no connection between node i and j; k =1 if there is connection between node i and j. Let a be a vector containing the positions of the anchors $a = (a_i)_{i \in V_a} \in \Re^{2m}$.

The goal of the network localization problem is to determine the coordinates of all the sensors: $b = (b_i)_{i \in V_b} \in \Re^{2n}$, such that b satisfies the following constraints. Let R be the maximum distance (called the range) within which connectivity can be established. $\|a_i - b_j\|_2^2$ is the distance between one anchor and one unknown node; $\|b_i - b_j\|_2^2$ is the distance between two unknown nodes.

$$\text{If } k=1 \begin{cases} \|a_i - b_j\|_2^2 \le R^2 \ \ for\ (i,j) \in E_{ab} \\ \|b_i - b_j\|_2^2 \le R^2 \ \ for\ (i,j) \in E_{bb} \end{cases}$$

$$\text{else } k=0 \begin{cases} \|a_i - b_j\|_2^2 > R^2 \ \ for\ (i,j) \in E_{ab} \\ \|b_i - b_j\|_2^2 > R^2 \ \ for\ (i,j) \in E_{bb} \end{cases}.$$

From the inequalities based on whether any two sensors

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

(including one sensor and one anchor) are in connection or not, the constraints behind those inequalities can be classified into convex constraints and non-convex constraints as shown in Figure 1 and Figure 2.

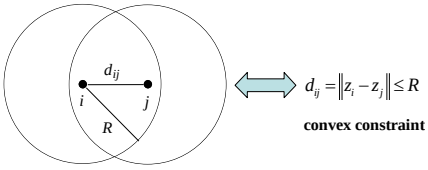

Figure 1 A convex constraint is established when k_{ij} =1, d_{ij} is the distance between node *i* and node *j*; z_i , z_j are the coordinates of node *i* and node *j*

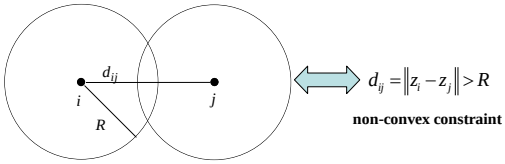

Figure 2 A non-convex constraint is established when k_{ij} =0, d_{ij} is the distance between node *i* and node *j*; z_i , z_j are the coordinates of node *i* and node *j*

III. The Modified DE with Heuristics Algorithm for Sensor Network Localization

There are many methods for non-convex optimization, such as Particle Swarm Optimization (PSO), Simulated Annealing (SA), Genetic Algorithm (GA), and other evolutionary algorithms. Researchers have applied them in sensor network localization, but just limited to range-based scenario. Terwilliger *et al.* [17] and Zhang *et al.* [18] both use evolutionary algorithm to tackle the localization problem in which the distances among the nodes are known. The target of the evolutionary algorithm is to minimize the difference between the known distances and the distances based on estimated nodes position. Tam *et al.* [19] use a genetic algorithm (GA) /evolutionary algorithm to estimate the position of one single node based on its hop counts to its three nearest anchors. In their method, there is a GA for estimating the position of each unknown node. The computational complexity is small also because the scale of the GA is small (population<30). In each GA, only part of the population with better performance is used in computation, which also decreases the computation cost. Hence, the evolutionary algorithm by Tam should be used for many times to estimate all the unknown nodes. The accuracy is similar to the DV-Hop due to the same principle in utilizing the hop count to anchors. Diana *et al.* [20] also utilize soft computing approach to the range-based localization, which is summarized as two objective functions: Cost Function and Soft Constraint Violation. Other popular non-convex optimization algorithms, such as interior-point algorithm [21], are also tried in range-based scenario. In total, current non-convex optimization methods achieved good accuracy in range-based scenario. However, for the range-free scenario, there is still no accurate algorithm because the distance information is replaced by connectivity information, which makes the problem more difficult. In this paper, we analyze the characteristics of connectivity-based nodes, and utilize them to modify evolutionary algorithm. As an evolutionary algorithm, differential evolution algorithm is used in this paper not only because it is easy to handle, but also because its idea of using difference between two individuals is similar with movement of a node from one position toward another position. Therefore, differential evolution is suitable for the node position estimation in nature.

A. Introduction of Differential Evolution (DE) Algorithm

Differential Evolution (DE) is a population-based method that optimizes a problem by iteratively trying to improve a candidate solution with regard to the value of its objective function(s). The problem that it can solve can be nonlinear and non-differentiable. During every iteration of its computation, there is a group of candidate solutions called population, and each candidate is called an individual in this population. Population is improved generation (same to iteration) by generation untill one individual in it is found to be satisfactory. The improvement is the core part of this algorithm, which contains three procedures: mutation, crossover and selection in Figure 3. DE, which is firstly introduced by Storn and Price [22], shows more efficiency than other non-convex algorithms such as simulated annealing, genetic algorithms [22] and evolutionary programming [23]. Differential evolution is called "differential" because its mutation procedure introduces the difference between two individuals into the next population. To help illustrate our proposed algorithm, the normal DE algorithm is first introduced below.

An optimization target has been formulated to minimize *f(x)*, where *x* is the vector containing the variables. Each individual in any population is an estimate of *x*. In the first stage, the first population is generated by randomness or given information. Then each individual in the population will be operated by the following procedures: mutation, crossover and selection. The "better" individuals (individuals whose objective function *f(x)* is smaller) are selected into the next population, i.e. the population in the next generation. The loop will stop when a satisfactory individual has been found.

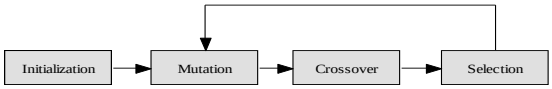

Figure 3 The main procedures of differential evolution (mutation, crossover, and selection are included in the loop)

Mutation: Assume that the number of individuals in a population is *NP* (number of population members, also called population size). For the first population, each individual can be initialized randomly. The individuals in the next population will be generated from the current population. For each individual in a population indexed by *G*, a trial vector *v* is generated according to

$$v = x_{i,G} + \lambda(x_{best,G} - x_{i,G}) + F(x_{a,G} - x_{b,G}) \quad (1)$$

where $x_{i,G}$ is the *i*th individual in population *G*, $x_{a,G}$ and $x_{b,G}$ are two other individuals in population *G*, which are different from $x_{i,G}$; $x_{best,G}$ is the "best" individual in this population (i.e. the objective function of $x_{best,G}$: *f*($x_{best,G}$) is the smallest among all the individuals in population *G*); λ and *F* are two parameters less than 1, which make *v* contains the information of the best individual, and the difference between two random individuals. λ makes *v* close to the best individual. That is: *v* is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

closer to $x_{best,G}$ when λ becomes larger. However, λ cannot be 1, otherwise v will become $x_{best,G}$ and lost its own diversity. On the other hand, F brings some randomness from two random individuals, which should be smaller than 1 to avoid vibration. Besides, these two parameters cannot be too small to influence v. In a typical implementation, to obtain enough information from $x_{best,G}$, $x_{a,G}$ and $x_{b,G}$, λ is set to be 0.8 and F is 0.9 [24, 25]. When x consists of only two variables, this mutation procedure can be illustrated by Figure 4. The small circles are the individuals in population G. The newly generated vector v is denoted as the black dot. By using $x_{a,G}$, $x_{b,G}$, and $x_{best,G}$, v has a chance to become "better" than $x_{i,G}$

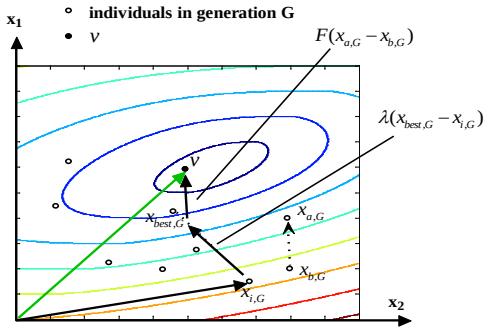

Figure 4 Geometrical illustration of mutation to generate v for a two–dimensional case

Crossover: After deriving the trial vector v in mutation, the next step is crossover. In order to increase diversity of the individuals, the crossover procedure uses some elements in the current individual $x_{i,G}$ and other elements in v, and combines them into a new vector u. An example to illustrate how the new vector u is generated can be found in Figure 5. The elements of u are copied from the corresponding elements v or $x_{i,G}$ with the same index. In the example, elements 3, 4, and 6 are chosen from v, while other elements, i.e. 1, 2, 5, and 7 are copied from $x_{i,G}$. It must be noted that in a typical DE, for any element in u, whether it is chosen from v or from $x_{i,G}$ is totally random. This can introduce randomness to the population, and help generate individuals which may be more suitable for the optimization target. This procedure can also be explained by the equation (2), where the function *rand(j)* returns a random number from 0 to1; *CR* is a constant that defines which vector will make more contribution to u. It is always bigger than 0.5 because v has higher chance to be "better" than $x_{i,G}$. In most cases, it is usually set to be around 0.8 by experience to make the crossover more efficient [24].

$$u_j = \begin{cases} v_j & if\ rand(j) \le CR \\ (x_{i,G})_j & if\ rand(j) > CR \end{cases} \quad (2)$$

where $v_j, u_j, (x_{i,G})_j$ are the jth elements in vector $v, u, x_{i,G}$, respectively; $rand(j)$ is a random number from 0 to 1.

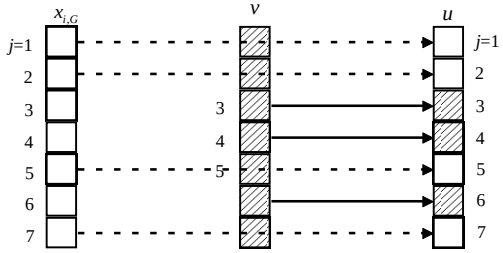

Figure 5 Random choice of elements in crossover of a typical DE

Selection: In the end, the individual $x_{i,G+1}$ in the next population is generated. The "better" vector between u and $x_{i,G}$ is selected as $x_{i,G+1}$.

$$x_{i,G+1} = \begin{cases} u & if\ f(u) < f(x_{i,G}) \\ x_{i,G} & otherwise \end{cases} \quad (3)$$

Since DE was developed, there have been many applications of DE to solve the optimization problems in various domains of engineering including electromagnetics [26], power saving [27], control systems [28] and image processing [29, 30]. The DE algorithm becomes popular because it has demonstrated good convergence properties and is easy to understand in principal [31]. Nowadays, there are still some researchers focusing on improving DE's performance. They mostly concentrate on finding proper setting of the control parameters, i.e. *F*, *CR*, and *NP*, to expedite the convergence velocity. The determination of values for those three parameters has been studied. It has been suggested that a good choice would be: *F*=0.4 to 0.95, *CR*=0.9, and *NP* is 3 to 8 times of the dimension of the variable vector [32]. A fuzzy adaptive DE is introduced by Liu *et al.* [33], which uses fuzzy logic controllers to adapt the parameters *F* and *CR*. The simulation on some standard test functions shows that the fuzzy adaptive DE can converge faster than DE with fixed *F* and *CR* when the dimensionality of the problem is high or the problem concerned is complicated [34]. Other self-adaptive DE algorithms [31] use self-adaptive method to determine the value of *F* and *CR*, which considers *F* and *CR* as the last two additive elements in each variable vector.

Moreover, there are some variants of DE to speed up convergence such as more complicated mutations [35], and "current to *pBest*" mutation [36] which is similar to the crossover procedure in this section, with the current individual is replaced by the top 10% individuals in the past populations. In [37], computational complexity of DE has been discussed and various stopping criteria is investigated from the viewpoint of computational complexity, which is $O(NP*D*G_{\max})$, where D is the dimension of variable vector, $G_{\max}$ is generation number that the algorithm will stop at. DE is also used in multi-objective optimization. A mathematical modeling and convergence analysis of a continuous multi-objective differential evolution is studied in [38]. It must be noted that like many other evolutionary algorithms, there is no proof of convergence for DE.

Our wireless sensor network localization problem is a non-convex and non-differential problem. The original DE algorithm was attempted in our problem, but no convergence has ever been achieved when the number of nodes exceeds ten (i.e. with twenty variables). In the following sections, a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

modified DE algorithm will be presented.

B. The Objectives and the Variable Vector

The connectivity based localization problem is modeled as minimizing two objectives: "*wrong_connection_count*" and "*wrong_distance*", which describe the number of wrong connections and the error distance caused by these wrong connections, respectively. DE evaluates the two objectives of the candidate solutions in each generation, and uses them to find out the 'better' candidates. Wrong connection contains two cases: connection between a pair of nodes is mistaken/violated as disconnection, and disconnection is violated as connection. "*Wrong_connection_cout*" is used to count wrong connections of all node pairs in a candidate solution. If the connectivity (including connection and disconnection) between a pair of nodes is violated, the distance between them indicates how serious the violation is. Therefore, "*wrong_distance*", which describe the error between this distance and the range value, is set to be the second objective.

The modified DE algorithm aims to minimize the above two objectives, with a combination of the original evolutionary algorithm and some heuristics. Besides, the value of the communication range of a sensor can be assumed to be unknown, which makes the search for a solution more difficult. The variable vector (i.e. candidate solution) for this localization problem includes the unknown coordinates and the estimated range $\hat{R}$. Let n be the number of unknown nodes in the localization problem. The dimension of an individual, which is an estimation of the variable vector (T) is therefore $2n+1$ as below. (Note that m denotes the number of anchors; unknown nodes are node $m+1$, node $m+2$, …, node $m+n$).

$\hat{x}_{m+1}$	$\hat{y}_{m+1}$	$\hat{x}_{m+2}$	$\hat{y}_{m+2}$	$\hat{x}_{m+3}$	$\hat{y}_{m+3}$	…	$\hat{x}_{m+n}$	$\hat{y}_{m+n}$	$\hat{R}$

The pseudo-code for the two objectives: "*wrong_connection_count*" and "*wrong_distance*" are shown as Figure 6 and Figure 7, where T is the variable vector, and E_{ab} , E_{bb} have been introduced in Section II. Note that k_{ij} =0 or 1 shows that node i and j are disconnected or connected, which is the given information. On the other hand, ($\hat{R}$ - $\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$)>=0) and ($\hat{R}$ - $\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$)<0) indicates the node i and node j in the candidate solution T are disconnected or connected.

```
function wrong_connection_count(T)
   wrong_connection_count ←0;
   for all (i,j) ∈ E_ab ∪ E_bb
      if ( k_ij =1&( R̂ - sqrt((x_i-x_j)^2+(y_i-y_j)^2) )<0) |
      ( k_ij =0&( R̂ - sqrt((x_i-x_j)^2+(y_i-y_j)^2) )>=0)
         wrong_connection_count←wrong_connection_count+1;
      end if
   end for
return wrong_connection_count
```

Figure 6 Pseudo-code for the first objective *wrong_connection_count(T)*

```
function wrong_distance (T)
   wrong_distance ←0;
   for all (i,j) ∈ E_ab ∪ E_bb
      if ( k_ij =1&( R̂ - sqrt((x_i-x_j)^2+(y_i-y_j)^2) )<0) |
      ( k_ij =0&( R̂ - sqrt((x_i-x_j)^2+(y_i-y_j)^2) )>=0)
         wrong_distance ←wrong_distance +abs( R̂ - sqrt((x_i-x_j)^2+(y_i-y_j)^2) );
      end if
   end for
return wrong_distance
```

Figure 7 Pseudocode for the second objective *wrong_distance (T)*

C. The Characteristics of Sensor Network Localization Problem

As the range value is unknown, the problem is more complex because other variables (the coordinates of the unknown nodes) depend on this range value to count the number of "wrong connection" and calculate the "*wrong_distance*". However, different from the usual optimization problems, the localization problem has its own characteristics, which can be used to help with convergence. Below are the characteristics of the problem:

1. The variable vector (T) is formed by the coordinate pairs of the unknown nodes. When searching for the positions of the n unknown nodes, the position of one node may be moved while the other n-1 nodes and the range value remain static. Therefore, it is reasonable to partition an individual $x_{i,G}$ as in the formulation of T. Furthermore, there exists a direction to search for the node locations. That is: between the two nodes in a wrong connection, the node with less wrong connections to its neighboring nodes is preferred to stay unchanged.

2. Once a wrong connection is encountered, the searching for the new position of the related nodes also has a direction. There are two kinds of wrong connections:

(a) A connection has been mistaken as a disconnection;

(b) A disconnection has been mistaken as a connection.

Situation (a) can be illustrated by the relationship between node i and k in Figure 8. From the known connectivity information, they are connected, which means the distance between them should be less than the range value. However, the estimation (shown in their positions in Figure 8) violates this relationship. Therefore, one node should be moved towards the other untill the distance between them is less than the range value (as shown by the gray arrows). On the other hand, situation (b) illustrated by node i and j should be corrected by moving away from each other (as shown by the white arrows). Therefore, the direction of node movement should not be completely random, and the above heuristic should be incorporated into the algorithm.

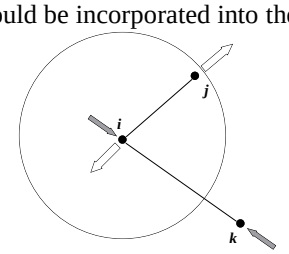

Figure 8 The illustration of wrong connections

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

D. New Crossover

Based on the first characteristic, each movement of a single node would correspond to the change of two variables in the crossover procedure of DE. The new crossover, which is illustrated in Figure 9, utilizes this characteristic and makes crossover follow a direction. Compared with the typical crossover, this new procedure makes the crossover not "blind" any more. The variables in *u* is still constructed by part of *v* and part of $x_{i,G}$, however the choice is not totally random, which is different from typical crossover. The rule of the choice is: if a pair of variables (presenting one node location) in *v* (the mutation result) has better performance than the corresponding pair of variables in $x_{i,G}$, the pair of variables in *v* will be chosen as the elements in *u*. Otherwise, the pair of variables in $x_{i,G}$ is copied to *u*. In this manner, *u* has higher chance to outperform $x_{i,G}$, which makes crossover more efficient.

The evaluation should be reasonable, while it must be with acceptable computation complexity because it is done as a part of crossover. In this algorithm, the performance of each pair of variables is evaluated by the two objectives: wrong_connection_count and wrong_distance. The searching direction is based on these objectives with respect to each node. In the example of Figure 9, the node *m*+1: (x_{m+1}, y_{m+1}) and node *m*+3: (x_{m+3}, y_{m+3}) in *v* are assumed to make *wrong_connection_count* smaller than the corresponding nodes in $x_{i,G}$, or with equal *wrong_connection_count* but smaller *wrong_distance*. In this step, to reduce the computational complexity, the two objectives are simplified. When counting wrong connections of a particular node, for example, node *m*+1, only the wrong connections related to node *m*+1 (i.e. wrong connections between node *m*+1 and any other nodes) are counted. The wrong connections not related to node *m*+1 are ignored to reduce computation cost. The two objectives with respect to one node are just used in the new crossover and can be found in equation (4). It must be noted that the last element of *u*, which represents the value of *R* (range), is set to be the last element of *v*.

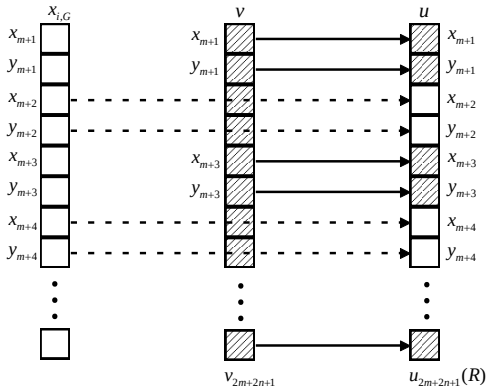

Figure 9 Choice of the elements in the new crossover, which should satisfy equation (4)

$$[u_{2k-1}\ u_{2k}] = \begin{cases} [v_{2k-1}\ v_{2k}] & \text{if } case1: \\ & wrong\_connection\_count([v_{2k-1}\ v_{2k}]) < \\ & wrong\_connection\_count([(x_{i,G})_{2k-1}\ (x_{i,G})_{2k}]) \\ & or\ case2: \\ & wrong\_connection\_count([v_{2k-1}\ v_{2k}]) = \\ & wrong\_connection\_count([(x_{i,G})_{2k-1}\ (x_{i,G})_{2k}]) \\ & and\ wrong\_distance([v_{2k-1}\ v_{2k}]) < \\ & wrong\_distance([(x_{i,G})_{2k-1}\ (x_{i,G})_{2k}]) \\ [(x_{i,G})_{2k-1}\ (x_{i,G})_{2k}] & otherwise \end{cases} \quad (4)$$

where *k* is the index of the unknown nodes, $k = m+1, m+2, \ldots, m+n$; $v_{2k}, u_{2k}, (x_{i,G})_{2k}$ mean the 2*k*th element in vector $v, u, x_{i,G}$, respectively. *wrong_connection_count* and *wrong_distance* are functions with respect to one node (two variables in a variable vector) here, because the wrong connections only related to node *k* are checked in this step.

In the crossover procedure of the original DE, the value *CR* is modified by considering only randomness in the search. In the new crossover, a search direction has been given. This will accelerate the search, but may lead to difficulty in jumping out from local minimum. Therefore, two steps: "Alternative generation" and "Final selection" have been added in the next section.

E. Flowchart of the proposed algorithm

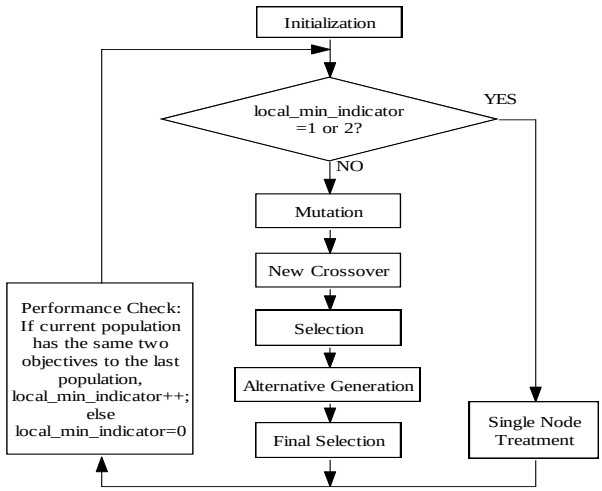

Figure 10 Flowchart of modified DE algorithm

An overall flowchart of the modified DE algorithm is shown in Figure 10. One important item in the algorithm is an indicator called 'local_minimum_indicator'. It is used to keep track of the progress during the iterations. The value of this indicator would be incremented whenever there is no change to the performance with respect to the two objectives ("*wrong_connection_count*" and "*wrong_distance*") obtained by the best individual in the population. It is initialized to 0 at the beginning. The initial population contains many individuals, which greatly affect the convergence of the proposed algorithm. If these initial values are totally random numbers, the proposed algorithm is hard to be convergent even after thousands of iterations. Therefore, reasonable initial values are needed. As we known from the introduction section of this paper, although not accurate, MDS which only considers connections can give a roughly estimation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

of the nodes' positions. Individuals of the initial population are generated based on the MDS result. Each individual is a variation of the MDS result. MDS is utilized as the initial value because it is quick and easy to obtain. Furthermore, compared with other methods only considering connections, MDS estimates nodes to be clustered, and therefore have the nature to produce less local minimum if being set as initial value [16]. "Single node treatment" is used to treat local minimum, which will be introduced in the next section. The details on the procedures of the modified DE algorithm are as follows:

Step 1: Mutation: for each individual $x_{i,G}$ in the current population *T*, use (1) to generate *v*, where λ and F are set as 0.8 and 0.9.

Step 2: New Crossover: generate u from v and $x_{i,G}$ as equation (4).

Step 3: Selection: generate $x_{i,G+1}$ by the function (5)

$$x_{i,G+1}=\begin{cases} u & \text{if } case1: \\ & wrong\_connection\_count(u) < \\ & wrong\_connection\_count(x_{i,G}) \\ & \text{or } case2: \\ & wrong\_connection\_count(u) = \\ & wrong\_connection\_count(x_{i,G}) \\ & and\ wrong\_distance(u) < \\ & wrong\_distance(x_{i,G}) \\ x_{i,G} & otherwise \end{cases} \quad (5)$$

where *wrong_connection_count* and *wrong_distance* are functions with respect to one variable vector. The last element (the $2m+2n+1$th element) of $x_{i,G+1}$, which represents the value of range is $u_{2m+2n+1}$

Step 4: Alternative Generation: an alternative individual $x\_new_{i,G+1}$ is then generated based on adding some randomness to the current $x_{i,G+1}$. The idea is to generate an alternate coordinate for each node coordinate in $x_{i,G}$ as shown in Figure 11. The angle is random but the radius would depend on the number of wrong connections. If the number of wrong connections is large, the newly generated alternate coordinate would be further away.

Let $x_{i,G+1}=([x_{2m+1}\ x_{2m+2}],[x_{2m+3}\ x_{2m+4}],...,[x_{2m+2n-1}\ x_{2m+2n}],x_{2m+2n+1})^T$

$$x\_new_{i,G+1}=([x\_new_{2m+1}\ x\_new_{2m+2}],[x\_new_{2m+3}\ x\_new_{2m+4}],...,$$
$$[x\_new_{2m+2n-1}\ x\_new_{2m+2n}],x\_new_{2m+2n+1})^T$$

The new node coordinates (as a pair of variables) are

$$[x\_new_{2k-1}\ x\_new_{2k}]=[x_{2k-1}\ x_{2k}]+radius*[\cos\alpha\ \sin\alpha] \quad (6)$$

where

$radius=rand(0-1)*wrong\_connection\_count([x_{2k-1}\ x_{2k}])/5$; 5 is used to ensure the distance between the new node and the original node is less than two twice of the node density. α is a random value from 0 to 2π; k is the index of the unknown nodes, $k=m+1,m+2,\ldots,m+n$.

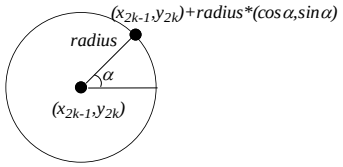

Figure 11 An alternative individual is generated

Still, the last element of $x\_new_{i,G+1}$ is set to be the same to that of $x_{i,G+1}$, which means the range value keeps unchanged in this step.

Step 5: Final Selection: the new generation of variables obtained at the end of the current iteration would depend on $x_{i,G+1}$ and $x\_new_{i,G+1}$. In a typical situation (i.e. when local_min_indicator = 0), the new generation is formed by choosing the best 50% from both $x_{i,G+1}$ and $x\_new_{i,G+1}$. However, when in a special situation encountering local minimum, and with no improvement in convergence (i.e. when local_min_indicator ≧ 3) even after using a special procedure called "single node treatment" (to be discussed in the next subsection), the following selection would be carried out:

- All the individuals in $x_{i,G+1}$ and $x\_new_{i,G+1}$ would be ranked according to the *wrong_connection_count*.
- The first 50% of the new generation is obtained from adding randomness to the top 50% of $x_{i,G+1}$, in which each node coordinate would be added with a value of 1 or -1 randomly. The variation of this value is designed to be less than the length of the network area divided by the square root of the number of nodes.
- The next 25% of the new generation is obtained from 25% to 50% of $x_{i,G+1}$. The idea is to discard the top 25% of $x_{i,G+1}$ deliberately so as to jump out from the local minimum.
- The final 25% of the new generation is obtained from the top 25% of $x\_new_{i,G+1}$.

Three parts provide necessary population in the next generation. The first part is the fundamental population which should take the majority. The second part is the 'radical' population with more randomness. The third part is used to enhance the weight of the 'typical' population. There is no an optimal proportion setting of the three parts, but the setting should fulfill that the first part should take at least 50% while the last two parts should not be too small.

F. Single Node Treatment

Two reasons contribute to the convergence of the search for coordinates. The first is the new crossover step, which introduces a direction during crossover. The second is "alternative generation" and "final selection", which introduce randomness in the population. After using these two effective methods, fast convergence is observed in the search for the nodes' positions. However, they are not enough to reach the final result as there may still be some local minimums near to the final answer. In other words, most nodes are localized, while only a few nodes produce wrong connections. In this section, a newly developed procedure called "single node treatment" will be described to help jump out from local minimums. The method is motivated from the second characteristic discussed in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

section III.C. When a wrong connection is encountered, two questions need to be answered:

- How far should a node be moved?
- Which node should be moved?

To answer the first question, the proposed algorithm would move a node for a distance equals to the absolute difference between the range value $\hat{R}$ (in T) and $\hat{d}_{i,j} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$, which is the estimated distance between node i to node j. An example is shown in Figure 12, which is based on the wrong connections described in Figure 8.

There are three nodes under consideration. There is a disconnection between node i and k, but they are supposed to be connected. There is a connection between node i and j, but they are supposed to be disconnected. Hence, node i has two wrong connections and should be moved (see the first figure in Figure 12). The direction and the distance of the two movements are determined in this step. First, node i is moved towards node k (the second figure), and then moved away from node j (the third figure). However, in the second move, the disconnection between node i and k re-appears. In other words, the wrong connection between node i and node j is fixed, but the connection between node i and k is still wrong. It can be fixed in the next iteration by moving k, which is shown in the last figure.

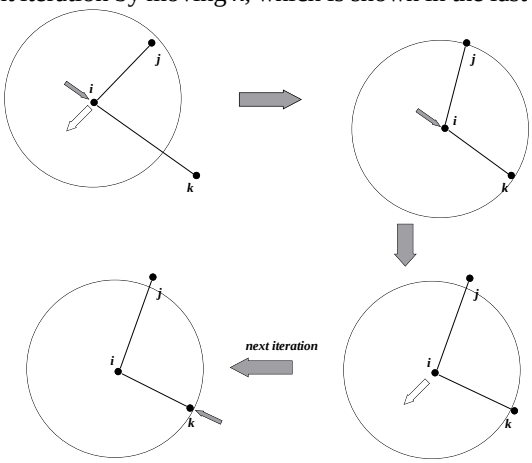

Figure 12 An example of the single node movement

Next, the second question regarding node movement is addressed. In most cases, the node with more wrong connections has a higher probability to be the wrong node than the node with less or zero wrong connections. Therefore, the node with more wrong connections should be chosen as the node to be moved. However, there are some situations where the node with less wrong connections should be moved. Figure 13 illustrates the above situation using an example. The blue lines and red lines represent the wrong connections. The blue lines denote the mistaken connections for disconnections, and the red lines denote the mistaken disconnection for connections. Triangles are the estimated positions, while circles are the real positions. We only focus on the nodes that involve wrong connections. In the first figure, node 50' has two wrong connections which are with node 2 and 12. Only node 50' can be moved because node 2 and 12 are both anchors. However, after node 50' steps forward to node 12, a new red line between it and node 89' appears (the middle figure). If the node with more wrong connections should be moved, then node 50' should be moved again rather than node 89', which would result in a situation very similar to before (the third figure). The above would result in endless movement of node 50', which cannot resolve the situation. Hence, it is not always correct that a node with more wrong connections should be moved. In most cases, we should move the node with more wrong connections, but we still need to leave some chances for moving the node with less wrong connections to get rid of the endless movement case in Figure 13. Therefore, the chance of moving the node with more wrong connections should be larger than 1/2, but not too close to 1. In the proposed algorithm, this chance is assigned to be 2/3. That means there is still a 1/3 chance to move the node with less wrong connections.

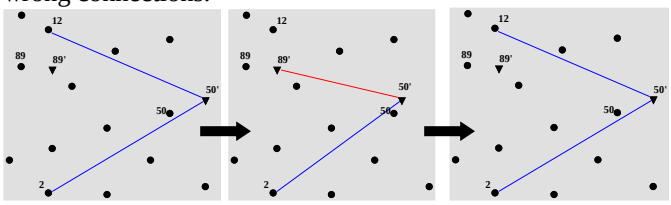

Figure 13 Illustration on node movement

IV. Simulation Results and Comparison

A. Comparison with other current convex-constraint methods

The simulations of this paper are conducted in different topologies (different graphs) to obtain convincing result. Different topologies introduce different location arrangement of the nodes. This arrangement in our topologies is random. Thus, in some topologies most nodes may be clustered, and in other topologies, nodes are nearly evenly distributed. The clustered network and sparse network may be found in even one topology, while most nodes are crowded in one half of this topology and there are fewer nodes in the other half. More topologies can produce higher chance to get different networks, and therefore show more convincing result.

The proposed algorithm is applied in a 100-node network, in which the first 20 nodes are anchors. All the nodes are placed in a 10 by 10 square region. The only known information is whether any two nodes are connected or not. This connection information is obtained when setting the range value to 2. Results are obtained based on ten different topologies and the algorithm is attempted for 10 times for each topology. Furthermore, because the evolutionary algorithm contains some randomness in computation, the result of our proposed algorithm is not unique. Therefore, in each topology, the algorithm is carried out for 10 times to obtain convincing result range. TABLE I shows the related parameters.

The accuracy of the estimation is evaluated by the difference between the estimated positions of unknown nodes and the positions of the corresponding unknown nodes when the problem is setup. The average error per unknown node (average error) is calculated by the formula next.

$$error\ \ per\ node = \frac{\sum_{i=m+1}^{m+n} \sqrt{(x_i - \hat{x}_i) + (y_i - \hat{y}_i)}}{n}$$

where $(\hat{x}_i, \hat{y}_i)$ is the estimated position of node i, (x_i, y_i) is the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

real position of the node *i*.

The average errors of the proposed algorithm for the 10 topologies are shown in the box plot of Figure 15. The mean values are around 0.3. It must be mentioned that there are two outliners in No.8 which are above 4.5. They are caused by two un-convergent results. In No.8, eight trials are convergent, the result of the other two trials are obtained when the maximum number of iteration is reached. They may be convergent when more generations are obtained, but for this algorithm, the convergence cannot be guaranteed. Out of the 100 different simulations using the DE with heuristics algorithm, these are the only two un-convergent cases. The estimated range values are very close to the real value of 2, except the two un-convergence cases which are shown as red crosses of No.8 in Figure 14. The comparison between our algorithm and other algorithms is shown in Figure 15, which indicates that the proposed algorithm has error around 25% of centroid localization, 30% of DV-Hop and 30% of MDS. It must be noted that MDS, DV-Hop and centroid method give a unique result in different trials. Hence, they are presented by dots, not boxplot. In summary, the proposed algorithm can improve the accuracy greatly. This sacrifices on the computational complexity. In the same simulation environment (a normal PC with MATLAB installed), MDS and DV-Hop use about 15s to finish one simulation, while the centroid localization spends much less. Our proposed algorithm uses about 10s to finish each iteration, and there are at least 30 iterations in a single simulation. Therefore, the proposed algorithm will spend about 5 minutes in a normal PC (Intel Core Quad CPU 3GHz, RAM 4GB) and be over 20 times slower than other algorithms such as MDS, centroid and DV-Hop shown in TABLE II. Generally speaking, a network with one hundred nodes can be accurately localized within five minutes by a normal PC using the proposed algorithm. However, this proposed algorithm can reach the best accuracy in the theory because it uses all known information. The computation time will be reduced with the development of computer hardware. It is useful in network with slower nodes, such as glacier monitoring, which needs accurate localization and have slow moving or quiet nodes.

TABLE I
PARAMETERS OF THE SENSOR NETWORK

Parameter	Value
m	20
n	80
R (range)	2
Boundary	[0,10]*[0,10]
Number of topology	10
Number of trials	10
Max number of iteration	400

TABLE II
COMPARISON OF COMPUTATIONAL TIME OF DIFFERENT ALGORITHMS

Algorithm	Computational time
MDS	15-20s
centroid	less than 1s
DV-Hop	15-20s
DE with heuristics	around 300s

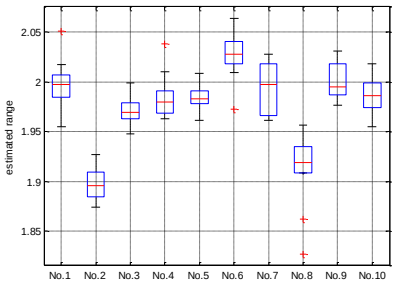

Figure 14 Estimated range when range=2

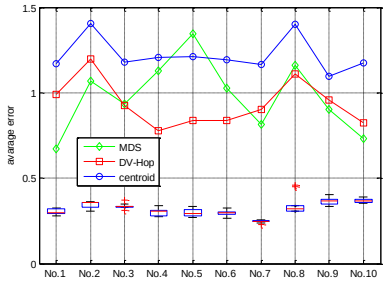

Figure 15 The average error compared with other algorithms' average error when range=2

Above comparison is implemented in the randomly distributed networks without any obstacles in the square area. More obvious accuracy improvement by the proposed algorithm can be found in the networks with big obstacle(s) such as C-shape networks. The simulation on below C-shape network is interesting, which significantly demonstrates the advantage of the proposed algorithm. To more straightforwardly show the accuracy improvement, we assume there is wall in the right half of a square area and randomly generate a map with 25 nodes as Figure 16. The prior known information contains the position of 8 anchors (marked as small green dots) and the connectivity information. The green lines are the connections among the nodes. The other 17 nodes' positions and the range value are needed to estimate. The estimated position should be near to the real positions of the 17 nodes (small circles in the figure). Real value of the range is 3. In the current implementation, the number of population (*NP*) is set to be 100. The algorithm makes use of the MDS result and its vibration as the initial population.

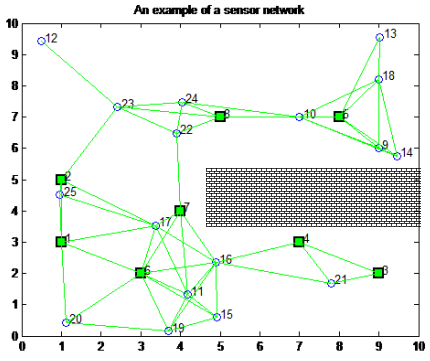

Figure 16 A sensor network with 25 nodes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

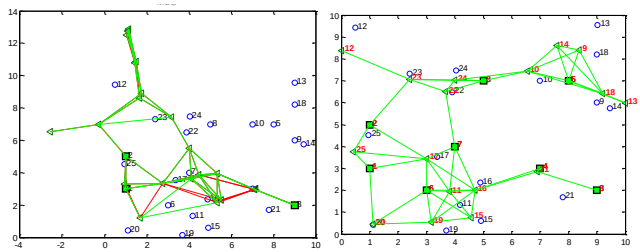

Figure 17 The estimation result of MDS (the left one) and the proposed algorithm(the right one)

For the estimation results, we use MDS to compare the proposed algorithm because MDS not only ignores disconnections, but also is the starting point of the iteration of our algorithm. The accuracy improvements can be clearly shown from the comparison. In Figure 17, the left figure shows MDS's results while the proposed algorithm's estimation is the right one. MDS shows very coarse estimation, most estimated nodes (small triangles) are far away from their real positions and almost failed in the C-shape network. Even with this coarse estimation as the starting point, our algorithm can reach reasonable solutions, in which the small triangles are the estimation of the proposed algorithm. The two objectives in different iteration can be shown in TABLE III. In the first iteration, the best individual among the first population generated from MDS result and random range value has *wrong_connecion_count*: 10, and *wrong_distance*: 32.962. The single node treatment procedure is applied in the 9th iteration, when detecting the last iteration with *local_min_indicator* being 1. The values of the two objectives, specially the value of *wrong_distance* decrease greatly due to the operation of single node treatment. Other iterations utilize the modified DE algorithm while the *local_min_indicator* is 0. In the last iteration, the two objectives are both 0, which means the estimated positions of the unknown nodes satisfy all the convex and non-convex constraints. The estimated range value is 2.906, which is very close to the real range value of 3.

TABLE III

TWO OBJECTIVES FOR EACH ITERATION OF THE PROPOSED ALGORITHM IN 25-NODE NETWORK

Iteration	WC	WD	Indicator	Range
1	22	32.961773	0	2.533473
2	19	31.422041	0	2.460921
3	16	27.675494	0	2.768528
4	14	24.637071	0	2.612727
5	12	19.094194	0	2.789369
6	10	27.307769	0	2.318219
7	9	20.679309	0	2.605094
8	9	20.679309	1	2.605094
9	7	3.358320	0	2.855567
10	5	2.486910	0	2.925609
11	4	1.258011	0	2.938039
12	2	0.325556	0	2.714406
13	2	0.055372	0	2.841072
14	0	0.000000	0	2.906340

B. Comparison with 'active-set' algorithm and PAES (non-convex methods)

In [15], a non-convex optimization algorithm called "active-set" has been used to solve the localization problem. It assumes that the range value is known, and the target is to minimize the first objective "*wrong_connection_count*". In [39], a two-objective evolutionary algorithm PAES is used to solve the connectivity-based localization problem. These two methods provide solutions for general non-convex optimization problem. In this section, the modified DE with heuristics algorithm is compared with those two methods.

"Active-set" algorithm does not need gradient in the objective function, and its constraints can be nonlinear and non-convex. It is implemented by Matlab and uses the result of semi-definite programming (SDP) as a starting point. The implementation of PAES can be found from jMetal[40]. PAES may represent the simplest possible nontrivial algorithm capable of generating diverse solutions in the Pareto optimal set. The simplest form: (1+1) evolution strategy, which is applied in this paper, employs local search but uses a reference archive of previously found solutions in order to identify the approximate dominance ranking of the current and candidate solution vectors [41]. PAES comprises three parts: the candidate solution generator, the candidate solution acceptance function, and the non-dominated-solution (NDS) archive. The candidate solution generator is similar to simple random mutation hill-climbing, but prefers the less crowded solutions in order to keep diversity preservation. It maintains a single current solution and, at each iteration, produces a single new candidate via random mutation. The detail information of 'active-set' and (1+1)-PAES algorithm can be found in [15] and [41].

The simulation is under the No.1 topology of the section A, with a range of 1.5. Every algorithm (except "active-set") is carried out for 30 trials and the results are drawn as boxplots. The error per node of the algorithms is displayed in Figure 18. The "active-set" algorithm has uniform result in different trials, and so it is displayed as a short red line. The performance of PAES is better than "active-set". Modified DE with heuristics can give better accuracy than PAES even if the range value is assumed unknown. It must be mentioned that no convergence (i.e. no results) are obtained by the "active-set" and PAES algorithm if the range value is assumed unknown. In addition, there is no convergence if the starting points are random.

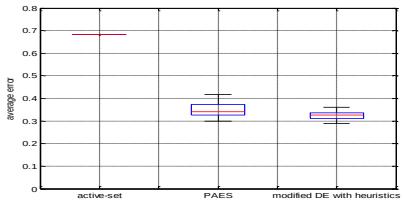

Figure 18 Comparison of accuracy of the non-convex algorithms

The computational complexity of our proposed algorithm is compared with PAES which is also an evolutionary algorithm. Here, we analyze their time complexity in terms of the number of computation in meeting the two objectives. The two objectives **with respect to one node** (i.e. two variables) can be found in the description of the "New Crossover" (subsection C in section III). Its computational complexity is *nt*, which is *O(n)*, where *n* is the number of nodes, *t* is time used to compute distance between two nodes and compare it with range value. The two objectives **with respect to one individual** (i.e. all unknown nodes together with the range value) are given in Figure 6 and Figure 7. Its computational complexity is n^2t. which is $O(n^2)$. It's because all possible pairs of nodes needs to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

be evaluated. For each individual in one generation, during the step of "New Crossover", every individual must compute the two objectives with respect to each node, but this computation should be on *n* nodes. So, the complexity in "New Crossover" is *n*nt*. The steps of "Selection" and "Final Selection" (Step 3 and 5 in subsection III.C) both compute the two objectives with respect to one individual, and their computational complexity is $2* n^2t$. Therefore, together with the computational complexity of "New Crossover", the complexity of each individual in one generation is $3* n^2t$. Assume that g_1 generations for population *NP* are needed to be convergent, the total computational complexity is $3\, g_1\, NP\, n^2t$. On the other hand, the (1+1) PAES's complexity is $g_2\, a\, n^2t$, where g_2 is the average number of generations needed for convergence, and *a* is the archive size [41]. In the simulation, *a*=100, and *NP*=100. It is observed that g_1 is around 30 and g_2 is around 100000. Therefore, the complexity of the modified DE with heuristics is nearly 0.1% of that of PAES. Considering that the current PAES algorithm has the range value assumed to be known while our algorithm assumes range value to be unknown, the reduction in computational complexity of the proposed algorithm compared with PAES is more obvious.

C. Results on the Scalability of the Algorithm

The performance of this algorithm for different sizes of networks is evaluated in the next simulation. In section A, the simulation is on a 100-node network in a 10 by 10 square region, and the communication range is 2. In this section, with the same settings, four different sizes of networks (the number of node are changed to 50, 75, 125, and 150) are tested to compare with the result of the 100-node network. For the comparison to be meaningful, the communication range has to be changed in order to keep the same connection degree. The number of nodes is changed to 50, 75, 125, and 150, while each node is connected to 10 nodes on average (which is the connection degree). These four networks are all with anchor ratio of 20%.

To achieve similar "connection degree" in the former network (100 nodes, range=2), the ranges for different cases of the networks are shown in TABLE IV. The ranges of those cases are computed based on that the square of range should be inversely proportional to the number of nodes [42].

TABLE IV
THE PARAMETERS OF THE SIMULATIONS ON DIFFERENT NETWORK SCALES

Number of nodes (q)	Range (r)	Degree	Anchor ratio
50	$2\sqrt{2}$	9.56	20%
75	$4/\sqrt{3}$	10.133	20%
100	2	10.04	20%
125	$4/\sqrt{5}$	10.608	20%
150	$4/\sqrt{6}$	10.84	20%

TABLE V
COMPARISON OF THE SETUP RANGE AND THE ESTIMATED RANGE

Number of nodes	Range (in TABLE III)	Mean value of estimated range (in Figure 20)
50	2.8284	3.1038
75	2.3094	2.3265
100	2	1.9972
125	1.7889	1.7554
150	1.6330	1.6441

The estimated average error for different scales of network is shown in Figure 19. The error becomes smaller when the scale of the network is increased. It can be explained by the fact that within the same region, the network with more nodes would lead to more constraints and this can improve the location accuracy. The estimated communication range is shown in Figure 20. The mean values of the estimated range are compared with the setup (true) range values in TABLE V. It is noticed that only the error of the estimated range for the 50-node network is a little bit larger than the other cases. It may be caused by having less constraints in the 50-node network.

The next simulation result shows the result of the proposed algorithm for different anchors ratios. The range is set to 2, and the total number of nodes is 100. The number of anchors varies from 10, 20, 30, 40 to 50. Each set of simulation is carried out for 10 times, and all of them are based on the 1st topology (No.1). The error per node (average error) is drawn as boxplots in Figure 22. When these boxplots in Figure 19 are compared with the average error of MDS, DV-hop, and centroid method, the improvement of our proposed algorithm (modified DE) is clearly shown as Figure 21. It can be observed that the error is decreased when the anchor ratio is increased. The result on the estimated range is shown in Figure 23. The final results are all close to the actual range value of 2. In Figure 24, it can be seen that the iteration time of the simulations for different anchor ratios is decreased when the anchor ratio is increased. This is not only because the number of parameters become less, but also because the number of constraints decreases. They both make the optimization less complex, which is the reason that less iterations are used.

The simulations are conducted in networks with 100 to 150 nodes due to the computational complexity. For networks with more nodes, the application of our proposed algorithm can be implemented by the distributed method described in [9, 43-45]. The distributed manner calculates the unknown sensors' position by using their neighbors' locations, not all the nodes. The procedure includes: divide, calculate, and stitch. A graph is first divided into several sub-graphs, and then the position of the unknown nodes are found for every sub-graph separately, and then the pieces of sub-graphs are stitched together [46]. The public nodes of two nearby sub-graphs are used for the foundation in the stitching process. Networks with several hundreds of nodes can be tackled by being divided into several sub-networks, and computation cost on each sub-network is acceptable using our proposed algorithm.

The convergence cannot be guaranteed. This is the nature of DE algorithm. However, it is a good algorithm based on two reasons. The first is the chance to be convergent is 99% from all simulations in our paper. From the simulations, within 400 iterations, only two un-convergent cases appear in 200 trials. The probability to encounter un-convergence is just 1% within 400 iterations. This percentage will become lower while more iterations are conducted. The second is the un-convergent cases still give acceptable estimations of positions. The results of un-convergent cases (the two outliers of No.8 in Figure 15) are much better than other algorithms, and are comparable with the convergent results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

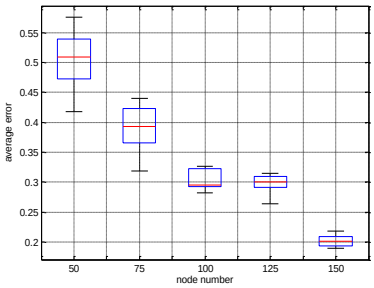

Figure 19 The average error in different scales of network

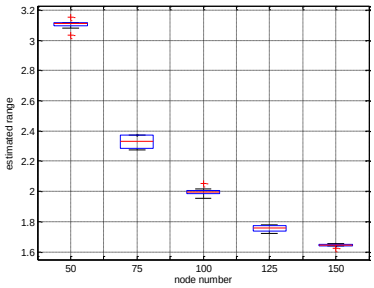

Figure 20 The estimated range in different scale of network

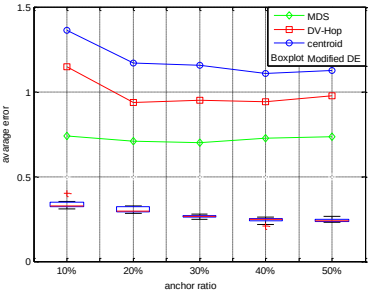

Figure 21 Comparison of error with different anchor ratios

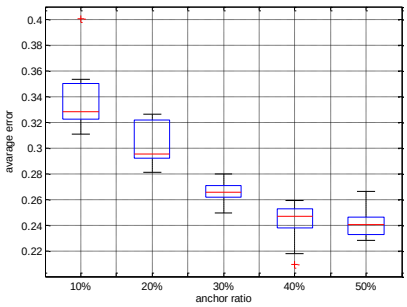

Figure 22 The average error of an 100-node problem with different anchor ratios (only to show the boxplots in Figure 21)

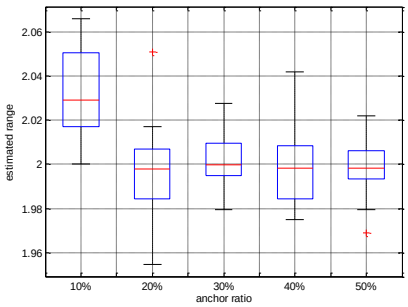

Figure 23 The estimated range for different anchor ratios

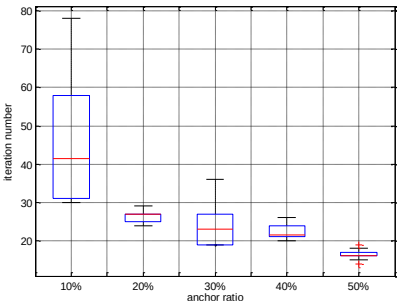

Figure 24 The iteration number for different anchor ratios

I. Conclusion

In this paper, a revised formulation of the connectivity-based localization problem is proposed which would require the search algorithm to find the sensor range as well as the location of the unknown nodes. The communication range of the sensor node is not assumed to be known a priori. A modified differential evolution (DE) algorithm for connectivity-based sensor network localization has been developed to solve this problem. The two objectives are on minimizing the violated constraints and the amount of the violation. The localization problem has two particular characteristics when compared with other common optimization problem. The first characteristic is on constructing a "new crossover" step of the modified DE. It gives a direction of search, but loses some randomness which is necessary of DE. Therefore, another step "alternative generation" is added to produce more randomness. "Final selection" selects the best individuals from the population produced by "new crossover" and "alternative generation". The second characteristic is used when the modified DE encounters a local minimum. This is formulated as "single node treatment" to correct the violated connections and disconnections for each node-pair with these violations. This treatment can make the iteration of the modified DE to jump out from the local minimum quickly. In the simulations, the modified DE with heuristics obtained much better accuracy. The error of this algorithm is only 20% of the error of former algorithms. In our work, only two out of more than two hundred simulation cases are un-convergent. The algorithm is also compared with general optimization algorithms such as "active-set" algorithm and PAES. The computational time, the performance on different anchor ratio, and the scalability of this algorithm are also discussed based on many simulation results.

Sensor network localization is a NP-hard problem with huge computational complexity. Non-convex constraints introduced by disconnections are much difficult to solve than convex constraints, and thus are always avoided in tackling the localization problem. However, this has led to coarse accuracy and a lot of useful information has been wasted. From the result of this paper, it is shown that the non-convex constraints can be handled when using the proposed algorithm.

References

[1] T. Eren, O. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, *et al.*, "Rigidity, computation, and randomization in network localization," in *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, 2004, pp. 2673-2684.

[2] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, pp. 28-34, 2000.

[3] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," presented at the the 9th Annual International Conference on Mobile Computing and Networking (MobiCom 2003), San Diego, CA, USA, 2003.

[4] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," presented at the the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2003), Annapolis, Maryland, USA, 2003.

[5] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," *Kluwer journal of Telecommunication Systems*, pp. 267–280, 2003.

[6] L. Doherty, K. S. J. Pister, and L. El Ghaoui, "Convex position estimation in wireless sensor networks," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, 2001, pp. 1655-1663.

[7] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Transactions on Sensor Networks*, vol. 2, pp. 188-220, 2006.

[8] (2015). *http://en.wikipedia.org/wiki/Multidimensional_scaling*.

[9] Y. Shang and W. Ruml, "Improved MDS-based localization," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, 2004, pp. 2640-2651.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2409319, IEEE Transactions on Vehicular Technology

[10] S. Minhan, C. Wook, and C. Hyunseung, "A cluster-based MDS scheme for range-free localization in wireless sensor networks," in *Proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2010)*, 2010, pp. 42-47.

[11] (2015). *http://en.wikipedia.org/wiki/Singular_value_decomposition*.

[12] J. Seung-Hwan and Y. Sang-Jo, "Improved positioning scheme based on DV-hop for wireless sensor network," in *the 9th International Symposium on Communications and Information Technology (ISCIT 2009)*, 2009, pp. 69-74.

[13] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Anchor-Based Localization via Interval Analysis for Mobile Ad-Hoc Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 57, pp. 3226-3239, 2009.

[14] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2001)*, 2001, pp. 2926-2931.

[15] D. Qiao and G. K. H. Pang, "Solutions for connectivity-based sensor network localization," in *Proceedings of IEEE International Conference on Mechatronics and Automation (ICMA'11)*, 2011, pp. 1056-1062.

[16] D. Qiao and G. K. H. Pang, "Accuracy improvement of connectivity-based sensor network localization," presented at the the 25th Canadian Conference on Electrical and Computer Engineering (CCECE'12), 2012.

[17] M. Terwilliger, A. Gupta, A. Khokhar, and G. Greenwood, "Localization using evolution strategies in sensornets," in *the IEEE Congress on Evolutionary Computation*, 2005, pp. 322-327.

[18] Q. G. Zhang, J. H. Wang, C. Jin, J. M. Ye, C. L. Ma, and W. Zhang, "Genetic algorithm based wireless sensor network localization," in *Proceedings of the 4th International Conference on Natural Computation (ICNC 2008)* 2008, pp. 608-613.

[19] V. Tam, K. Y. Cheng, and K. S. Lui, "Using micro-genetic algorithms to improve localization in wireless sensor networks," *Journal of Communications*, vol. 1(4), 2006.

[20] D. Manjarres, J. Del Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, and R. Lopez-Valcarce, "A novel heuristic approach for distance- and connectivity-based multihop node localization in wireless sensor networks," *Soft Computing*, vol. 17, pp. 17-28, 2011.

[21] J. Senshan, S. Kam-Fung, Z. Zirui, A. C. So, and Y. Yinyu, "Beyond convex relaxation: A polynomial-time non-convex optimization approach to network localization," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 2499-2507.

[22] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[23] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*: Springer, 2005.

[24] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 4-31, 2011.

[25] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 398-417, 2009.

[26] P. Rocca, G. Oliveri, and A. Massa, "Differential evolution as applied to electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 53, pp. 38-49, 2011.

[27] H. A. Hejazi, H. R. Mohabati, S. H. Hosseinian, and M. Abedi, "Differential evolution algorithm for security-constrained energy and reserve optimization considering credible contingencies," *IEEE Transactions on Power Systems*, vol. 26, pp. 1145-1155.

[28] C. Cheng-Hung, L. Cheng-Jian, and L. Chin-Teng, "Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, pp. 459-473, 2009.

[29] G. W. Greenwood, "Using differential evolution for a subclass of graph theory problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 1190-1192, 2009.

[30] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, pp. 266-277, 2001.

[31] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 646-657, 2006.

[32] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *The IEEE Congress on Evolutionary Computation*, 2005, pp. 506-513.

[33] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," presented at the Proceedings of the 8nd international conference on soft computing (MENDEL 2002), 2002.

[34] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, pp. 448-462, 2005.

[35] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, pp. 105-129, 2003.

[36] Z. Jingqiao and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945-958, 2009.

[37] D. P. K.Zielinski, and R. Laur, "Run time analysis regarding stopping criteria for differential evolution and particle swarm optimization," *Proceeding of the 1st iternational conference of Exp./Process/System Modelling/Simulation/Optimization*, , 2005.

[38] F. Xue, A. C. Sanderson, and R. J. Graves, "Modeling and convergence analysis of a continuous multi-objective differential evolution algorithm," in *The IEEE Congress on Evolutionary Computation*, 2005, pp. 228-235 Vol.1.

[39] D. Qiao and G. K. H. Pang, "Evolutionary approach on connectivity-based sensor network localization," *Applied Soft Computing*, vol. 22, pp. 36-46, 2014.

[40] (2015). *http://jmetal.sourceforge.net/*.

[41] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, pp. 149-172, 2000.

[42] D. Qiao, "Solutions for Wireless Sensor Network Localization," Doctor of Philosophy, Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, 2012.

[43] L. Zhang, L. Liu, C. Gotsman, and S. J. Gortler, "An as-rigid-as-possible approach to sensor network localization," *ACM Transactions on. Sensor Networks*, vol. 6, pp. 1-21, 2010.

[44] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *Proceedings of the 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, 2004, pp. 2652-2661.

[45] M. Cucuringu, Y. Lipman, and A. Singer, "Sensor network localization by eigenvector synchronization over the Euclidean group," *ACM Transactions on Sensor Networks* In press 2011.

[46] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, III, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 22, pp. 54-69, 2005.

Dapeng Qiao (S'11) received the B.Eng. degree in electronic engineering and information science from Shan Dong University, Jinan, China, in 2005, the Master degree from Harbin Institute of Technology, Harbin, China, in 2007, and the Ph.D. degree from The University of Hong Kong, Hong Kong, in 2012. He is currently an assistant professor in the School of Automation, Beijing Institute of Technology, China. His current research interests include sensor network localization, motion control, and optimization with applications in sensor networks and control theory.

Grantham K. H. Pang (S'84–M'86–SM'01) obtained his Ph.D. degree from the University of Cambridge in 1986. He was with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, from 1986 to 1996 and then joined The University of Hong Kong. Since 1988, he has published over 70 journal papers and 120 international conference papers. He has also obtained five U.S. patents, one European patent and one Chinese patent. His research interests include bio-medical informatics, visual surveillance, machine vision for surface defect detection, optical communications, logistics, intelligent control and expert systems