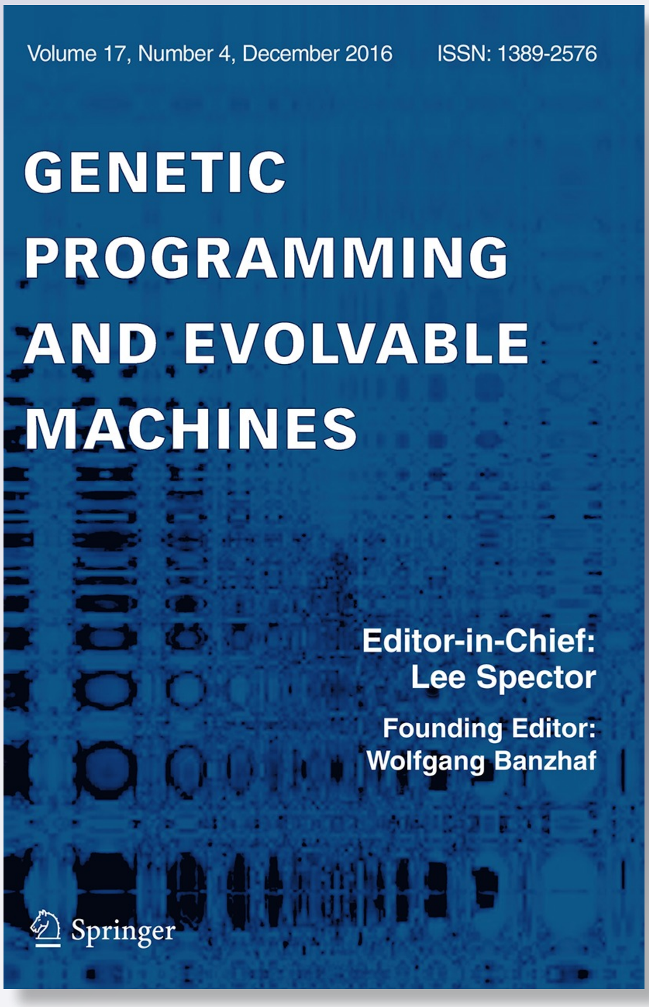
A two-objective memetic approach for the node localization problem in wireless sensor networks

Mahdi Aziz, Mohammad-H Tayarani-N & Mohammad R. Meybodi

Genetic Programming and Evolvable Machines

ISSN 1389-2576
Volume 17
Number 4

Genet Program Evolvable Mach (2016)
17:321-358
DOI 10.1007/s10710-016-9274-8


Volume 17, Number 4, December 2016
ISSN: 1389-2576
GENETIC PROGRAMMING AND EVOLVABLE MACHINES
Editor-in-Chief: Lee Spector
Founding Editor: Wolfgang Banzhaf
Springer

Genet Program Evolvable Mach (2016) 17:321–358
DOI 10.1007/s10710-016-9274-8

A two-objective memetic approach for the node localization problem in wireless sensor networks

Mahdi Aziz[1] · Mohammad-H Tayarani-N[2] · Mohammad R. Meybodi[1]

Received: 18 November 2015 / Revised: 15 May 2016 / Published online: 28 July 2016

Abstract Wireless sensor networks (WSNs) are emerging as an efficient way to sense the physical phenomenon without the need of wired links and spending huge money on sensor devices. In WSNs, finding the accurate locations of sensor nodes is essential since the location inaccuracy makes the collected data fruitless. In this paper, we propose a two-objective memetic approach called the Three Phase Memetic Approach that finds the locations of sensor nodes with high accuracy. The proposed algorithm is composed of three operators (phases). The first phase, which is a combination of three node-estimating approaches, is used to provide good starting locations for sensor nodes. The second and third phases are then utilized for mitigating the localization errors in the first operator. To test the proposed algorithm, we compare it with the simulated annealing-based localization algorithm, genetic algorithm-based localization, Particle Swarm Optimization-based Localization algorithm, trilateration-based simulated annealing algorithm, imperialist competitive algorithm and Pareto Archived Evolution Strategy on ten randomly created and four specific network topologies with four different values of transmission ranges. The comparisons indicate that the proposed algorithm outperforms the other algorithms in terms of the coordinate estimations of sensor nodes.

Keywords Memetic approach · Multi-trilateration technique · Self-adaptive local search · Localization problem · Wireless sensor networks

✉ Mahdi Aziz
mahdi_aziz@aut.ac.ir

[1] Computer Engineering and Information Technology Department, AmirKabir University of Technology, Tehran, Iran

[2] School of Computing Science, University of Glasgow, Glasgow, United Kingdom

1 Introduction

A wireless sensor network is a modern and advanced communication technology, used for sensing and controlling physical environments. WSNs are often composed of thousands or even hundreds of thousands of cheap and resource-limited sensor nodes. Although having first been designed for military purposes, they are now being used for search, rescue, target tracking, crisis management, monitoring of temperature, humidity control, vehicular movement and lightning condition [1, 2].

The localization problem has become one of the hottest topics in WSNs in recent years. This is because the localization information is useful for routing [3], coverage [4], boundary detection [5], clustering [6] and topology control [7]. This information is obtained through numerous tiny sensor nodes that are scattered across a network. If sensor nodes are used for monitoring a small area such as home or a retail store, it is easy to know the exact location of each sensor though manual configuration. However, if they are utilized for remote and vast environments such as a forest or a military zone, it is practically impossible to find out their exact location. Therefore, in such cases, we need to employ a systematic technique to find the positions of sensor nodes. To do so, we need to first compute the inter-node distances and then use that information to estimate the location of nodes. Algorithms for measuring the distances among sensor nodes can be categorized into two classes: the range-free algorithms and the range-based algorithms [8, 9]. The range-free algorithms use the connectivity-based knowledge, obtained by radio communication, such as neighborhood or hop count to measure distances among nodes [10]. These algorithms are comparatively simple and cheap, and they impose no requirement of additional hardware. However, they are less accurate than the range-based algorithms. The range-based algorithms, on the other hand, utilize range-based information such as location, range and angle to measure pair-wise distances of nodes [10, 11]. This information usually comes from Time of Arrival (TOA), Time Difference of Arrival (TDOA) and Received Signal Strength (RSS) measurement techniques [2, 9, 12, 13]. The former group are called the course-grained algorithms and the latter are called the fine-grained algorithms [9, 10].

In the TOA method, the time in which one signal travels from one node to another one is measured in order to estimate the distance between two neighboring nodes. Because of limited resolutions of timers, sonic and ultrasonic signals are usually utilized along with this approach. There is one challenge in using this method and that is all sensor nodes need to be synchronized. There have been several approaches for bypassing the time synchronization, including ping-pong style round trip messaging [14] and using two types of signals with different speeds like RF and ultrasonic [15]. This approach leads us to another technique based on time of arrival, called TDOA. In this approach, by calculating the time delay between the two nodes, the distance between them can be found. In the RSS, however, the distance between two nodes can be found by means of the strength of the signal. This means that the distance between transmitter and receiver can be calculated by considering the path loss of a particular medium [16]. We use RSS technique for calculating the inter-node distances as it can be easily configured, deployed and calibrated with low cost [13].

Using the data collected from the first stage (computing the inter-node distances), we can estimate the locations of sensor nodes. There are several approaches proposed to achieve this goal. First, we can find the exact location of sensor nodes through GPS receivers [17]. Despite having good accuracy, this method has several problems such as being prohibitively expensive in case of large WSNs and not working well in a place, surrounded by large buildings or underground sites [10]. Finally, GPS receivers can be embedded on a few nodes called anchor nodes or reference nodes, and other nodes called non-anchor nodes that can find their positions by computing their distances to those nodes [18]. Given this approach, algorithms for WSNs can be classified into three classes: multidimensional scaling, relaxation and stochastic techniques [9, 19]. The multidimensional scaling is a connectivity-based technique that uses distance-based information to estimate the relative position of nodes. This method works well on the RSS measurement. However, in this method, all sensor nodes need to be in the vicinity of each other so that each one can estimate its location through the relative positions of other nodes [10].

Considering the localization problem as a nonconvex optimization problem, several localization algorithms have been designed. The relaxation is a technique that was firstly suggested by Doherty [20]. In this approach, the localization problem is turned into a Semi-Definite Programming problem (SDP), which is seemingly easier to solve; then, geometric constraints among sensor nodes in the network are represented as linear matrix inequalities (LMIs). Finally, these LMIs join together to form a single semi-definite problem, which is tackled to create a bounding region for each node. Although promising in case of large-scale localization problems, this method has two major setbacks. First, it cannot provide high accuracy estimations. In some applications such as fire detection in forests, it is necessary to know the exact position of sensor nodes in order to deal with the problem immediately. Second, all geometrical constraints in the network cannot be formulated as LMIs and only constraints that form convex regions can be transformed to the LMIs [10]. Another type of numerical method for the localization problem is gradient decent algorithm that was recently proposed in [21] where it showed promising results if a noise-free distance measurement is used for calculating the inter-node distances. One advantage of this method is that it does not need the blind nodes to be in the convex set of the anchor nodes.

Stochastic techniques are another group of sensor-node-localizing approaches that were originally proposed by Kannan [22] where she employed simulated annealing (SA) algorithm to find the location of sensor nodes. SA is easy to implement and requires small amount of computational efforts, but when the flip ambiguity problem[1] occurs, the performance of the algorithm significantly declines. To cope with this, she then proposes a new version of SA (SAL), which uses a refinement phase for mitigating the effect of the problem [23]. Population-based algorithms are another group of optimizers, which use a set of search agents

[1] This problem happens when a set of nodes are collinear.

(solutions) to localize sensor nodes. For instance, reference [24] proposes a genetic algorithm that employs two genetic operators called the single-vertex-neighborhood mutation and the descend-based arithmetic crossover to localize sensor nodes effectively. Considering the localization problem as a two-objective optimization task, [19] proposes a two-objective evolutionary approach called the Pareto Archived Evolution Strategy (PAES), which attempts to address both the localization error and flip ambiguity problem simultaneously. They showed that the PAES algorithm can successfully deal with the node localization problem and, in comparison to the SAL algorithm, it can provide solutions with higher quality. There are other localization-based optimization algorithms that have recently come to surface such as Genetic and Backtracking Search Optimization Algorithms [25], Multiobjective Particle Swarm Optimization [26] and harmony search [27].

In general, optimization techniques for the fine-grained localization problem in WSNs can be categorized into two different groups. The first group uses only a stochastic technique such as SAL [23], Imperialist Competitive Algorithm (ICA) [28] or GAL [24]. These algorithms do not employ any approximation operators or priori information to provide good initial locations for sensor nodes. The second group, on the other hand, uses not only a stochastic optimizer such as simulated annealing or genetic algorithm but also an approximation stage such as multi-trilateration [13] or priori knowledge such as node-categorizing information[2] [19] to find better locations of sensor nodes.

Optimization algorithms, which have been applied to tackle fine-grained localization problem in WSNs, can also be classified into exploratory algorithms and exploitative algorithms. Exploratory algorithms [19, 24, 28] are the ones that employ longer-range operators such as crossover and mutation to search a longer radius of the problem's search space during the optimization process. Exploiting algorithms [22, 23], on the other hands, are the ones that apply shorter-range operators to search a shorter radius of that during the optimization process. Apart from these approaches, memetic approaches that are a combination of exploratory and exploiting algorithms take advantage of both remarkable exploitative and explorative capabilities. Making a trade-off between exploratory and exploitative capabilities, these paradigms (memetic approaches) can achieve high-quality solutions, compared to both exploratory and exploiting algorithms [29].

Memetic approaches have been designed to solve various optimization problems such as numerical functions [30], digital IIR filter design [29], partitioning problem [31] and constructing transductive discrete support vector machines [32]. However, to the best of our knowledge, they have not been applied to the localization problem in WSNs. Therefore, this encouraged us to design a new memetic approach to tackle fine-grained localization problem in WSNs. More specifically, the contribution of this paper is threefold. First, we show that memetic approaches can successfully be applied to the fined-grained localization problem in WSNs. The second contribution is based on the advantage of approximation approach to find good initial locations of sensor nodes. The third contribution lies in

[2] In this approach, first, all sensor nodes are categorized into three groups and later by using this information the algorithm can estimate non-anchor node locations with higher accuracy.

the benefit of the self-adaptive memetic operator of the algorithm to improve the location estimation of the sensor nodes. In order to test our algorithm, we compare it with several recent optimization-based localization algorithms, including SAL [23], ICA [28], GAL [24], PSO [33], TSA [13] and PAES [19]. Among these algorithms, some have specific merits; for example, PSO and SAL are fast and can be easily implemented, and TSA and PAES are more promising in terms of localization estimations.

The remainder of this paper is structured as follows. In Sect. 2, the fined grained localization problem is described. The proposed algorithm is introduced in Sect. 3. In Sect. 4, we compare the proposed algorithm with SAL [23], ICA [28], GAL [24], PSO [33], TSA [13] and PAES [19] on ten randomly created and four model-based created network topologies. Finally, the paper is concluded in Sect. 6.

2 Problem definition

In this section, we define the fine-grained localization problem in WSNs, concentrating on the system model, the evaluation of the proposed algorithm during the optimization process and the assessment of the performance of the algorithm after the optimization process.

2.1 System model

A wireless sensor network can be considered as a network that consists of anchor nodes and non-anchor nodes. In this network, anchor nodes are the ones that are fully aware of their positions. This knowledge comes from their GPS receivers or their individual records. The non-anchor nodes, on the other hand, are the ones, that do not know their positions. All sensor nodes have an equal connectivity range, r, and are distributed uniformly in a two-dimensional squared region with a range of $[0,1] \times [0,1] \subset R^2$. We use some measurement techniques introduced in Sect. 1 to estimate the inter-node distances $\tilde{d_{ij}}$. We assume that $\tilde{d_{ij}}$ is computed as,

$$\tilde{d_{ij}} = d_{ij} \times (1.0 + \gamma \times NoiseFactor), \tag{1}$$

where $\tilde{d_{ij}}$ and d_{ij} are the measured and the true distance between the i-th and j-th nodes, respectively and γ is a Gaussian noise, added to the distances because of the measurement noise. The mean and standard deviation of the Gaussian noise are equal to 0 and 1, respectively. We assume that the measurement errors are distributed uniformly across the network. We use a simple disk model that are typically used in the literature [13, 19, 22–24, 28] for network communication in that sensor nodes can communicate with each other as long as the actual distances among them are less than the communication range. For instance, node i can communicate with node j if $d_{ij} < r$.

2.2 Objective functions and performance evaluation

Considering the fine-grained localization problem in WSNs as one-objective optimization task, Kannan [22] proposed an optimization technique to estimate the positions of non-anchor nodes. This nonconvex optimization task is targeted at only the localization accuracy and it does not touch the flip ambiguity problem. Later, the authors proposed a new version of SA algorithm that addresses both the location accuracy and the flip ambiguity problem in two sequential stages. In another work, authors [19] proposed a two-objective optimization task that attempts to tackle both the localization accuracy and flip ambiguity problem simultaneously. In this paper, we use this two-objective function for evaluating the proposed algorithm during the optimization process. The first objective function CX is found as,

$$CX = \sum_{k=1}^{m} \sum_{j \in S_k} (\hat{d_{kj}} - \tilde{d_{kj}})^2 + \sum_{i=1}^{n} \sum_{j \in S_i} (\hat{d_{ij}} - \tilde{d_{ij}})^2, \tag{2}$$

where m and n are the number of anchor nodes and non-anchor nodes respectively, $\hat{d_{kj}}$ is the estimated distance between the anchor node k and the non-anchor node j, calculated as $\hat{d_{kj}} = \|a_k - \hat{x_j}\|$, $\hat{d_{ij}} = \|\hat{x_i} - \hat{x_j}\|$, where a_k is the real position of anchor node k and $\hat{x_i}$ and $\hat{x_j}$ are the estimated positions of nodes i,j. In Eq. 2, $\tilde{d_{ij}}$ represents the measured distance between non-anchor node i and j and $\tilde{d_{kj}}$ is the measured distance between anchor node k and non-anchor node j. As mentioned, the distances among nodes are measured through formula 1.

The second objective function CV attempts to minimize the connectivity constraints, violated by the current estimated locations of non-anchor nodes. It was argued [19] that the minimization of a violation of connectivity constraints helps the algorithm alleviate the flip ambiguity problem by placing nodes in their appropriate neighborhoods. The function is defined as,

$$CV = \sum_{i=1}^{n} \left(\sum_{j \in S_i} (\delta_{ij}) + \sum_{j \in \hat{S_i}} (1 - \delta ij) \right), \tag{3}$$

where δij=1 if $\hat{d_{ij}} > r$ and 0 if not.

In order to evaluate the performance of the proposed algorithm, we use the following metric that calculates the distance between the estimated and the real positions of non-anchor nodes in the network.

$$LE = \frac{1}{n} \sum_{i=1}^{n} \frac{(||\hat{x_i} - x_i||)^2}{r^2} \times 100, \tag{4}$$

where LE is the localization error and x_i is the real position of i-th non-anchor node.

3 The proposed algorithm

Evolutionary algorithms are powerful optimizers which use a set of candidate solutions to solve optimization problems. Although very effective, they require long time to find the exact position of the local optimum [34, 35]. Local search algorithms are usually capable of exploiting the local optimum in a faster way. In case of large-scale optimization problems, however, local search techniques tend to get trapped in a local optimum [36]. On both small and large scale optimization problems, on the other hand, memetic approaches can find promising regions in the search space in a shorter period of time by hybridizing the explorative ability of evolutionary approaches and exploitative capability of local search approaches [29, 30]. Here, we design a memetic approach that involves three memetic operators (phases) of which each is designed to fulfill a specific aim. In the first phase, to make good starting locations for non-anchor nodes, the algorithm uses an approximation procedure in which the positions of non-anchor nodes are estimated by repeatedly executing the trilateration [37], the two-node localization and rough localization procedures. The approximation process tends to facilitate the search process and thus decreasing the time budget required for the estimations of the coordinates of non-anchor nodes. In the second phase, to create a new solution, the algorithm uses an exponential crossover operator in the fashion of differential evolution [38]. In this operator, a new solution is created by randomly sampling genes from the current solution and a randomly selected solution of the Pareto optimal set. The Pareto set, *A*, is the set of solutions that are created during the optimization process and not dominated by the current solution *E* and by other members of the Pareto set. The second phase is called the exploration phase since attempting to explore a new region of the search space by combining the current solution with an optimal solution in the Pareto set. In the third phase, the algorithm utilizes a self-adaptive memetic operator which employs eight different directions with the objective of moving the solution toward a better area of the search space.

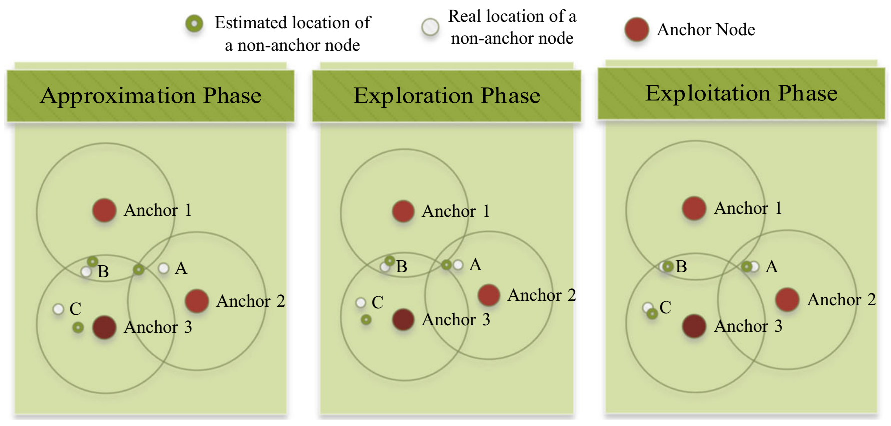


Fig. 1 An example of the process of the localization by the proposed algorithm

These directions include up, down, right, left, up left, up right, down left and down right directions.

An example of the proposed algorithm is presented in Fig. 1. In Fig. 1, three non-anchor nodes are shown where the non-anchor node A is initialized by the multi-trilateration operator, node B by the two-node localization and node C by the rough localization of the approximation stage. In the exploration phase, using the exponential crossover, the estimated locations are improved and in the exploitation phase, the locations are further enhanced.

Although our proposed algorithm and PAES [19] are similar in several aspects such as objective function and selection mechanism, (both use two-objective function for fitness evaluation during optimization process and Pareto-optimal-set-based selection mechanism for the best solution retention) they are different when it comes to the sensor node initialization and optimization processes. While PAES utilizes an initialization technique that categorizes sensor nodes based on their distances to anchor nodes, our approach adopts an approximation operator that contains multi-trilateration, two-node localization and rough localization, for the same reason. In Sect. 5, we test and compare our initialization approach with Vecchio's initialization approach on ten different network topologies and on two different transmission ranges.

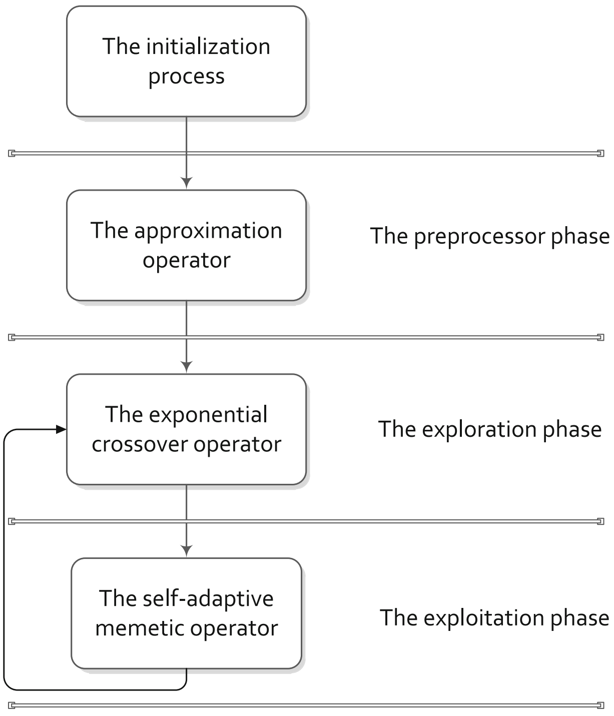

The initialization process
The approximation operator
The preprocessor phase
The exponential crossover operator
The exploration phase
The self-adaptive memetic operator
The exploitation phase

Fig. 2 The block diagram of the 3PMA

For the optimization process, unlike Vecchino's approach that employs evolution strategy, our algorithm uses a memetic approach, that includes exponential crossover and a new local search operator. We believe, as many papers suggested, that using memetic approach is more promising. In this case, evolution strategy like other evolutionary algorithms tends to show slower convergence rate than memetic approaches, thereby exploring the search space in a longer time.

The block diagram of the proposed algorithm is shown in Fig. 2.

Besides from the three main phases, initialization process is adopted where the probability matrix P, used in the exploitation phase and the parameters of the algorithm, is initialized. The initialization procedure is performed at the first lines of the algorithm as,

1. initialize A, M, I_r, p, β,d
2. initialize P as the probability matrix

The description of the initialization process is as below:

Step 1: In this step, A the coordinates of anchor nodes, M measured distances among all sensor nodes, I_r an inheritance factor used in the exploration phase, p the number of iterations spent to improve the position of each non-anchor node in the exploitation phase, β the distance-declining factor and d the initial move distance are initialized by the user before the optimization.

Step 2: The probability matrix used in the exploitation phase is initialized as,

$$P_{i,k} = 1/8, \tag{5}$$

for $i = 1, 2, \ldots, n, \quad k = 0, 2, \ldots, 8$. Here 8 is the number of neighborhoods for each non-anchor node.

In the proposed algorithm, a solution is denoted by a 2-dimensional array called, T, where n is the number of non-anchor nodes, and X and Y are the coordinates of non-anchor nodes (see Fig. 3).

The first phase includes a preprocessor memetic operator that attempts to estimate the initial positions of non-anchor nodes effectively and the last two phases, the exploration and exploitation phases, comprise two memetic operators, attempting to iteratively and sequentially optimize the approximate solution obtained at the first phase of the algorithm.

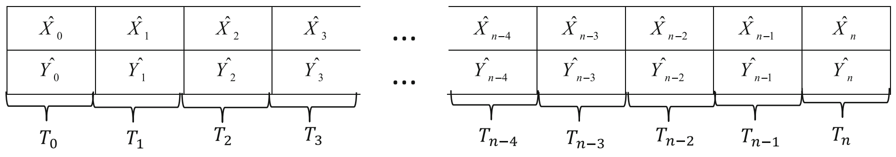


Fig. 3 The solution coding

3.1 The approximation phase

To provide good initial locations for non-anchor nodes, the algorithm utilizes an approximation procedure, which is described in the following steps.

The Approximation Phase

```
1. E = multiTrilateration(A,M)
   if there is at least one non-anchor node left in A
2. E = twoNodeLocalization(A,M)
   if there is at least one non-anchor node left in A
3. E = roughLocalization(A,M)
   End Phase
```

The describtion of the approximation phase is as below,

Step 1: the location of non-anchor nodes with three or more known neighboring nodes is estimated. First, all nodes are divided into two sets, the set of anchor nodes, *A*, and the set of non-anchor nodes, *F*. Then, the non-anchor nodes in *F* are localized by the trilateration technique[37]. Using the trilateration technique, each non-anchor node in F that has three neighboring nodes in A is localized and moved to set A. This movement is iterated until the location of all the non-anchor nodes, with at least three neighboring nodes in A, is found.

Step 2: This process is not performed unless there is at least one non-anchor node in *F* that has not been localized by means of the multiTrilateration technique. Like two-reference area-based localization, as described in [10], in the two-node localization process, the non-anchor nodes that have two known node neighbors, are localized. Using the locations of two neighboring nodes in A, the location of a non-anchor node in F is found. In other words, a non-anchor node in *F* can obtain its estimated location through the computation of the intersection of the overlapping coverage region and the selection of a random position within this region. Let c' be the selected non-anchor node and a and b denote its known node neighbors. The location of unknown node c' is randomly determined within the coverage region constructed by a and b.

Step 3: If there is at least one non-anchor node in *F* that has one known neighbor in A, its position can be estimated by its only one neighboring node. This method is similar to the single reference area estimation method described in [10]. In the single reference area estimation, the location of a node is estimated according to its neighboring reference node (a node that knows its exact location). Likewise, in this step, the position of the selected non-anchor node is randomly estimated within the range of its neighboring node in A.

It is evident that the last two steps do not calculate the positions of non-achor nodes, rather they determine where those nodes might be. These rough estimations increasingly limit our estimation about the locations of the non-anchor nodes, resulting in a more precise estimation about other adjacent nodes.

3.2 The exploration phase

During the optimization process, the algorithm may get trapped in a local optimum and thus fail to reach the better region in the search space. In such situation, the algorithm needs to be shored up by some exploration operators such as mutation and crossover operators [19]. Here, in order to help the algorithm escape from the local optimum and thus to improve its exploration ability, an exponential crossover in the form of conventional differential evolution is used. In the exponential crossover, first a solution called S is randomly selected from the set of non-dominated solutions (A_r). Then, genes from the current solution E are iteratively copied to the A, as long as the inheritance factor I_r is greater than a randomly created number in a range of [0, 1]. The following steps show how this procedure is performed.

```
The Exponential Crossover Procedure
1. Randomly select a solution called $S$ from the Pareto optimal set $A$
2. i = randint(0,n)
3. S.x[i] = E.x[i], S.y[i] = E.y[i]
4. while rand(0,1)< $I_r$ do
5.    S.x[i] = E.x[i], S.y[i] = E.y[i]
6.    i++
7.    if(i==n)
8.       i = 1
      end if
end while
9. if(S $\preceq$ E)
10.   replace(S,E)
11. else if(S$\preceq\succeq$E)
12.         addToParetoSet($E$) End Procedure
```

Step 1: a solution called S is randomly selected from the Pareto set of non-dominated solutions, A.

Step 2: an number is randomly created in the range of [0,n].

Step 4–7: In the 'while' loop, genes from the current solution E are iteratively sampled in order to randomly combine the current solution with a non-dominated solution. The 'while' loop is continued until the inheritance rate I_r is less than a randomly generated number between 0 and 1.

Step 9, 10: If the new solution S dominates E, E is replaced by it. Note that, E is dominated by S if S is better than or at least the same as the first or second objective function. The two-objective function used here is presented in Eqs. 2 and 3.

Step 11, 12: If the new solution is non-dominated, it is stored in the Pareto set A as a non-dominated solution.

3.3 The exploitation phase

The location of sensor nodes found in the previous phases (approximation and expolarion phases) can be further improved using the adaptive local search. We call this adaptive local search the exploitation stage as it exploits the search space further with the objective of finding better solutions. In order to show the performance of this stage, we measure the performance of the algorithm before and after the exploitation stage. The analysis is included in Sect. 5.

This phase not only attempts to lead the solution to a better area of the search space, but also creates some non-dominated solutions that are then used to generate a new solution in the exploration phase. The pseudo-code of this phase is described below,

```
The Exploration Phase
begin
1. Q = generatePermutation(n)
2. fōr all non-anchor nodes
3. Snode = Q(i)
4. adaptiveLocalSearch($A_r$,E,p)
   end for
5. d = d*$\beta$
End Phase
```

The description of the phase is presented below:

Step 1: To randomly select a non-anchor node and avoid choosing a node that has been selected before, a random permutation *Q* of the positions of non-anchor nodes is generated.

Step 2: For all non-anchor nodes, steps 3, 4 are taken.

Step 3: *i*-th element of the random permutation is assigned to *Snode*, the selected non-anchor node.

Step 4: In the adaptiveLocalSearch procedure, the location of *Snode* is attempted to be enhanced. To show how this procedure is performed, an example of it is represented in Fig. 4 where *p* is set to 10.

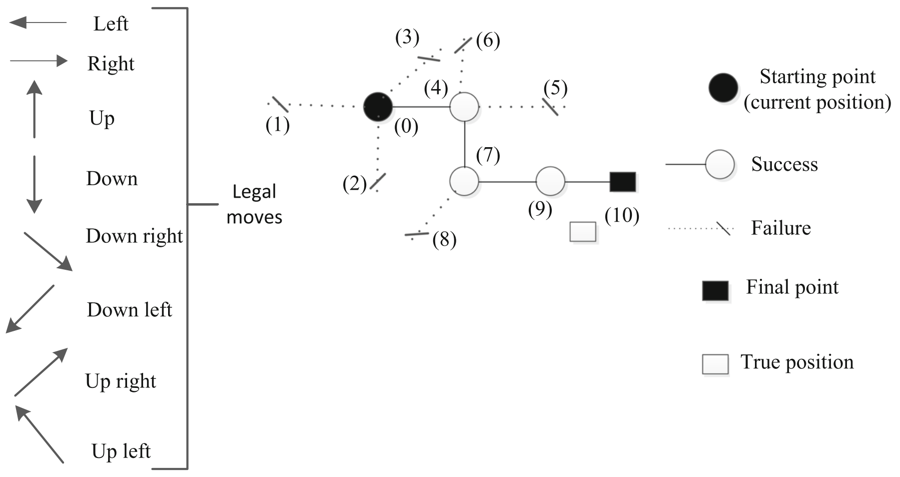


Fig. 4 An example of the adaptiveLocalSearch

As shown in Fig. 4, the 'adaptiveLocalSearch' adaptively tries to find the best position for each non-anchor node where at first it gives equal opportunity for each walk operator (walk operators are specified in the Fig. 4), and then depends on how much each walk operator obtains better fitness value, their chance of being selected in the next iteration would be improved. So, this process continues until the maximum number of iteration is reached.

As shown in Fig. 4, from (0) to (7), first four different directions and then three different ones are tested, where two of them are successful, resulting in actual moves. After that, because of the feedbacks received from both successful and unsuccessful moves, two successive successful steps are taken. Intuitively, since the algorithm does not have priori knowledge about the real position of the sensor node, it firstly tests several moves (mostly unsuccessful); however, after receiving the feedbacks, the probability of the affected directions are adjusted, either being increased or decreased depending on their success. The local search procedure is performed as,

```
Procedure adaptiveLocalSearch
4.1 for $j = 1$ to $p$
4.2    copyTo(E,N)
4.3    **d** = rankBasedSelection(P)
/* Walk Operator */
4.4    *Snode* is moved in one of directions determined by **d**
/* End of Walk Operator */
4.5.   update(P,**d**,Snode)
4.6    evaluateSolution(N)
4.7    if N$\preceq$ E
4.8        addToArchive(N,A)
4.9        replace(N,E)
4.10   else if E $\prec\succ$N
4.11       addToArchive(N,A)
       end if
       end for
```

Step 4.1: In the for loop, the location of the *Snode*, is iteratively enhanced. The number of iterations assigned for improving the position of the Snode is *p*, which is set by user.

Step 4.2: In order to keep the new solution, *N* intact from the bad moves, the current solution is copied into *N* at each iteration.

Step 4.3: To decide which direction of movement, **d**, should be selected for the walk operator, a rank-based selection procedure[39] with linear ranking model is used. In the rank-based selection, first, the selection probabilities of directions of the nodes are sorted in the descending order. Then, each of them is compared to a random number between [0, 1]. If greater, it is selected; otherwise, the next one is checked. The selected node is assigned to *Snode*.

Step 4.4: The position of *Snode* is changed according to **d**. More specifically, if $\mathbf{d} = 0$ (the 'right' direction is selected) *Snode* takes one step with the size of *d* toward the right. In a similar vein, if each of the $\mathbf{d}$ = (1,2,3,4,5,6,7) (the 'left', 'up',

’down’, ’up right’, ’down left’, ’down right’ and ’up left’ directions is selected), *Snode* takes one step toward the selected direction of movement.

Step 4.5: After the walk operator, the selection probability of *Snode* is updated as,

$$P_{Snode,\mathbf{d}} = (N_{cx} - E_{cx} + N_{cv} - E_{cv})/w, \tag{6}$$

where $P_{Snode,\mathbf{d}}$ is the selection probability of the **d**-th direction of Snode, N_{cx} and N_{cv} are the first and second objective function of N, E_{cx}, E_{cv} is the objective functions of E and w is a constant value used for adjusting the selection probabilities.

Step 4.6: The new solution, N, is evaluated using Eqs. 2 and 3 and then it takes the place of E if it is better than E in at least one objective function.

Step 4.7: If N dominates E, the two following steps are taken.

Step 4.8–9: N is added to A_r if dominated by no solutions in A_r, and E is replaced with N.

Step 4.10: If N does not dominate E and vice versa, the following step is taken.

Step 4.11: N is added to A_r, provided no solution in A_r is better than it.

Step 5: In order to increasingly refine the estimated positions of sensor nodes, the movement distance is decreased every time the exploration phase is called.

4 Simulation results

The aim of this section is to investigate how well the proposed algorithm is employed in solving the node localization problem in WSNs. In our simulations, first, we constructed 10 random network topologies named as TOP0–TOP9 and then

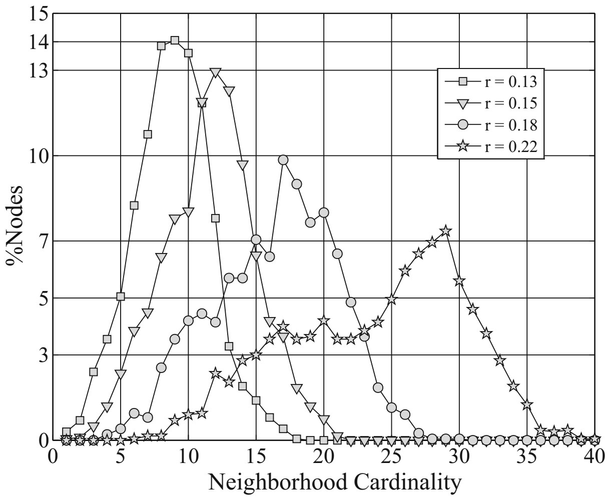


Fig. 5 The mean percentage of sensor nodes versus neighborhood cardinality for 4 different values of the communication range r

built four specific network topologies. In each network topology, we have set the value of transmission ranges of nodes to 0.13, 0.15, 0.18 and 0.22.

4.1 Comparison on random network topologies

In this subsection, we first examine the randomly generated network topologies in terms of node neighbor rate for each sensor node and anchor-node neighbor rate for each non-anchor node. Then, we perform a comparison between the proposed algorithm with the six existing optimization approaches on the network topologies studied. In these topologies, the noise factor *NF* is 0.1% and the number of anchor and non-anchor nodes are 20 and 180, respectively. The nodes are uniformly placed in a square region of $[0,1] \times [0,1] \subset R^2$.

4.1.1 Topology setup

Figure 5 demonstrates the mean percentage of network nodes (anchor and non-anchor nodes) against the number of neighboring nodes each with 10 random network topologies.

As shown in Fig. 5, the greater the communication range, the greater the neighborhood cardinality. For instance, for r = 0.13, about 13 % of the nodes have 10 adjacent nodes. Furthermore, it can also be observed that for r = 0.13 no node has more than 9 neighboring nodes, respectively.

Figure 6 demonstrates that the mean percentage of non-anchor nodes in the 10 random network topologies against the number of their adjacent anchor nodes for different values of *r*.

As shown in Fig. 6, there are numerous non-anchor nodes that have only one or even zero neighboring anchor node. For example, when $r = 0.13$, about 55 % of

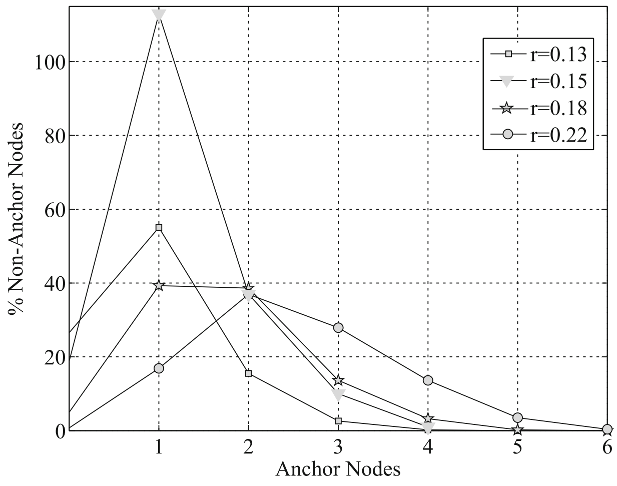


Fig. 6 The mean percentage of non-anchor nodes versus anchor nodes for different values of *r*

Table 1 Parameter configuration of the SAL, GAL, PSOL, TSA, ICA, PAES and 3PMA

Parameter	Value	Description
SAL		
T_0	0.1	The initial temperature
N_f	50,000	The number of function evaluations
P	10	The number of iterations in the inner loop
Q	2	The number of iterations in the outer loop
D_0	0.1	The initial move distance
α	0.80	The cooling factor
β	0.94	The distance-declining factor
GAL		
M_r	0.85	The mutation rate
I_r	0.75	The crossover rate
S_p	50	The size of population
M_I	1000	The maximum number of iterations
PSOL		
W	0.7	The mutation rate
C	1.494	The crossover rate
S_p	50	The size of population
M_I	1000	The maximum number of iterations
ICA		
N_I	8	The number of empires
N_C	50	The number of countries
M_I	1,000	The maximum number of iterations
PAES		
A_s	20	Archive size
N_r	5	Number of regions
N_f	50,000	The number of fitness evaluations
P_M	0.9	Node mutation probability
P_N	0.3	Node rigid translation probability
TSA		
T_0	0.1	The initial temperature
N_f	50,000	The number of function evaluations
P	4	The number of iterations in the inner loop
Q	$2 * n$	The number of iterations in the outer loop
D_0	0.1	The initial move distance
α	0.80	The cooling factor
β	0.94	The distance-reducing factor
μ	0.2	The parameter used for threshold value calculation
λ	0.1	The parameter used for threshold value calculation
γ	0.05	The parameter used for threshold value calculation

Table 1 continued

Parameter	Value	Description
3PMA		
A_s	20	Archive size
N_f	50,000	The number of fitness evaluations
I_f	0.94	Inheritance factor
β	0.99	The distance-declining factor
D_0	0.05	The initial move distance
P	10	The number of iterations for improving the position of each node
w	1000	The constant value used for adjusting the selection probabilities

non-anchor nodes have only one adjacent anchor node and about 27 % of non-anchor nodes have no anchor nodes nearby. This clearly indicates that solving the localization problem on the proposed network topologies is very demanding.

4.1.2 *Comparison against existing optimization techniques*

In order to test the proposed algorithm, we compare it with the SAL [22], GAL [24], PSO [33], TSA [13], ICA [28] and PAES [19] on ten randomly generated network topologies. The best parameters for the algorithms as recommended in [13, 19, 22, 24, 28, 33] are used in order to make sure that each algorithm reaches its best possible results. The parameters used are represented in Table 1.

To perform a fair comparison, the termination condition for all the algorithms is set to 50,000 function evaluations (FCs) and when no better solution has been reached for 100 iterations; that is, for the population-based algorithms (GAL, PSOL, ICA), the size of the population is set to 50 and the maximum number of iterations is set to 1000, and for the single-solution algorithms (SAL, TSA, PAES and 3PMA) the maximum number of iterations is set to 50,000. Moreover, due to the fact that we set a new termination condition for each algorithm, the other termination conditions are eliminated. For instance, for the SAL, T_f (the final temperature) is removed.

Table 2 summarizes the mean and the standard deviation of the *LE* values obtained by each algorithm over 20 trials for the ten randomly generated network topologies with four different values of communication ranges. The best results are typed in bold.

It can be observed that for $r = 0.13$ the 3PMA achieves the best LE values for all network topologies (except *TOP*4); after that, the PAES, TSA and SAL are the second to forth best algorithms. Furthermore, for $r = 0.15, 0.18$ and 0.22, the 3PMA maintains its superiority over the other algorithms and it outstrips all its competitors in all network topologies. It is also shown that as the value of connectivity range increases, the performance of the PAES and TSA significantly changes. For $r = 0.13$, the PAES is superior over the TSA on the 9 cases and for $r = 0.15$ on 7 cases. However, for $r = 0.18$, 0.22, the TSA is better than PAES on 8 and 10 cases, respectively. Intuitively, a combination of an approximation procedure such as the

Table 2 The mean and standard deviation of the *LE* values obtained by 7 optimization techniques over 20 runs on the ten network topologies and for 4 different values of communication ranges

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
$r = 0.13$										
3PMA										
Mean	69.22	33.95	23.96	70.51	40.68	105.31	124.23	26.55	29.71	29.14
Std	7.30	3.43	2.72	12.75	9.92	15.08	40.09	3.31	4.40	2.98
Skewness	0.70	0.03	−1.15	−0.84	1.32	0.40	−1.45	0.49	−1.54	−0.12
Kurtosis	2.13	2.63	3.21	3.75	4.36	2.02	3.18	2.21	3.78	1.39
TSA										
Mean	175.45	158.32	175.12	127.33	148.78	179.70	174.42	151.98	154.09	136.78
Std	6.45	12.25	14.63	16.17	24.57	31.18	8.97	15.31	12.05	14.86
Skewness	−0.08	0.33	−1.11	−0.52	0.26	−0.74	0.80	0.58	0.98	−0.24
Kurtosis	1.14	1.97	2.30	1.72	2.01	2.03	2.14	2.04	2.20	1.35
PEAS										
Mean	91.74	97.07	123.45	98.62	94.00	92.25	99.90	99.34	106.78	89.11
Std	10.71	2.30	9.75	11.67	5.26	3.26	5.31	6.12	19.82	1.22
Skewness	−1.08	0.06	0.51	−0.08	−0.13	0.83	0.19	−0.13	0.88	0.77
Kurtosis	2.27	1.45	2.05	1.98	1.74	2.06	1.94	1.91	2.15	1.67
SAL										
Mean	292.96	265.41	293.57	249.22	289.95	238.51	293.10	277.02	276.33	261.21
Std	43.12	32.21	26.50	42.82	29.81	40.65	34.20	41.45	33.41	51.70
Skewness	−0.37	0.46	0.20	0.51	0.17	0.01	−0.24	0.48	−0.16	−0.08
Kurtosis	1.66	2.46	1.50	2.81	2.57	1.62	1.83	2.37	2.18	1.55
GAL										
Mean	388.76	411.45	400.09	390.24	405.21	417.00	421.60	401.49	392.50	397.37
Std	2.98	15.30	11.46	16.22	6.54	0.26	0.15	0.25	6.49	0.09
Skewness	−1.94	−0.75	0.80	1.15	−1.15	0.62	−1.10	0.02	−0.00	0.96
Kurtosis	4.83	2.14	1.98	2.33	2.33	1.82	2.29	1.19	1.00	2.20
PSOL										
Mean	375.78	352.54	366.50	360.32	380.18	342.21	360.97	378.66	364.54	359.51
Std	4.44	6.58	4.53	4.02	5.07	14.86	5.75	5.13	4.42	8.02
Skewness	0.75	0.76	0.74	1.04	0.49	0.32	0.50	0.44	−0.02	−0.34
Kurtosis	2.07	1.96	1.98	2.23	1.88	1.44	1.65	1.65	1.75	1.68
ICA										
Mean	426.74	433.22	429.61	432.43	432.15	424.31	422.22	438.97	428.11	438.75
Std	10.84	6.65	11.56	11.82	13.89	11.25	4.15	12.69	9.12	9.97
Skewness	−0.59	−0.21	0.16	0.06	1.06	0.99	0.14	0.01	−0.04	−0.77
Kurtosis	3.04	2.14	2.96	2.78	3.69	2.19	1.60	1.04	1.58	2.07
$r = 0.15$										
3PMA										
Mean	21.21	25.59	32.23	38.02	66.65	33.69	23.01	22.48	16.19	73.16
Std	4.81	3.51	11.19	3.57	7.40	13.51	2.46	5.46	0.58	28.28
Skewness	0.75	0.15	−1.07	−0.62	−0.92	0.03	0.98	0.24	−0.27	−0.79
Kurtosis	2.13	1.47	2.28	1.82	2.17	1.04	2.23	1.54	2.00	2.00

Table 2 continued

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
TSA										
Mean	106.83	106.39	109.56	99.84	95.14	87.15	103.04	104.78	86.92	88.90
Std	6.27	6.53	20.98	10.08	10.86	16.65	8.18	15.97	7.88	13.28
Skewness	−0.53	−0.89	−1.05	−0.41	−0.40	0.33	0.03	−0.65	0.40	−0.08
Kurtosis	1.67	2.18	2.24	1.59	1.82	1.46	1.56	1.86	1.58	1.11
PAES										
Mean	76.19	86.15	89.53	85.77	78.55	75.40	80.98	82.86	80.68	73.45
Std	2.39	7.09	15.60	8.27	3.23	3.60	8.61	3.96	5.05	0.46
Skewness	0.19	1.03	0.30	0.90	0.19	−0.20	1.11	−0.38	−0.36	−0.09
Kurtosis	1.28	2.23	1.63	2.10	1.65	1.91	2.30	1.60	1.77	1.36
SAL										
Mean	209.84	202.64	215.50	178.21	216.28	185.26	224.18	204.06	203.01	186.15
Std	34.56	27.18	20.35	30.49	31.50	31.17	26.63	40.83	45.99	42.21
Skewness	−0.27	−0.45	0.04	0.20	−1.64	−0.18	−0.11	0.92	0.01	−0.07
Kurtosis	2.82	2.01	2.71	2.79	6.71	2.39	2.28	3.69	2.09	1.59
GAL										
Mean	339.32	356.95	340.64	354.74	325.93	347.14	348.37	342.37	349.95	346.19
Std	0.44	5.55	7.02	0.27	0.39	17.93	0.35	7.46	0.27	0.21
Skewness	−0.18	−1.49	1.11	0.21	−0.20	1.15	−0.72	−1.15	0.60	0.20
Kurtosis	2.06	3.24	2.56	1.32	1.27	2.33	2.09	2.33	2.08	1.29
PSOL										
Mean	290.80	311.44	336.70	276.94	318.33	310.61	312.27	330.49	303.03	323.00
Std	10.33	11.57	3.41	13.00	4.63	6.46	4.41	3.33	6.52	4.16
Skewness	0.43	0.74	0.72	0.87	0.42	−0.04	0.63	0.00	0.65	0.04
Kurtosis	1.85	1.95	2.01	2.16	1.66	1.76	1.95	1.04	1.89	1.75
ICA										
Mean	368.69	375.47	373.12	371.62	369.67	375.66	372.81	375.38	375.82	371.65
Std	8.95	8.19	8.20	6.56	11.39	7.70	10.86	9.28	3.75	9.15
Skewness	−0.58	−0.31	0.08	−0.00	0.14	0.42	0.11	0.51	−0.40	−1.15
Kurtosis	2.65	2.26	2.21	2.04	5.50	2.48	2.47	1.96	2.02	2.33
$r = 0.18$										
3PMA										
Mean	22.75	10.81	21.40	15.66	23.46	25.20	32.85	51.07	15.57	50.45
Std	6.76	0.27	3.52	2.35	1.98	4.04	5.61	21.91	0.51	27.08
Skewness	0.60	0.63	0.14	−0.11	−0.13	0.07	0.96	−0.94	0.65	1.08
Kurtosis	1.88	1.82	1.59	1.25	1.20	1.10	2.21	2.17	1.88	2.28
TSA										
Mean	53.74	46.76	75.23	51.76	48.94	40.00	65.45	39.82	42.13	55.40
Std	10.02	11.23	15.24	2.89	18.35	11.71	5.10	4.42	15.78	13.62
Skewness	−0.11	−1.18	0.71	−1.13	−0.01	0.86	0.81	0.09	0.47	0.28
Kurtosis	1.66	2.90	1.90	2.31	1.12	2.15	2.10	1.17	2.04	1.40

Table 2 continued

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
PAES										
Mean	66.33	68.73	67.31	64.13	64.76	59.13	61.33	62.73	62.27	62.08
Std	3.01	3.13	3.36	1.26	3.43	1.11	1.50	1.39	1.16	2.29
Skewness	0.51	0.78	0.65	−0.26	0.44	−0.03	0.01	0.81	−0.50	−0.16
Kurtosis	1.65	1.96	1.84	1.40	2.02	1.04	1.02	2.15	1.76	1.56
SAL										
Mean	148.96	140.72	152.12	104.81	141.85	129.78	153.95	142.25	138.56	131.09
Std	35.60	28.79	28.36	18.85	33.51	36.83	27.01	34.20	34.63	34.33
Skewness	0.14	−0.28	0.48	0.54	0.02	0.08	0.59	0.63	0.14	0.31
Kurtosis	1.99	1.79	2.33	2.17	1.87	1.79	2.28	2.22	2.48	1.88
GAL										
Mean	287.92	279.27	292.66	288.32	280.30	289.51	284.88	296.30	266.56	264.45
Std	3.59	1.34	5.89	7.76	11.01	3.26	0.11	18.87	0.30	0.38
Skewness	1.18	0.21	−1.76	1.15	1.15	0.75	−0.59	0.00	−0.93	−0.72
Kurtosis	3.89	1.27	4.33	2.33	2.33	1.94	1.96	1.00	2.21	1.96
PSOL										
Mean	236.07	228.32	237.19	253.38	253.00	238.76	238.62	218.09	234.11	230.84
Std	4.34	10.51	8.55	5.22	8.32	7.62	6.53	8.30	8.54	9.40
Skewness	0.51	0.40	0.70	0.35	0.02	0.91	0.29	0.82	−0.25	0.78
Kurtosis	1.69	1.68	1.94	1.93	1.27	2.11	1.81	2.07	1.78	2.02
ICA										
Mean	309.28	312.86	310.91	303.62	303.72	308.93	314.82	313.49	314.59	309.98
Std	8.58	5.84	7.81	13.12	6.97	8.30	7.09	3.53	8.30	5.96
Skewness	−0.62	−1.13	0.86	−1.03	0.23	1.08	−1.10	−0.07	0.33	0.23
Kurtosis	2.41	3.80	3.05	2.22	1.32	2.28	2.29	1.09	1.58	1.55
$r = 0.22$										
3PMA										
Mean	18.39	12.23	14.17	14.02	11.11	22.37	18.75	15.34	16.36	19.75
Std	1.86	0.73	1.30	0.44	1.00	1.34	3.47	1.05	0.69	1.76
Skewness	−0.10	−0.19	−0.33	−0.75	−0.17	−0.74	1.58	−0.33	−0.59	0.14
Kurtosis	1.25	1.58	2.74	2.16	2.48	3.31	4.41	2.13	2.24	1.18
TSA										
Mean	23.14	27.03	24.30	19.71	22.61	36.64	32.15	18.71	29.73	24.43
Std	10.10	7.65	13.99	5.01	6.71	7.40	6.38	6.11	9.74	3.64
Skewness	0.45	0.13	0.65	0.30	−0.85	0.59	0.34	1.01	0.84	0.04
Kurtosis	1.58	1.17	1.88	1.40	2.15	2.06	1.65	2.22	2.16	1.16
PEAS										
Mean	52.04	53.66	53.21	54.09	48.00	51.63	52.90	55.05	53.55	54.43
Std	1.30	4.15	1.37	0.89	0.52	1.95	2.37	3.03	1.46	1.81
Skewness	0.46	0.03	1.11	−0.17	−1.15	−0.04	0.33	−0.08	−0.34	0.45
Kurtosis	1.88	1.26	2.30	1.32	2.33	1.45	1.58	1.13	1.68	1.60

Table 2 continued

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
SAL										
Mean	97.90	100.42	107.86	49.22	45.65	47.93	64.61	58.07	70.46	53.04
Std	33.13	34.46	27.31	5.69	11.00	13.04	8.04	7.79	11.54	14.36
Skewness	0.11	−0.04	0.26	−0.18	−1.15	0.49	−0.52	1.06	−0.58	0.95
Kurtosis	1.98	2.16	1.90	2.08	2.33	2.05	2.00	2.27	1.98	2.16
GAL										
Mean	230.36	233.06	222.25	239.97	219.08	230.31	234.22	233.46	224.43	233.10
Std	2.33	8.25	5.05	8.04	3.54	0.19	16.68	0.36	3.71	0.20
Skewness	2.57	0.94	1.82	1.15	1.14	−0.24	−0.19	0.05	1.15	0.46
Kurtosis	7.82	2.52	5.10	2.33	2.32	1.47	1.25	1.07	2.33	1.87
PSOL										
Mean	205.89	184.90	172.84	168.72	198.09	162.90	180.52	187.09	203.34	182.50
Std	5.43	8.42	10.20	9.98	5.66	14.45	5.79	10.72	6.95	6.02
Skewness	0.32	0.93	0.76	0.54	0.59	0.66	0.61	0.78	0.14	0.85
Kurtosis	1.83	2.16	1.96	1.80	1.90	1.98	1.83	2.07	1.81	2.10
ICA										
Mean	249.99	254.98	251.18	257.06	259.12	253.02	253.03	255.51	245.01	256.48
Std	6.28	6.03	7.60	8.28	7.97	7.77	5.24	8.11	5.28	8.15
Skewness	−0.91	0.57	−0.38	−0.27	−0.66	1.02	0.39	−0.79	0.98	0.01
Kurtosis	2.90	2.13	2.98	1.67	1.83	2.23	1.50	1.97	2.23	1.17

multi-trilateration technique with an optimization process such as the SA or self-adaptive local search results in better performance (see the results of 3PMA, TSA). It is also observed that the pure optimization algorithms could not yield high-quality results (see the results of the GAL, PSOL and ICA). To shed more light on the LE values, Fig. 7 demonstrates the estimated coordinates of network nodes, obtained by each algorithm on *TOP*0, $r = 0.13$ where black solid stars, rectangles, multiplication signs and straight lines represent the coordinates of anchor nodes, the real positions of non-anchor nodes, the estimated positions of non-anchor nodes and the Euclidean distance between the real and estimated positions of non-anchor nodes, respectively.

It is demonstrated that the 3PMA estimates the positions of non-anchor nodes with higher accuracy than the other algorithms. After that the PAES and TSA offer good accuracy, compared to the other algorithms.

4.2 Comparison on network topologies with holes

In this section, we investigate the performance of the proposed algorithm on four topologies (network with O-shape, H-shape, E-shape and C-shape hole) that are used in WSNs [40]. In the first topology (O-shape topology) network nodes are scattered in O-shape fashion. In this topology, the hole blocks the signal transmitting among nodes, so it is hard to find out the positions of non-anchor

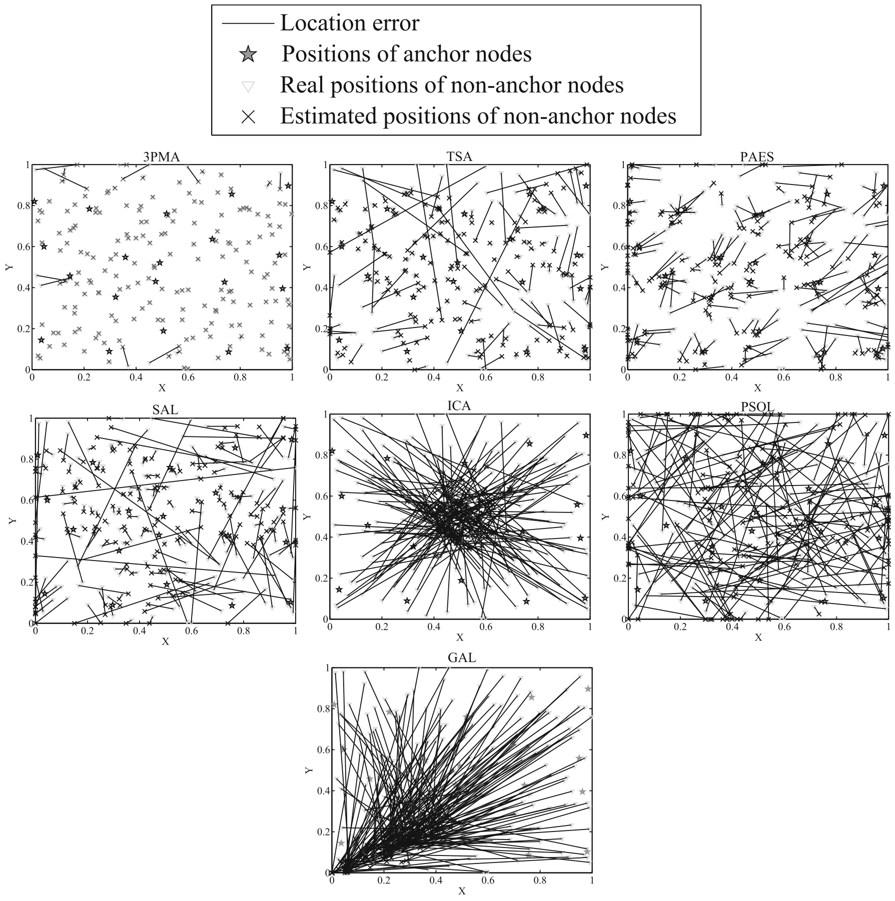


Fig. 7 Location estimates using different optimization techniques on *TOP*0 and $r = 0.13$

nodes for inter-node distance data are lost in the hole. Likewise, in H-topology, C-topology and E-topology, the sensor nodes are distributed in H, C and E-shape fashions, respectively. Similar to the random network topologies, the number of anchor nodes and non-anchor nodes are set to 20, 180 in these topologies, respectively. We also use the same values of $r = (0.13, 0.15, 0.18, 0.22)$ and noise factor ($NF = 0.1$). Figure 8 shows how the network nodes are distributed in the model-based topologies.

Figure 9 represents the number of non-anchor nodes versus the number of anchor nodes in their neighborhoods on four different values of transmission ranges of nodes, respectively.

It is shown in Fig. 9, in the O-shape topology, that about 68 % of non-anchor nodes have 4 neighboring anchor nodes where $r = 0.13$, and none of non-anchor nodes has more than 8 adjacent anchor nodes. Furthermore, in these topologies each

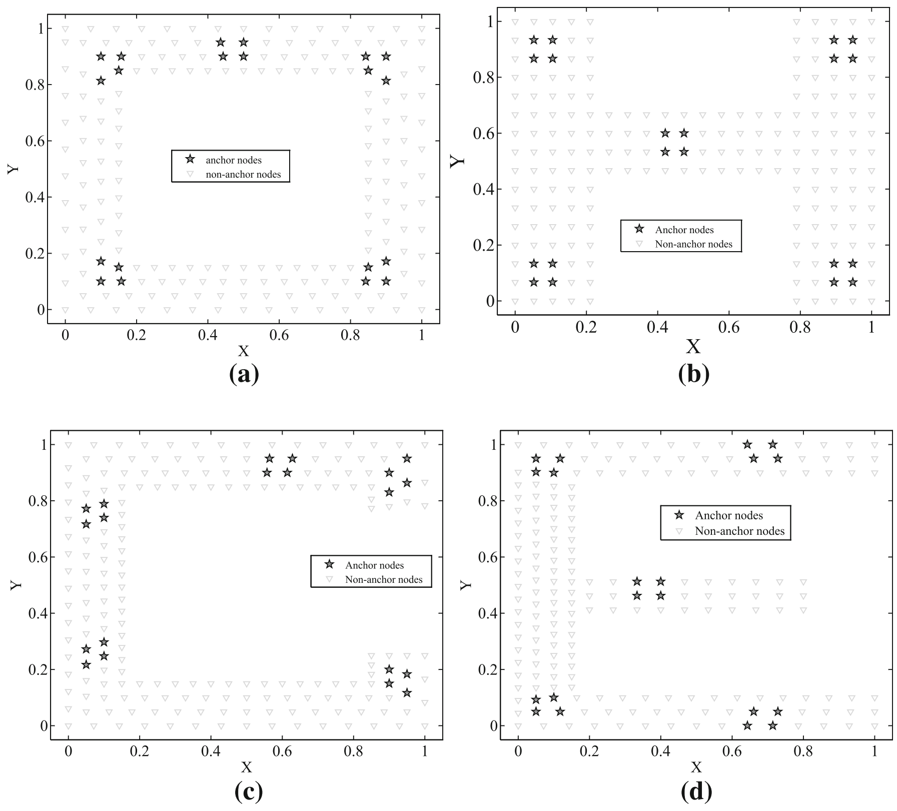


Fig. 8 Proposed network topologies

non-anchor node has more neighboring anchor nodes than it does in the random topologies, making them easier to solve. However, the existence of a hole is a major hindrance to estimate the locations of anchor nodes.

To make sure that the results obtained by each algorithm is good enough, we use the best parameters for all the algorithms as represented in Table 1. We also use the same termination condition for all the algorithms; that is 50,000 FCs and when no better solution has been reached for 100 iterations as described in the previous subsection. The mean and standard deviation of the *LE* values obtained by each algorithm on four model-based network topologies with 4 different values of communication ranges are summarized in Table 3. The best results are typed in bold.

As expected, 3PMA outperforms the other algorithms in all cases. After that, the TSA and PAES gain the second and third best results in terms of the LE values. The data corroborate the results obtained by the algorithms on random network topologies. To show how well the algorithms estimate the positions of non-anchor nodes, we provide an example of coordinate estimation of sensor nodes executed by each algorithm where the H-shape topology is used and *r* is equal to 0.13 in Fig. 10.

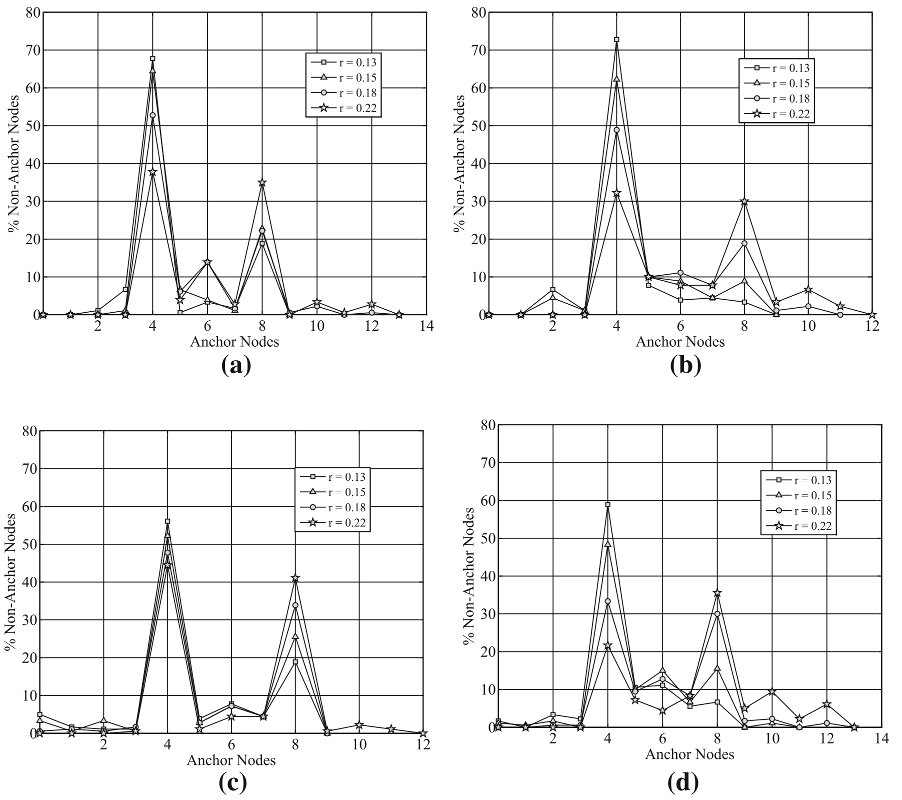


Fig. 9 The percentage of non-anchor nodes versus anchor nodes in their neighborhoods on 4 different values of communication ranges

Note that the estimated coordinates obtained by each algorithm are the best solution in 20 trials.

To show how the optimization process affects the LE values as well as *Cx* and *Cv* values, we provide an example of average performance trends of the algorithms on the *H*-shape topology and *r*=0.13, as represented in Fig. 11.

As shown in Fig. 11, the 3PMA is best at estimating the positions of nodes since it achieves the best *LE* values in the final stage of the optimization. Note that although the TSA gains the best *LE* values at the beginning of the optimization, after that phase, its performance deteriorates since the algorithm does not know its *LE* values during the optimization. Further, the 3PMA shows the steep trend at the beginning of the optimization. After that phase, the 3PMA does not stagnated but continues to advance toward the global optimum at a slower speed. It is also shown that the 3PMA is remarkably better than the other algorithms on *CX* values and is significantly better than other two-objective optimization technique (PAES) on *CV* values.

Table 3 The mean, standard deviation, skewness and kurtosis of the *LE* values obtained by 7 optimization techniques over 20 runs for the four model-based network topologies with 4 different values of communication ranges

Topologies	$r = 0.13$							$r = 0.15$						
	3PMA	TSA	PAES	SAL	GAL	PSOL	ICA	3PMA	TSA	PAES	SAL	GAL	PSOL	ICA
H-Shape														
Mean	**65.45**	187.54	271.34	306.86	388.04	374.79	447.51	**39.63**	166.25	166.19	259.93	356.93	295.70	391.21
Std	15.89	31.37	15.71	19.04	0.36	9.52	7.90	10.50	23.29	7.15	19.78	0.28	20.27	12.14
Skewness	−0.01	−0.61	−0.52	−0.86	−0.94	0.21	0.38	−0.03	−0.40	−0.02	0.24	0.21	0.62	1.02
Kurtosis	1.01	1.88	1.50	2.07	2.19	1.36	1.55	1.08	2.25	1.03	2.00	1.28	1.94	2.25
O-Shape														
Mean	**18.56**	213.73	270.34	276.38	400.83	332.73	479.45	**16.13**	163.86	158.62	204.15	346.57	305.42	411.46
Std	1.63	28.62	11.13	9.61	5.53	23.61	6.66	2.51	15.77	6.91	5.92	0.19	10.78	7.78
Skewness	−0.06	−0.71	−0.77	0.35	1.14	0.79	0.20	−0.31	0.43	1.00	−0.44	0.85	0.47	−1.14
Kurtosis	1.50	1.99	1.96	1.49	2.33	2.00	1.96	1.40	2.03	2.22	1.87	2.15	1.74	2.32
C-Shape														
Mean	**48.54**	215.23	265.23	322.65	406.03	357.10	486.08	**18.41**	181.21	179.57	274.14	356.36	273.98	412.30
Std	8.07	22.93	19.08	15.74	4.57	19.96	9.03	1.83	22.77	3.14	9.36	1.72	28.16	10.70
Skewness	−0.77	0.52	0.97	0.55	1.15	0.76	−0.84	−1.08	−0.03	0.99	−0.03	0.01	0.24	0.51
Kurtosis	1.97	1.98	2.18	2.02	2.33	2.12	2.03	2.28	1.07	2.20	1.08	1.03	1.54	1.70
E-Shape														
Mean	**21.93**	219.69	281.20	331.70	425.25	342.71	472.03	**31.70**	150.87	178.90	271.27	344.16	328.21	407.27
Std	2.26	26.60	7.27	39.29	7.14	20.67	11.13	7.43	20.87	4.01	20.12	3.00	19.31	10.41
Skewness	0.71	0.55	0.00	0.09	1.15	0.71	−0.74	−0.05	0.04	0.00	−0.34	0.00	0.33	−0.98
Kurtosis	1.99	2.00	1.00	1.16	2.33	1.99	2.09	1.94	1.83	1.00	1.92	1.00	1.48	2.18

Table 3 continued

Topologies	$r = 0.18$							$r = 0.22$						
	3PMA	TSA	PAES	SAL	GAL	PSOL	ICA	3PMA	TSA	PAES	SAL	GAL	PSOL	ICA
H-Shape														
Mean	**27.39**	63.82	117.27	207.02	292.09	279.57	327.79	**25.49**	50.36	88.49	145.10	252.63	191.02	274.22
Std	3.56	35.47	5.77	4.95	0.20	10.07	10.41	7.51	44.50	2.44	3.74	4.48	12.85	4.15
Skewness	0.46	0.26	0.52	0.85	0.79	0.48	1.08	−0.02	0.43	−0.37	0.32	0.01	0.71	0.71
Kurtosis	1.60	1.78	1.50	2.17	2.06	1.66	2.27	1.03	1.33	1.91	1.68	1.02	1.93	2.01
O-Shape														
Mean	**37.91**	116.35	98.84	161.69	289.29	240.86	344.16	**38.10**	78.27	81.19	103.20	240.73	184.52	279.27
Std	18.48	11.14	4.59	13.93	0.40	15.19	6.79	10.04	17.57	0.58	4.22	0.30	17.49	8.16
Skewness	−0.76	0.09	−0.37	−0.21	1.00	0.17	−0.65	−0.12	0.17	−0.13	0.59	0.01	1.00	0.16
Kurtosis	2.13	1.52	2.00	1.94	2.20	1.39	2.10	1.15	1.43	1.84	2.07	1.71	2.21	2.00
C-Shape														
Mean	**17.02**	144.72	130.17	207.43	291.78	242.49	344.10	**14.00**	67.59	91.01	160.36	246.50	203.08	285.09
Std	0.49	17.84	3.15	4.73	0.38	19.36	4.50	1.98	33.82	1.50	5.11	4.52	20.73	7.81
Skewness	−0.00	1.13	0.16	−0.54	0.51	0.38	0.22	0.27	0.10	−0.18	0.33	−0.00	0.50	0.18
Kurtosis	1.01	2.32	2.01	1.74	1.84	1.87	1.65	1.42	2.00	1.63	1.71	1.03	1.71	1.44
E-Shape														
Mean	**68.79**	125.50	116.23	210.20	297.81	282.39	349.22	**17.57**	56.06	76.14	164.53	247.79	231.60	272.45
Std	17.21	21.62	2.11	11.09	0.17	10.88	12.68	3.52	23.32	1.81	7.68	5.08	7.98	3.51
Skewness	−0.34	0.96	0.00	−0.44	0.34	0.99	−0.64	−0.42	−0.77	0.53	−1.14	−1.03	0.57	0.48
Kurtosis	1.49	2.21	1.00	1.88	1.44	2.21	1.92	1.50	2.11	1.68	2.33	2.22	1.97	1.62

The best results are typed in bold

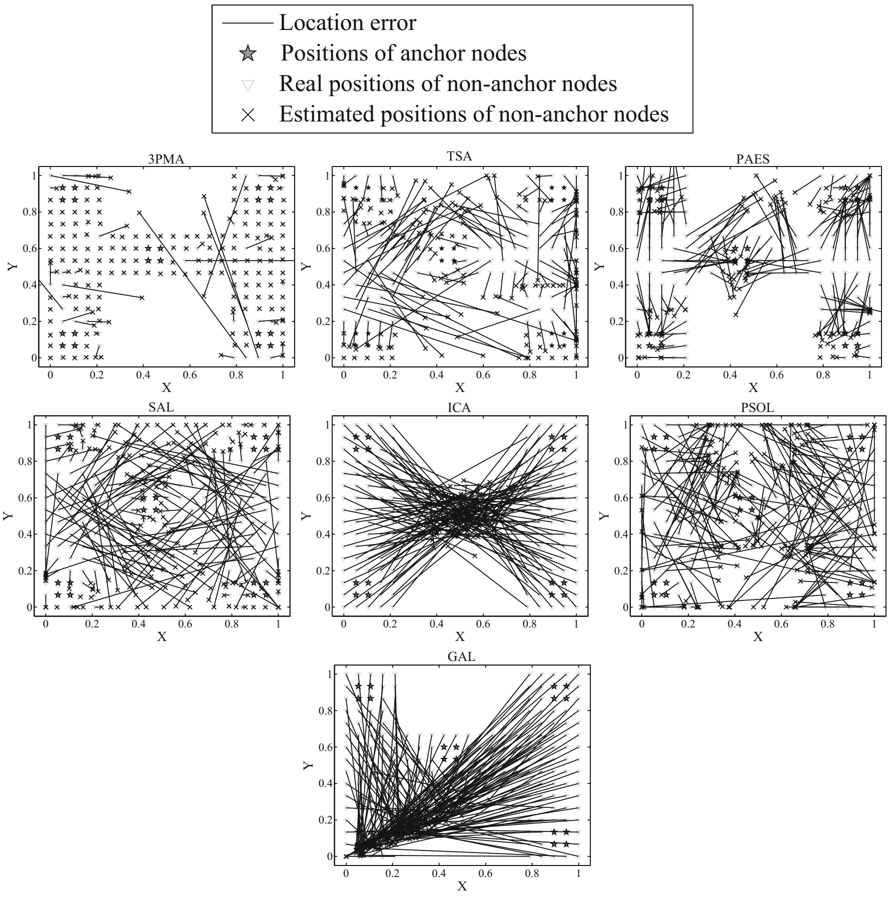


Fig. 10 The estimated coordinates of network nodes obtained by each algorithm on the H-shape topology and $r = 0.13$

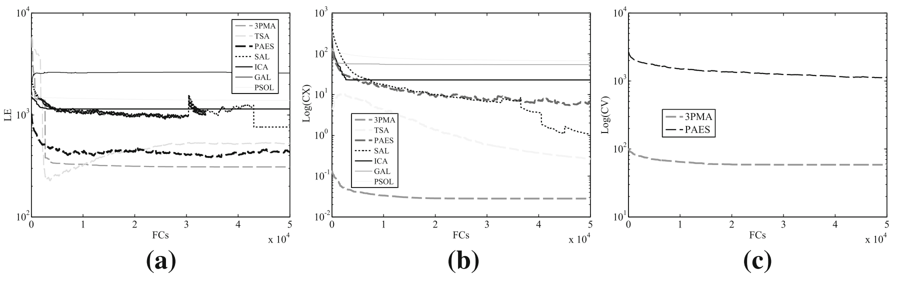


Fig. 11 Performance trends of the 3PMA and the other six optimization techniques on the H-shape topology and $r = 0.13$

4.3 Statistical rankings through Holm–Benferroni procedure

In order to provide statistical rankings for all the algorithms, we use the Holm–Benferroni procedure that is commonly used in the literature [41, 42]. This procedure is performed on the entire testbed (ten randomly created and four model-based created networks with four different values of connectivity ranges). Generally speaking, the procedure ranks the algorithms according to their respective performance. To this end, we first score each algorithm on each network topology and each connectivity range. The smaller LE value an algorithm has, the higher score it gets. For example, if an algorithm displays the second best performance, it gets a score of 6. Likewise, if an algorithm obtains the worst performance, it gets a score of 1. Then, the algorithms are sorted based on the scores they have obtained. Finally, z_j of each algorithm is calculated as,

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{tp}}}} \quad for\, j = 1, \ldots, N_A, \tag{7}$$

where R_0 is the rank of the 3PMA, R_j is the rank of j-th algorithm calculated over the whole testbed (10 random and 4 model-based network topologies with 4 different values of connectivity ranges which is equal to 56) and N_A is the number of competitive algorithms which is equal to 6. Taking z_j values into consideration, we can calculate the respective cumulative normal distribution values, p_j. We then compare the p_j values with corresponding δ/j values where δ is the level of confidence, set to 0.05. Further, the null-hypothesis is provided to show the outperfomance of the 3PMA statistically. A null-hypothesis can be 'rejected' or 'Accepted', depending on the performance similarity between the proposed algorithm and the other algorithms. This means that if the performance of the proposed algorithm is statistically better than the other algorithm, the null-hypothesis is rejected and if no remarkable difference between the performance of the 3PMA and the other algorithm is shown, the null-hypothesis is accepted. Table 4 shows z_j values, the *ranks*, p_j values, the corresponding δ/j and null-hypothesis of the Holm test.

As shown, the proposed algorithm attains the highest rank among all the other algorithms and TSA, PAES and SAL achieve the second to forth best ranks.

Table 4 Holm test on the whole testbed and algorithms, reference algorithm = 3PMA(Rank=6.89)

j	Algorithm	Rank	z_j	p_j	δ/j	Hypothesis
1	TSA	5.57	−0.06	7.83e−12	5.00e−02	Accepted
2	PAES	5.46	−0.07	1.21e−14	2.50e−02	Rejected
3	SAL	4.05	−0.14	4.23e−17	1.67e−02	Rejected
4	PSOL	3.01	−0.20	5.68e−20	1.25e−02	Rejected
5	ICA	2.00	−0.25	3.70e−20	1.00e−02	Rejected
6	GAL	1.00	−0.30	3.70e−20	8.33e−03	Rejected

Table 5 The mean, standard deviation, skewness and kurtosis of the *LE* values obtained by 3 different version of our proposed algorithm over 20 runs on the ten network topologies and for 2 different values of communication ranges

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
$r = 0.13$										
3PMA										
Mean	**118.64**	**44.74**	**38.98**	**100.65**	**53.59**	**114.26**	**122.59**	**32.56**	**121.44**	**41.77**
Std	9.77	3.47	4.90	9.89	6.32	17.39	9.90	0.58	8.95	0.78
Skewness	0.17	−0.91	−0.42	0.19	1.43	0.04	−0.23	−0.75	0.30	1.04
Kurtosis	3.59	2.36	1.61	2.54	4.46	2.05	1.83	3.24	2.14	3.12
2PMAV										
Mean	149.43	150.43	155.16	137.26	139.10	145.29	157.22	144.79	150.07	145.20
Std	5.47	5.53	4.47	5.18	4.90	4.74	6.12	5.50	4.43	4.92
Skewness	0.14	−0.50	-0.11	−0.10	0.28	0.53	0.08	−0.64	−0.33	0.51
Kurtosis	2.93	2.33	2.21	2.63	2.40	3.04	2.71	2.90	3.79	3.06
2PMAR										
Mean	439.22	449.41	441.61	442.33	440.94	442.39	446.36	445.06	444.20	440.86
Std	15.25	11.89	10.94	12.08	8.38	13.30	13.75	13.51	13.53	13.85
Skewness	0.46	−0.38	0.03	−0.04	0.41	0.21	0.39	−0.18	0.71	0.26
Kurtosis	2.26	2.37	2.16	2.19	3.67	2.93	1.91	1.96	3.52	2.24
	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
$r = 0.22$										
3PMA										
Mean	**51.29**	**30.97**	**28.83**	**27.47**	**22.11**	**78.84**	**28.69**	**59.93**	**61.86**	**69.70**
Std	0.40	0.25	0.43	0.28	1.20	1.07	0.46	0.57	1.03	0.62
Skewness	−0.12	0.02	−0.62	0.02	−0.78	−1.26	0.005	0.41	0.54	−0.93
Kurtosis	2.14	1.44	2.04	1.44	2.06	2.91	1.95	1.88	1.96	2.44

Table 5 continued

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
2PMAV										
Mean	139.37	143.25	144.94	138.16	135.16	137.01	144.08	138.42	139.48	139.56
Std	6.42	5.88	4.02	5.13	5.63	3.17	5.22	4.70	4.52	4.53
Skewness	0.84	−0.63	0.20	0.09	−0.38	0.29	0.23	0.29	0.53	0.57
Kurtosis	3.82	3.31	2.82	1.92	2.30	2.43	2.89	2.25	2.61	2.01
2PMAR										
Mean	257.64	265.38	259.23	262.19	260.75	261.15	261.86	263.39	261.30	261.79
Std	6.39	9.00	6.69	9.68	8.83	8.27	7.04	6.44	6.33	6.74
Skewness	−0.24	−0.64	0.11	0.67	−0.89	−0.01	1.06	0.09	−0.20	0.61
Kurtosis	2.39	3.69	2.31	2.63	3.16	2.16	3.72	1.98	3.54	2.99

The results are obtained before the optimization process

Table 6 The mean, standard deviation, skewness and kurtosis of the *LE* values obtained by 3 different version of our proposed algorithm over 20 runs on the ten network topologies and for 2 different values of communication ranges. The results are obtained after the optimization process

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
$r = 0.13$										
3PMA										
Mean	69.22	33.95	23.96	70.51	40.68	105.31	124.23	26.55	29.71	29.14
Std	7.30	3.43	2.72	12.75	9.92	15.08	40.09	3.31	4.40	2.98
Skewness	0.70	0.03	−1.15	−0.84	1.32	0.40	−1.45	0.49	−1.54	−0.12
Kurtosis	2.13	2.63	3.21	3.75	4.36	2.02	3.18	2.21	3.78	1.39
2PMAV										
Mean	131.17	139.38	144.25	124.29	128.81	134.46	150.10	136.04	141.19	128.15
Std	8.57	3.00	2.94	4.17	1.33	1.28	3.38	4.48	1.58	9.27
Skewness	0.16	−0.14	0.83	−0.71	0.31	0.49	−0.03	−0.37	0.17	−0.61
Kurtosis	1.40	1.35	2.05	1.93	1.63	1.72	1.77	1.58	1.25	1.82
2PMAR										
Mean	415.58	417.53	417.76	420.84	418.31	413.09	418.50	415.98	395.38	424.19
Std	7.00	6.81	3.32	5.19	9.93	12.78	7.70	3.84	25.76	6.38
Skewness	0.76	−0.86	0.42	0.89	−1.03	−0.01	−0.00	0.56	0.00	−0.99
Kurtosis	2.12	2.06	1.98	2.14	2.25	1.48	1.01	1.85	1.01	2.21
$r = 0.22$										
3PMA										
Mean	**38.25**	**15.80**	**15.82**	**15.59**	**11.38**	**31.85**	**18.70**	**24.71**	**22.71**	**31.10**
Std	8.71	5.58	1.37	2.34	1.40	4.58	2.26	10.89	6.50	6.61
Skewness	0.16	1.91	0.71	0.36	0.50	−0.24	1.18	1.03	0.72	−0.04
Kurtosis	2.45	5.24	3.16	1.69	2.06	1.44	4.81	2.82	2.24	1.89
2PMAV										
Mean	128.79	129.58	136.14	119.94	120.71	120.45	126.73	127.03	123.10	121.64

Table 6 continued

	TOP0	*TOP1*	*TOP2*	*TOP3*	*TOP4*	*TOP5*	*TOP6*	*TOP7*	*TOP8*	*TOP9*
Std	3.92	8.47	2.44	8.53	12.71	7.17	6.03	1.58	4.42	4.01
Skewness	−0.52	−0.52	−0.50	−1.07	−1.13	−0.84	−0.63	−0.06	0.15	0.53
Kurtosis	2.04	1.71	1.64	2.26	2.32	2.08	2.38	1.13	1.60	1.96
2PMAR										
Mean	200.72	201.11	238.97	203.66	228.34	197.10	208.69	221.72	202.66	222.97
Std	18.64	19.63	12.96	27.00	6.63	21.94	22.48	13.86	22.03	14.95
Skewness	−0.04	−0.90	−0.67	0.87	−0.43	1.12	0.69	−0.42	−0.23	−0.92
Kurtosis	1.10	2.15	2.09	2.17	1.91	2.34	2.05	2.73	1.53	2.14

Table 7 The mean, standard deviation, skewness and kurtosis of the *LE* values obtained by 3 different version of our proposed algorithm over 20 runs on the ten network topologies and for 2 different values of communication ranges. The results are obtained after the optimization process

	TOP0	*TOP1*	*TOP2*	*TOP3*	*TOP4*	*TOP5*	*TOP6*	*TOP7*	*TOP8*	*TOP9*
$r = 0.13$										
3PMA										
Mean	69.22	33.95	23.96	70.51	40.68	105.31	124.23	26.55	29.71	29.14
Std	7.30	3.43	2.72	12.75	9.92	15.08	40.09	3.31	4.40	2.98
Skewness	0.70	0.03	−1.15	−0.84	1.32	0.40	−1.45	0.49	−1.54	−0.12
Kurtosis	2.13	2.63	3.21	3.75	4.36	2.02	3.18	2.21	3.78	1.39
1PMAPAES										
Mean	41.69	35.64	29.58	36.05	32.14	98.13	58.15	31.15	30.67	30.72
Std	4.66	3.60	1.27	2.53	0.25	2.01	7.96	3.43	2.26	0.69
Skewness	−0.03	0.00	0.00	−0.00	−0.00	−0.00	0.00	−0.00	0.00	−0.00
Kurtosis	2.37	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1PMASA										
Mean	273.04	222.18	267.87	205.46	278.72	183.55	249.44	244.37	221.47	232.15
Std	52.71	17.43	57.52	26.26	30.26	16.48	27.01	33.30	26.88	46.85
Skewness	0.28	0.74	−0.00	0.00	0.00	−0.01	0.44	1.70	0.27	0.27
Kurtosis	1.95	3.42	1.00	1.00	1.00	2.74	2.28	5.22	2.06	2.44
$r = 0.22$										
3PMA										
Mean	**38.25**	**15.80**	**15.82**	**15.59**	**11.38**	**31.85**	**18.70**	**24.71**	**22.71**	**31.10**
Std	8.71	5.58	1.37	2.34	1.40	4.58	2.26	10.89	6.50	6.61
Skewness	0.16	1.91	0.71	0.36	0.50	−0.24	1.18	1.03	0.72	−0.04
Kurtosis	2.45	5.24	3.16	1.69	2.06	1.44	4.81	2.82	2.24	1.89
1PMAPAES										
Mean	19.68	12.43	16.93	14.20	12.84	16.93	18.40	17.93	16.86	20.95

Table 7 continued

	*TOP*0	*TOP*1	*TOP*2	*TOP*3	*TOP*4	*TOP*5	*TOP*6	*TOP*7	*TOP*8	*TOP*9
Std	1.38	0.16	2.88	0.14	0.93	0.27	0.30	0.00	0.78	1.92
Skewness	0.36	−0.00	−0.00	−0.00	0.00	0.00	0.00	NaN	−0.00	−0.00
Kurtosis	1.77	1.00	1.00	1.00	1.00	1.00	1.00	NaN	1.00	1.00
1PMASA										
Mean 50.38	63.51	77.44	48.10	57.02	44.31	78.29	62.51	66.75	45.07	
Std	14.56	5.78	1.40	4.95	2.99	7.90	8.17	9.99	7.33	8.23
Skewness	0.10	−0.71	0.00	0.00	0.00	−0.38	−0.72	−0.54	0.78	0.71
Kurtosis	1.64	3.34	1.00	1.00	1.00	2.02	2.85	2.59	2.80	2.21

5 Analysis of the proposed algorithm

In this section, we study the proposed algorithm, concentrating on the initialization and optimization processes. In order to investigate the efficiency of the initialization procedure, we test the proposed algorithm where three different ways of initialization, our proposed method (approximation stage), random initialization [22–24, 28, 33] and Vecchino's initialization approach [19], on ten random network topologies (*TOP*0–*TOP*9) are used. The two latter ones are called Two-Phase Memetic Approach with Random initialization (2PMAR) and Two-Phase Memetic Approach with Vecchino's initialization (2PMAV). For optimization stage, our proposed algorithm is tested when equipped with Pareto Archived Evolution Strategy (PAES) [19] and when employed the memetic approach (exploration plus exploitation stages). We measure the performance of the algorithm in two different times: after the initialization process and after the optimization process. The termination condition for all experiments are set to 200,000 function evaluations and when no better solution has been reached for 100 iterations. Table 5 summarizes the results of the comparison of three different versions of our algorithm before the optimization stage on ten network topologies and for two different values of communication ranges.

As shown in Table 5, the proposed initialization procedure is more promising than both other approaches as it gains success in all network topologies and two different transmission ranges. In order to show that our algorithm maintains its progress until the end, it is tested against its rivals after the optimization process. The results are summarized in Table 6.

According to Tables 5 and 6, we can claim that our proposed approach is more successful than Vecchino and random approaches as beating them in all cases.

In the first part, we showed that our proposed initialization (approximation phase) is very promising, and now in order to study the optimization part of our algorithm, we compare it with two successful methods in the previous section, namely PAES [19] and SA [13]. To do so, we detach the optimization procedure (exploration and exploitation stages) from our algorithm and then integrate the optimization part of SA and PAES methods instead. Finally, we test the algorithms with the problem and algorithm configurations similar to the first part and report the results in Table 7. We call the version of our algorithm equipped with SA, 1PMASA, and the one equipped with PAES, 1PMAPAES.

As shown in Fig. 7, our proposed algorithm gains better results in most cases, so we can conclude that our optimization part of the algorithm is also more promising than the optimization part of the other competitive algorithms specially when more neighboring anchor nodes are available for each non-anchor node. It is also clear that the results obtained by our approach can be more attributed to the approximation stage than the optimization (exploration and exploitation stages) as the approximation stage brings about more progress than the other one.

6 Conclusion

In this paper, we proposed a new optimization technique for the fine-grained localization problem in WSNs that uses three heterogeneous memetic operators. The first meme called approximation procedure is used in the preprocessor phase of the algorithm in order to find good initial locations for non-anchor nodes. The second meme, called exponential crossover, is utilized in the exploration phase of the algorithm to create a new solution during optimization process. Finally, a meme called the self-adaptive local search is employed in the exploitation phase of the algorithm to improve the positions of non-anchor nodes. To test the proposed algorithm, it was compared with six existing optimization approaches including TSA, PAES, SAL, GAL, ICA and PSOL on ten randomly created network topologies. To further test the algorithm, we compared it on the networks with (O, C, H or E)-shape hole. The results suggested that the proposed algorithm is better than the other algorithms in terms of the quality of location estimation of sensor nodes on the created network topologies. We have also observed that the use of approximation stage significantly helps the algorithm find better solutions in a smaller amount of time.

Beside from the advantages mentioned, our algorithms like other optimization algorithms suffers from some drawbacks. First, it has 5 parameters, taking a large amount of time to set them to the problem at hand. Second, like other algorithms presented, it is a stochastic technique, that is the results found by our algorithm is not exact and determined. Third, because of its nature (combination of the mathematics and the artificial intelligence), the algorithm requires a complicated implementation.

With some minor modifications, the proposed algorithms can be used for solving many optimization problems ranging from latin hypercube design problem [43] to university timetabling problem [44]. In one future work, we will apply the proposed algorithm on the university timetabling problem as it showed remarkable success in dealing with real-world optimization problem in this paper. In another future work, we will study the landscape of the localization problem and try to propose an algorithm based on that analysis. In one future work, we also plan to investigate the use of co-evolution strategy [45] in the proposed algorithm as it has shown remarkable improvement for the optimization algorithm. Using the opposition-based learning [46] is also a good option for the exploitation phase of the algorithm, as it has gained wide attentions in facilitating the search process.

References

1. I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
2. A. Pal, Localization algorithms in wireless sensor networks: current approaches and future challenges. Netw. Protoc. Algorithms **2**(1), 45–73 (2010)
3. O. Banimelhem, S. Khasawneh, Gmcar: grid-based multipath with congestion avoidance routing protocol in wireless sensor networks. Ad Hoc Netw. **10**(7), 1346–1361 (2012)

4. F. Xue, P.R. Kumar, On the coverage and connectivity of large random networks. IEEE/ACM Trans. Netw. **14**(SI), 2289–2299 (2006)
5. C. Zhang, Y. Zhang, Y. Fang, Localized algorithms for coverage boundary detection in wireless sensor networks. Wirel. Netw. **15**, 3–20 (2009)
6. K. Kim, A Clustering Algorithm Based on Geographical Sensor Position in Wireless Sensor Networks, in *Innovative Algorithms and Techniques in Automation*, ed. by T. Sobh, K. Elleithy, A. Mahmood, M. Karim (Industrial Electronics and Telecommunications, Springer, Netherlands, 2007), pp. 245–249
7. L. Sun, J. Guo, K. Lu, R. Wang, Topology control based on quantum genetic algorithm in sensor networks. Front. Electr. Electron. Eng. China **2**, 326–329 (2007)
8. G. Mao, B. Fidan, *Localization Algorithms and Strategies for Wireless Sensor Networks* (Premier Reference Source, Information Science Reference, Hershey, 2009)
9. E. Niewiadomska-Szynkiewicz, Localization in wireless sensor networks: classification and evaluation of techniques. J. Appl. Math. Comput. Sci. **22**(2), 281–297 (2012)
10. Y. Liu, Z. Yang, Location, localization, and localizability. J. Comput. Sci. Technol. **25**(2), 274–297 (2010)
11. S. Yun, J. Lee, W. Chung, E. Kim, S. Kim, A soft computing approach to localization in wireless sensor networks. Expert Syst. Appl. **36**(4), 7552–7561 (2009)
12. B. Hofmann-Wellenhof, H. Lichtenegger, J. Collins, *Global Positioning System: Theory and Practice*, 5th edn. (Springer, Berlin, 2001)
13. E. Niewiadomska-Szynkiewicz, M. Marks, Optimization schemes for wireless sensor network localization. Int. J. Appl. Math. Comput. Sci. **19**(2), 291–302 (2009)
14. Y. Zhang, L.T. Yang, J. Chen, *RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations*, 1st edn. (CRC Press Inc, Boca Raton, FL, USA, 2009)
15. F. Franceschini, M. Galetto, D. Maisano, L. Mastrogiacomo, A review of localization algorithms for distributed wireless sensor networks in manufacturing. Int. J. Comput. Integr. Manuf. **22**(7), 698–716 (2009)
16. M. Gholami, N. Cai, R. Brennan, An artificial neural network approach to the problem of wireless sensors network localization. Robot. Comput.-Integr. Manuf. **29**(1), 96–109 (2013)
17. J. Kuriakose, S. Joshi, R.V. Raju, A. Kilaru, A Review on Localization in Wireless Sensor Networks, in *Advances in Signal Processing and Intelligent Recognition Systems*, ed. by J. Fagerberg, D.C. Mowery, R.R. Nelson (Springer Netherlands, Netherlands, 2014), pp. 599–610
18. N. Patwari, Location estimation in sensor networks, Ph.D. thesis, Citeseer (2005)
19. M. Vecchio, R. Lpez-Valcarce, F. Marcelloni, A two-objective evolutionary approach based on topological constraints for node localization in wireless sensor networks. Appl. Soft Comput. **12**(7), 1891–1901 (2012). soft Computing Approaches in the design of energy-efficient wireless systems
20. L. Doherty, K. S. J. Pister, L. El Ghaoui, Convex position estimation in wireless sensor networks, in: *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*. Proceedings. IEEE, vol. 3, 2001, pp. 1655–1663
21. M. Naraghi-Pour, G.C. Rojas, A novel algorithm for distributed localization in wireless sensor networks. ACM Trans. Sen. Netw. **11**(1), 1:1–1:25 (2014). doi:10.1145/2632150
22. A. Kannan, G. Mao, B. Vucetic, Simulated annealing based localization in wireless sensor network, in: Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on, 2005, pp. 2 pp. – 514
23. A. Kannan, G. Mao, B. Vucetic, Simulated annealing based wireless sensor network localization with flip ambiguity mitigation, in: Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd, Vol. 2, 2006, pp. 1022–1026
24. Q. Zhang, J. Wang, C. Jin, J. Ye, C. Ma, W. Zhang, Genetic algorithm based wireless sensor network localization, in: Natural Computation, 2008. ICNC '08. Fourth International Conference on, Vol. 1, 2008, pp. 608–613
25. A. O. Sá, N. Nedjah, L. Macedo Mourelle, Computational Science and Its Applications – ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30 – July 3, 2014, Proceedings, Part V, Springer International Publishing, Cham, 2014, Ch. Genetic and Backtracking Search Optimization Algorithms Applied to Localization Problems, pp. 738–746
26. Localization algorithm in wireless sensor networks based on multiobjective particle swarm optimization, Int. J. Distrib. Sens. Netw. (2015)
27. D. Manjarres, J.D. Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, S. Salcedo-Sanz, R. Lopez-Valcarce, On the design of a novel two-objective harmony search approach for distance- and

connectivity-based localization in wireless sensor networks. Eng. Appl. Artif. Intell. **26**(2), 669–676 (2013)
28. M. Sayadnavard, A. Haghighat, M. Abdechiri, Wireless sensor network localization using imperialist competitive algorithm, in: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, **9**, 818–822 (2010)
29. G. Iacca, F. Neri, E. Mininno, Y.-S. Ong, M.-H. Lim, Ockhams razor in memetic computing: three stage optimal memetic exploration. Inf. Sci. **188**, 17–43 (2012)
30. F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures. Inf. Sci. **227**, 60–82 (2013)
31. P.C. Pop, O. Matei, A memetic algorithm approach for solving the multidimensional multi-way number partitioning problem. Appl. Math. Modell. **37**, 9191–9202 (2013)
32. H. Brandner, S. Lessmann, S. Vo, A memetic approach to construct transductive discrete support vector machines. Eur. J. Oper. Res. **230**(3), 581–595 (2013)
33. P.-J. Chuang, C.-P. Wu, An effective pso-based node localization scheme for wireless sensor networks, in: Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on, 2008, pp. 187–194
34. Q.H. Nguyen, Y.-S. Ong, M.-H. Lim, A probabilistic memetic framework. IEEE Trans. Evolut. Comput. **13**(3), 604–623 (2009)
35. N. Krasnogor, Studies on the theory and design space of memetic algorithms, Ph.D. thesis (2002)
36. H. Hoos, T. Sttzle, *Stochastic Local Search: Foundations and Applications* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004)
37. H. Karl, A. Willig, *Protocols and Architectures for Wireless Sensor Networks* (John Wiley & Sons, UK, 2005)
38. F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. **33**(1–2), 61–106 (2010)
39. P. Merz, Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies (2001)
40. M. Huang, S. Chen, Y. Wang, Minimum cost localization problem in wireless sensor networks, in: Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010, pp. 1–9
41. S. Garca, A. Fernndez, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput. **13**(10), 959–977 (2009)
42. S. Holm, A simple sequentially rejective multiple test procedure. Scand. J. Stat. **6**, 65–70 (1979)
43. M. Aziz, M.-H. Tayarani-N, An adaptive memetic particle swarm optimization algorithm for finding large-scale latin hypercube designs. Eng. Appl. Artif. Intell. **36**, 222–237 (2014)
44. H. Babaei, J. Karimpour, A. Hadidi, A survey of approaches for university course timetabling problem, Computers & Industrial Engineering **86**,43–59 (2015) applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems. doi:10.1016/j.cie.2014.11.010. http://www.sciencedirect.com/science/article/pii/S0360835214003714
45. C.K. Goh, K.C. Tan, A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans. Evolut. Comput. **13**(1), 103–127 (2009)
46. H. R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAW-TIC'06), vol. 1, 2005, pp. 695–701