A Two–Objective Evolutionary Approach based on Topological Constraints for Node Localization in Wireless Sensor Networks

Massimo Vecchio[a,*], Roberto López-Valcarce[a], Francesco Marcelloni[b]

[a] *Departamento de Teoría de la Señal y las Comunicaciones, University of Vigo, C/ Maxwell s/n, 36310 Vigo - SPAIN*
[b] *Dipartimento di Ingegneria dell'Informazione, University of Pisa, Via Diotisalvi 2, 56122 Pisa - ITALY*

Abstract

To know the location of nodes plays an important role in many current and envisioned wireless sensor network applications. In this framework, we consider the problem of estimating the locations of all the nodes of a network, based on noisy distance measurements for those pairs of nodes in range of each other, and on a small fraction of anchor nodes whose actual positions are known *a priori*. The methods proposed so far in the literature for tackling this non-convex problem do not generally provide accurate estimates. The difficulty of the localization task is exacerbated by the fact that the network is not generally uniquely localizable when its connectivity is not sufficiently high. In order to alleviate this drawback, we propose a two–objective evolutionary algorithm which takes concurrently into account during the evolutionary process both the localization accuracy and certain topological constraints induced by connectivity considerations. The proposed method is tested with different network configurations and sensor setups, and compared in terms of normalized localization error with another metaheuristic approach, namely SAL, based on simulated annealing. The results show that, in all the experiments, our approach achieves considerable accuracies and significantly outperforms SAL, thus manifesting its effectiveness and stability.

*Tel: +34 986 818659; FAX: +34 986 812116
Email addresses: massimo@gts.uvigo.es (Massimo Vecchio), valcarce@gts.uvigo.es (Roberto López-Valcarce), f.marcelloni@iet.unipi.it (Francesco Marcelloni)

Keywords: Wireless Sensor Networks, Node Localization,
Range Measurements, Stochastic Optimization,
Multiobjective Evolutionary Algorithms.

1. Introduction

A Wireless Sensor Network (WSN) may consist of hundreds or even thousands of low–cost nodes communicating among themselves [1]. Among classical applications of WSNs, one finds environmental and structural monitoring, event detection, target tracking, etc. In many of these applications, tiny nodes are deployed in an area to be monitored, thus spanning a potentially large geographical region. Each node is a small device that collects information from the surrounding environment through one or more sensors, processes this information locally, and exchanges data through a wireless channel. The small size and low cost of the nodes impose several physical limitations; in particular, they cannot mount powerful microprocessors or large memory devices, thus computational and storage capabilities are tightly constrained. Moreover, they are typically powered by small batteries which in general cannot be easily changed or recharged. A consequence of these limitations is the need to save energy in order to extend the network lifetime [2, 3].

In many envisioned or existent applications, such as environment monitoring, precision agriculture, vehicle tracking, and logistics, knowledge about the location of sensor nodes plays a key role. In addition, location–based routing protocols can save significant energy by eliminating the need for route discovery, and improve caching behavior for applications where requests may be location–dependent. Finally, security can also be enhanced by location awareness (see [4] and the references therein). Although location awareness can be enabled in principle by the use of a Global Positioning System (GPS), this solution is not always viable in practice, as the cost and power consumption of GPS receivers are not negligible. In addition, GPS is not well suited to indoor and underground deployments, and the presence of obstacles like dense foliage or tall buildings may impair the outdoor communication with satellites.

These limitations have motivated alternative approaches to the problem, as reviewed in [5–7], among which *fine–grained* localization techniques may represent the most suitable ones. In these schemes, only a few nodes of the

network (the reference or *anchor* nodes) are endowed with their exact positions through GPS or manual placement, while all nodes are able to estimate their distances to nearby nodes by using any measurement technique. These distance–related techniques include Received Signal Strength (RSS) measurements, Time of Arrival (ToA), Time Difference of Arrival (TDoA), etc. (for a review of these techniques the reader is referred to [6, 7]). Thus, assuming that the coordinates of anchor nodes are known, and exploiting pairwise distance measurements among the nodes, the fine–grained localization problem is to determine the positions of all non–anchor nodes. This task has proved to be rather difficult, due to the following reasons. First, determining the locations of the nodes from a set of pairwise distance measurements is a nonconvex optimization problem. Second, the measurements available to nodes are invariably corrupted by noise. Third, even if the distance measurements were perfectly accurate, a sufficient condition for the topology to be uniquely localizable is not easily identified [8]. We will briefly discuss these issues in the following.

Given a statistical characterization of measurement noise (which will depend on the kind of adopted measurement technique [6]), the most natural path to the localization problem is the Maximum–Likelihood (ML) estimation approach. As stated above, this results in a multivariable nonconvex optimization task, for which three different approaches can be found in the literature: stochastic optimization, multidimensional scaling, and convex relaxation. The first class of techniques attempt to avoid local maxima of the likelihood function via global optimization methods, such as simulated annealing [9]. Multidimensional scaling (MDS) [10, 11] is a *connectivity–based* technique, i.e., it exploits information of "who is within range of whom" in the network, in addition to the distance measurements. This connectivity information imposes additional constraints on the problem, since nodes within range of each other cannot be arbitrarily far apart. The third class of methods relax the original ML formulation in order to obtain a Semi–Definite Programming (SDP) or a Second–Order Cone Programming (SOCP) problem. An approximate solution can be then obtained in a globally optimum fashion with reduced computational effort [8, 12]. Since the relaxation may incur in non–negligible estimation errors [13], additional refinements may be needed, for example via gradient–descent iterations starting from the approximate solution [8].

The main advantage of SDP and SOCP is that the relaxation of the original nonconvex problem reduces the computational load significantly, mak-

ing these methods well suited to large-scale and mobile network localization problems; in these scenarios, the localization algorithm must trade off computation time for some accuracy in the final estimate. On the other hand, there are practical applications of WSNs which do not demand such highly scalable or real–time solutions. Consider, for instance, a precision agriculture application: clearly, it is preferable to spend some additional time in order to obtain an accurate estimate of the node coordinates, rather than e.g. administer a fertilizer to a wrong zone of the monitored field. Moreover, current existent sensor network testbeds are rarely composed by more than one hundred nodes and their mobility is mainly enabled for security and surveillance applications. Motivated by these considerations, we focus on developing localization schemes yielding accurate estimates, having in mind that they may not be well suited to other applications that demand high scalability or require real–time operation. We advocate the use of a stochastic optimization technique, namely Multi–Objective Evolutionary Algorithms (MOEA), in order to solve the original nonconvex problem.

In particular, we propose a two–objective evolutionary algorithm in which the first objective function, referred to as *CF*, is given by the original nonconvex cost (the squared error between the estimated and the corresponding measured inter–node distances). The second objective function, referred to as *CV*, is defined as the sum of neighborhood violations in the candidate topology. This second objective function exploits the connectivity–based *a priori* information about the network, and is especially useful in order to alleviate *localizability* issues. Given a set of data comprised by the set of anchor nodes and the inter–node distance measurements, a network is said to be localizable if there is only one possible geometry compatible with the data. Localizability is a fundamental problem which can be studied within the framework of *rigid graph theory* [14]. If the network is not localizable, then multiple global minima will be present in the cost function, with only one of them corresponding to the actual geometry of the deployment. Thus, in settings which are close to not being localizable, any localization algorithm based on the data above will become extremely sensitive to these false minima of *CF*, resulting in very large location errors [15, 16].

Perhaps the simplest instance of lack of localizability is the so-called *flip ambiguity* phenomenon, illustrated in Fig. 1. As the neighbors of node i (i.e. nodes j, k, l and m) are almost collinear (see the double line in the figure), it is clear that if the location of node i is reflected (*flipped*) with respect to this line to the new position denoted by i', then the new geometry so obtained

is almost compatible with the original inter–node distance measurements (it would be totally compatible if nodes j, k, l and m were perfectly aligned). As mentioned above, connectivity considerations are helpful in order to combat this lack of localizability. Observing Fig. 1, one notices that whereas the flipped position i' is within the communication range of node n, the actual position i is not. In a sufficiently dense network, it can be expected that false minima of CF will result in some connectivity constraints of this sort being violated.

This observation was exploited in the approaches proposed in [9] and in [17]. The former, denoted SAL in the following, exploits two executions of a simulated annealing. In the first execution, the simulated annealing is used to obtain an accurate estimate of the node locations by minimizing CF. At the end of the optimization process, if the neighborhood of a sensor node is correct then it is elevated to an anchor node. Otherwise, it is identified as a non-uniquely localizable node and re-located during the second execution of the simulated annealing (*refinement phase*). In this phase, the simulated annealing minimizes a function which increases CF when the node is placed in a wrong neighborhood. Unlike SAL, which performs two executions of the simulated annealing, the second approach proposed in [17] exploits a unique execution, but introduces a *correction phase* that is executed for a pre-fixed number of iterations. The correction phase is triggered during the main simulated annealing loop whenever the value of the cost function is lower than a manually tuned threshold. During the correction phase an iterative multilateration scheme [18] tries to relocate the nodes placed in the wrong neighborhoods by exploiting the anchor nodes and/or the non–anchor nodes violating the smallest number of neighborhood constraints (thus provisionally elevating them to the status of anchor nodes). Although the idea of enforcing the constraints during the optimization is certainly correct, these implementations with the use of two separate optimizations (in cascade as in [9] or nested as in [17]) may bring to unsatisfactory results. Indeed, as observed by the authors themselves, especially when the node density is low, it is likely for a node to be flipped and still maintain the correct neighborhood. In such situations, the node will be identified as uniquely localizable and thus erroneously elevated to an anchor node. Since both the refinement and the correction phases perform the optimization relying on also these unreliable anchor node positions, these errors may lead the optimization process to generate poorly accurate solutions. On the contrary, since in our MOEA-based approach we exploit only the true anchor nodes

and, at each generation of the algorithm, we evaluate each solution in terms of both accuracy (CF) and constraint violation (CV), the drawback highlighted in the approaches proposed in [9, 17] is considerably mitigated.

The proposed approach is tested with ten different network topologies and three different connectivity ranges and compared in terms of normalized localization error with SAL (for the sake of brevity, we do not discuss the comparison with the approach proposed in [17] since in our experiments it has always achieved solutions characterized by a lower accuracy than the ones obtained by SAL). The results highlight that our approach considerably outperforms SAL, though using a lower number of fitness evaluations.

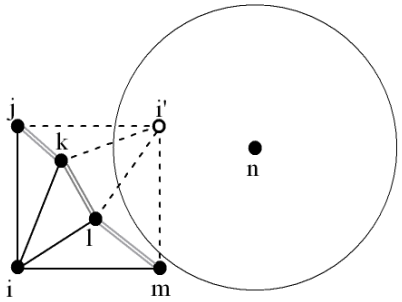


Figure 1: The flip ambiguity problem.

The remainder of this paper is organized as follows. In Section 2, we present the problem formulation. Section 3 introduces our multi–objective evolutionary approach to the problem. The experimental results of our performance analysis are presented in Section 4. Finally, in Section 5 we draw some conclusions and discuss our future work.

2. Problem formulation

In this section we introduce the system model, the objective functions adopted by the evolutionary algorithm and the performance metric used to asses the effectiveness of the approach. Finally, we introduce the geometrical constraints which can be defined on each non–anchor node, exploiting the *a priori* information about the network.

2.1. System model

Consider a WSN with n nodes uniformly deployed in $T = [0, 1] \times [0, 1] \subset \mathbb{R}^2$. Among these, nodes from 1 to m, with $m < n$, are anchor nodes whose

coordinates $\mathbf{p}_i = (x_i, y_i) \in \mathbb{R}^2$, $i = 1,\ldots, m$, are known. In the following, we assume that, if two sensor nodes, say i and j, are within communication range of each other, their inter–node distance d_{ij} can be estimated by using some measurement technique (see Section 1). In the following, we model distances d_{ij} as

$$d_{ij} = r_{ij} + e_{ij} \tag{1}$$

where $r_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|$ is the actual distance between nodes i and j ($\|\cdot\|$ denotes the Euclidean norm), and e_{ij} is the measurement error. Measurement errors are often assumed to be normally distributed. Similar to [9], we assume that these errors follow a zero-mean Gaussian distribution with variance σ^2. Further, we suppose that the random variables e_{ij} and e_{kl} are independent of each other if $(i, j) \neq (k, l)$.

We adopt a simple disk model for network connectivity: nodes i and j can communicate with each other if and only if $r_{ij} \leq R$, where R is the connectivity range. This model is commonly used in the literature, although empirical measurements on real WSNs have shown that it is only an approximation in practice. On the other hand, different connectivity models could be adopted by modifying the geometrical analysis in Section 2.3. We refer to nodes j such that $r_{ij} \leq R$ as *first-level neighbors* of node i. Further, we refer to all nodes j which are not first-level neighbors of node i, but which share at least a first-level neighbor with node i, as *second-level neighbors* of node i. Let

$$N_i = \{j \in 1 \ldots n, j \neq i : r_{ij} \leq R\} \tag{2}$$

$$\overline{N}_i = \{j \in 1 \ldots n, j \neq i : r_{ij} > R\} \tag{3}$$

be the set of the first-level neighbors of node i and its complement, respectively. We assume that sets N_i and $\overline{N}_i$ are known for all $i = 1, \ldots, n$. This is a reasonable assumption, since each node can easily determine which other nodes it can communicate with.

2.2. Objective functions and performance metric

Our goal is to estimate the positions of the non-anchor nodes as accurately as possible. Towards this goal, we aim to concurrently minimize two objective functions. Let $\hat{\mathbf{p}}_i = (\hat{x}_i, \hat{y}_i), i = m + 1, \ldots, n$ be the estimated positions of the non-anchor nodes i. The first objective *CF* is defined as:

$$CF = \sum_{i=m+1}^{n} \left(\sum_{j \in N_i} \left(\hat{d}_{ij} - d_{ij} \right)^2 \right), \tag{4}$$

where $\hat{d}_{ij}$ is the estimated distance between nodes i and j computed as follows:

$$\hat{d}_{ij} = \begin{cases} \sqrt{(\hat{x}_i - x_j)^2 + (\hat{y}_i - y_j)^2} & \text{if node } j \text{ is an anchor,} \\ \sqrt{(\hat{x}_i - \hat{x}_j)^2 + (\hat{y}_i - \hat{y}_j)^2} & \text{otherwise.} \end{cases} \tag{5}$$

Thus, CF is the squared error between the inter–node distances corresponding to the candidate geometry (as given by the estimated positions $\hat{\mathbf{p}}_i$, $i = m+1, \ldots, n$ of the non-anchor nodes and the positions of the anchor nodes) and the measured data.

The second objective function CV counts the number of connectivity constraints which are not satisfied by the current estimated positions of non-anchor nodes. CV is defined as

$$CV = \sum_{i=m+1}^{n} \left(\sum_{j \in N_i} \delta_{ij} + \sum_{j \in \overline{N}_i} (1 - \delta_{ij}) \right), \tag{6}$$

where $\delta_{ij} = 1$ if $\hat{d}_{ij} > R$ and 0 otherwise.

The goodness of an estimate can be evaluated *a posteriori* by using the *normalized localization error* NLE defined as:

$$NLE = \frac{1}{R} \sqrt{\frac{1}{(n-m)} \sum_{i=m+1}^{n} \left((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right)} \times 100\%. \tag{7}$$

2.3. Geometrical constraints

The connectivity ranges and the positions of the anchor nodes determine subspaces of the overall search space where each single non-anchor node can be positioned. These subspaces, which will be expressed by means of geometrical constraints, depend on the type of non-anchor node. We adopt the following classification based on the position of a non–anchor node with respect to anchor nodes:

- *Class 1 node:* a non–anchor node which is first-level neighbor to at least one anchor node.

  If a node belongs to Class 1, then its position must lie within the intersection of the circles of radius R centered in the anchor nodes which it is neighbor to.

- *Class 2 node:* a non–anchor node which is second-level neighbor to at least one anchor node.

  If a node belongs to class 2, then its position must lie within the intersection of the annuli with inner and outer radii R and $2R$, respectively, centered in the anchor nodes which it is second-level neighbor to. Fig. 2 shows two examples of class 2 nodes.

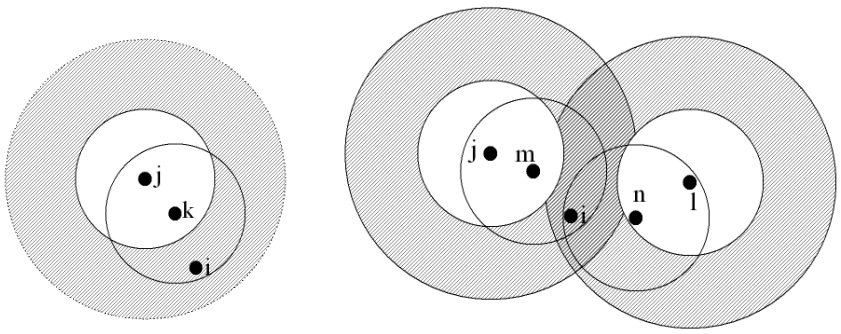
- *Class 3 node:* a non–anchor node which belongs to neither class 1 nor class 2.

  If a node is class 3, then its position must lie outside the union of the circles of radius R centered in all anchor nodes.

The membership of a non-anchor node to one of the three classes allows restricting the space where the node can be located. This information can be exploited both in the generation of the initial population of the MOEA and, during the evolutionary process, to constrain the application of the mating operators. Thus, by avoiding generating solutions which certainly cannot be optimal (since they violate the geometrical constraints determined by the connectivity ranges and by the known anchor node positions), we can speed up the execution of the evolutionary algorithm. Further, these constraints help alleviate the localizability issues discussed in Section 1 and in particular the flip ambiguity phenomenon. This phenomenon is much more likely to occur if the candidate positions of non–anchor nodes are not constrained within the subspace corresponding to its membership class.

3. The Optimization Framework

MOEAs have been investigated by several authors in recent years [19]. Although there exist a number of recently proposed MOEAs with different peculiarities, we have focused our attention on some of the most popular, namely the Strength Pareto Evolutionary Algorithm (SPEA) [20] and its evolution (SPEA2) [21], the Niched Pareto Genetic Algorithm (NPGA) [22], the different versions of the Pareto Archived Evolution Strategy (PAES) [23], and the Non–dominated Sorting Genetic Algorithm (NSGA) [24] and its evolution (NSGA–II) [25]. On the other hand, the main aim of this paper is to demonstrate that the localization problem in wireless sensor networks can be successfully tackled by an MOEA. Thus, we did not investigate whether recent MOEAs might improve the performance, but simply used well-known



(a) node i is neighbor to node k, which in its turn is neighbor to anchor node j.

(b) node i is neighbor to nodes m and n, which in their turns are neighbors to anchor nodes j and l, respectively.

Figure 2: Constraints imposed on *class 2 nodes*.

algorithms. Besides, these algorithms are often used as benchmarks when proposing new MOEAs. After some experimentation, we realized that PAES guaranteed fast convergence towards remarkable Pareto fronts. Further, the PAES evolutionary scheme is very similar to the simulated annealing process exploited by SAL in [9], thus making the comparison quite fair. Indeed, both PAES and SAL exploit a $(1+1)$ optimization scheme: a single random initial solution is generated and mutated (in PAES), or perturbed (in the simulated annealing), in order to obtain a single solution. The key difference between them relies on the generation of the mutated/perturbed solution: in PAES the algorithm exploits the principles of the genetic evolution, while the simulated annealing resembles the temperature–lowering process used in metallurgy to ensure good quality of the final metal cast [26]. We have used the jMetal [27] implementation of PAES for our optimization.

In the following subsections, we describe the chromosome coding, the objective functions, the genetic operators and the PAES algorithm.

3.1. Chromosome coding and objective functions

In our optimization framework each chromosome encodes the positions of all non–anchor nodes in the network. Thus, each chromosome consists of $n - m$ pairs of real numbers (see Fig. 3), where each pair represents the coordinates $\hat{x}$ and $\hat{y}$ of a non–anchor node. The variation range of each coordinate is bounded by the geometrical constraints described in Section 2.3.

We enforce compliance with these constraints in the initial population. Further, whenever mutations are applied during the evolutionary process, only mutated individuals satisfying these constraints are generated.

Figure 3: The chromosome coding.

Each chromosome is associated with a vector of two elements, which represent the values of the two objective functions CF and CV (Eqs. (4) and (6) in Section 2.2).

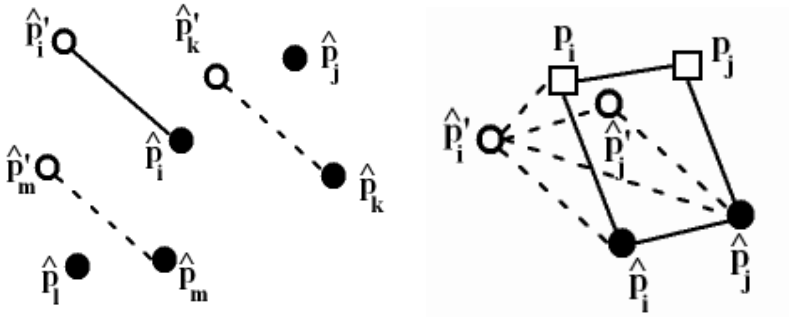
3.2. Genetic operators

PAES exploits only mutation during the evolutionary process. We have defined two mutation operators. The first mutation operator, denoted *node mutation* operator, performs a uniform–like mutation [28]: the position of each non–anchor sensor node is mutated with probability $P_U = 1/(n-m)$. Positions are randomly generated within the geometrical constraints imposed on the specific sensor location.

The second mutation operator, denoted *neighborhood mutation* operator, mutates, with probability $P_U = 1/(n-m)$, the position of each non–anchor sensor node within the geometrical constraints determined for the specific node, but unlike the first operator, it applies the same translation, which has brought the mutated node i from the pre–mutation position to the post–mutation position, to the neighbors of i with probability P_N. Fig. 4(a) shows an example of application of the neighborhood mutation operator. Let i be the sensor node to be mutated. In the figure, we denote with $\hat{\mathbf{p}}_i$ and $\hat{\mathbf{p}}'_i$ the positions of i before and after the application of the mutation operator. The translation applied to node i for shifting this node from $\hat{\mathbf{p}}_i$ to $\hat{\mathbf{p}}'_i$ is also applied to the nodes k and m, which are randomly selected from the set $\{j, k, l, m\}$ of neighbors of i.

The neighborhood mutation was introduced to deal with particular topological configurations such as the one shown in Fig. 4(b). Here, the actual positions $\mathbf{p}_i$ and $\mathbf{p}_j$ of nodes i and j are marked with squares, while the estimated positions are marked with circles. We note that the distance $\|\mathbf{p}_i - \mathbf{p}_j\|$

between the actual positions is similar to the distance $\|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|$ between the estimated positions, thus resulting in a low contribution to CF. Let us suppose that node i is moved from position $\hat{\mathbf{p}}_i$ to position $\hat{\mathbf{p}}'_i$ by applying the node mutation operator. By analyzing the figure, we can realize that, though $\hat{\mathbf{p}}'_i$ is closer to $\mathbf{p}_i$ than $\hat{\mathbf{p}}_i$, the distance $\|\hat{\mathbf{p}}'_i - \hat{\mathbf{p}}_j\|$ between the estimated positions is much larger than that between the actual positions, thus resulting in a considerable increase of CF. This increase will probably lead to discarding the solution with i', even though this solution is certainly better than the one with i. On the other hand, applying the neighborhood mutation, node j would have been translated, with a certain probability, together with i into j' and i', respectively, as shown in Fig. 4(b), thus leaving the distances between estimated and actual positions unchanged. It follows that the solution with i' and j' has the same contribution to CF (as far as these two nodes are concerned) as the solution with i and j, and thus the previous problem is avoided. We experimentally verified that this mutation operator performs better when not all the neighbors are translated with the mutated node. Indeed, if the estimated positions of the neighbors of a mutated node are very close to the actual positions, then translating all of them would considerably worsen the solution. Thus, the translation is applied only to a randomly chosen subset of neighbors.



(a) An example of application of the neighborhood mutation

(b) Impact of the application of the neighborhood mutation on the CF objective

Figure 4: The behavior of the neighborhood mutation operator.

3.3. PAES

The PAES algorithm was introduced in [23] and probably represents the simplest possible nontrivial algorithm capable of generating diverse solutions in the Pareto optimal set. Further, PAES is characterized by a lower computational complexity than traditional niching methods [23, 29]. PAES consists of three parts: the candidate solution generator, the candidate solution acceptance and the non-dominated solution archive. The candidate solution generator maintains a single current solution c, and, at each iteration, produces a new solution m from c, by using a mutation operator. The candidate solution acceptance compares m with c. Three different cases can arise:

1. c dominates m: m is discarded;
2. m dominates c: m is inserted into the archive and possible solutions in the archive dominated by m are removed; m replaces c in the role of current solution;
3. neither condition is satisfied: m is added to the archive only if it is dominated by no solution contained in the archive; m replaces c in the role of current solution only if m belongs to a region with a crowding degree smaller than, or equal to, the region of c.

The crowding degree is computed by firstly dividing the space where the solutions of the archive lie into a number ($numReg$) of equally sized regions and then by counting the solutions that belong to the regions. The number of these solutions determines the crowding degree of a region. This approach tends to prefer solutions which belong to poorly crowded regions, so as to guarantee a uniform distribution of the solutions along the Pareto front.

PAES terminates after a given number $maxEvals$ of evaluations. The candidate solution acceptance strategy generates an archive which contains only non-dominated solutions. On PAES termination, the archive includes the set of solutions which are an approximation of the Pareto front. At the beginning, the archive is empty and the first current solution is randomly generated.

In Fig. 5, we show the PAES pseudocode. Here, the operator $\preceq$ indicates dominance (*i.e.*, $m \preceq c$ means that the mutated solution m dominates the current solution c). At the beginning an empty archive of size *archiveSize* is allocated (line 2); an initial random solution obeying to the topological constraints is generated (line 4), evaluated in terms of CF and CV (line 5) and added to the archive (line 6). In the loop (lines 7-23), the new solution m is generated by applying the first mutation operator with probability P_M. If

this operator is not applied, then the second mutation operator is executed. If the new solution dominates the current one, then it substitutes the latter in the archive (lines 15-16) becoming the new current solution; otherwise, if it is dominated by the current solution or by any solution contained in the archive, then the mutated solution is discarded (line 18); if none of the conditions mentioned is met (*i.e.*, the mutated solution is not dominated by any member of the current archive of solutions) the function *applyTest* is called (line 20) in order to decide whether the mutated solution has to be included in the archive (and eventually which solution has to be discarded to make place for m, if the archive is full), and which solution will become the current solution for the next iteration. For more details on PAES the reader should refer to [23, 29].

4. Simulation results

In this section we show the effectiveness of the proposed two–objective evolutionary algorithm in tackling the fine–grained localization problem in WSNs. We have built different network topologies by uniformly placing 200 nodes in $T = [0, 1] \times [0, 1] \subset \mathbb{R}^2$. We have fixed the percentage of anchor nodes to 10% (thus each topology consists of 20 anchor nodes and 180 non–anchor nodes). Further, we have set the values of the connectivity range R to 0.13, 0.15 and 0.17. The distance measurements between neighboring nodes are generated according to the model (1), i.e. $d_{ij} = r_{ij} + e_{ij}$. We assume that these distance estimates are derived from RSS measurements, which are commonly affected by log-normal shadowing, such that the standard deviation of the errors is proportional to the actual range r_{ij} [6]; i.e. the variance of e_{ij} is given by $\sigma^2 = \alpha^2 r_{ij}^2$. A value of $\alpha = 0.1$ is used in the simulations.

For each value of R, 10 random network topologies are generated. We first characterize the generated topologies in terms of nodes' neighborhood cardinalities, number of anchor nodes in non–anchor nodes's neighborhood and classification of non–anchor nodes in terms of the classes introduced in Section 2.3. Then, we measure the performance of the proposed algorithm in terms of normalized localization errors. Finally, we compare our results with the ones obtained by the SAL algorithm which exploits a different stochastic approach, based on simulated annealing, to solve the localization problem and to alleviate the flip ambiguity threat.

```
1: procedure PAES(archiveSize, maxEvals, numReg, P_M, P_N)
2:     archive[archiveSize] ← []
3:     t ← 0
4:     c ← generateSolution()
5:     evaluateFitness(c)
6:     addSolution(archive)
7:     while t < maxEvals do
8:         r ← uniformRandom(0, 1)
9:         if r ≤ P_M then
10:            m ← applyNodeMutation(c)
11:        else
12:            m ← applyNeighborhoodMutation(c, P_N)
13:        end if
14:        if m ⪯ c then
15:            replace(c, m)
16:            addSolution(m)
17:        else if c or any member of the archive ⪯ m then
18:            discard(m)
19:        else
20:            c ← applyTest(c, m, archive, numReg)
21:        end if
22:        t ← t + 1
23:    end while
24: end procedure
```

Figure 5: The PAES algorithm.

4.1. Topology characterization

Fig. 6 shows the average percentages of nodes in the 10 random network topologies versus the neighborhood cardinality for the different values of R. For instance, for $R = 0.13$, about 12% of the network nodes (anchor and non–anchor nodes) have 10 neighbors. We note that for $R = 0.13$, 0.15 and 0.17, no node has more than 20, 26 and 29 neighbors, respectively. Also, note that increasing R induces an increase in the neighborhood cardinality, as expected. For $R = 0.13$, 0.15 and 0.17, the highest percentages of nodes correspond to values of cardinality around 10, 13 and 17, respectively.

Fig. 7 shows the average percentages of non–anchor nodes in the 10 random network topologies versus the number of anchor nodes in their neigh-

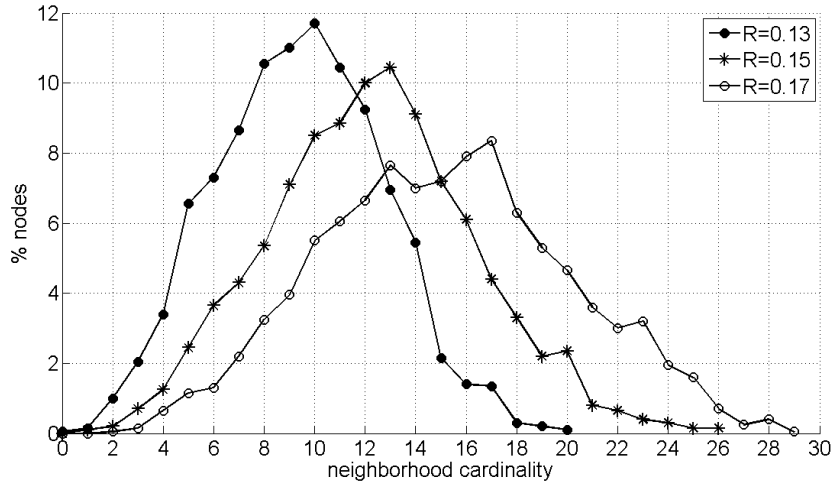


Figure 6: Average percentage of nodes versus neighborhood cardinality for different values of R.

borhood, for the different values of R. Note that, even when $R = 0.17$, the number of non–anchor nodes with no anchor neighbor is not negligible and the average percentage of non–anchor nodes with 3 or more anchor neighbors is significantly low. This highlights the difficulty of the localization problem for these network topologies.

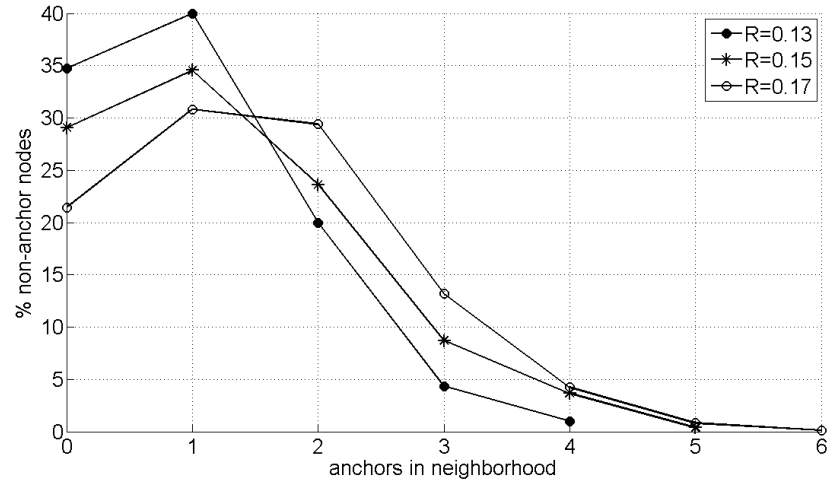


Figure 7: Average percentage of non-anchor nodes versus the number of anchor nodes in their neighborhood for different values of R.

Fig. 8 shows how the non–anchor nodes are distributed in the three classes introduced in Section 2.3 for all the network topologies generated. Each stacking bar in the figure, labeled as TOP0,. . . ,TOP9, corresponds to a different topology.

As expected, increasing R results in a larger percentage of non–anchor nodes in class 1 (and hence smaller percentages of nodes in classes 2 and 3). The average percentage of non–anchor nodes in classes 1, 2 and 3 are, respectively, 65%, 30% and 5% for $R = 0.13$; 71%, 24% and 5% for $R = 0.15$; and 79%, 20% and 1% for $R = 0.17$.

4.2. Results of our approach

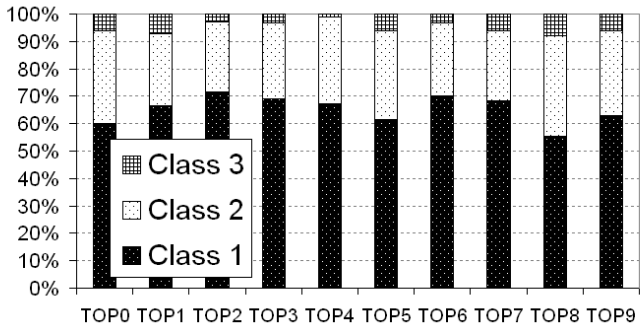
We performed 30 trials of PAES for each different network topology and for each value of R. Table 1 summarizes the values of the parameters used in the execution of PAES.

Parameter	Value
Archive size	20
Number of regions	5
Number of fitness evaluations	400000
Node mutation probability (P_M)	0.9
Node rigid translation probability (P_N)	0.3

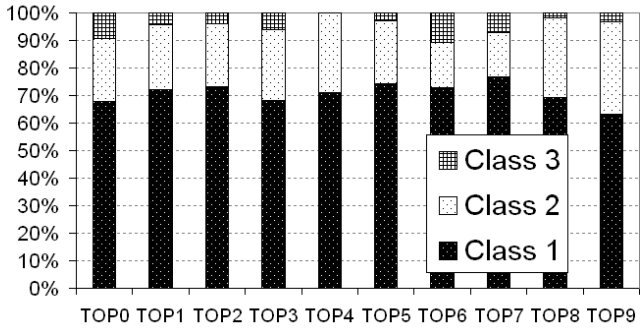
Table 1: parameter setup of PAES.

Fig. 9 shows an example of Pareto front approximation obtained by applying the PAES algorithm on a network topology generated for $R = 0.15$. Each solution in the front encodes the estimated positions of the 180 non–anchor nodes, and is associated with a different trade–off between the two objectives CF and CV.

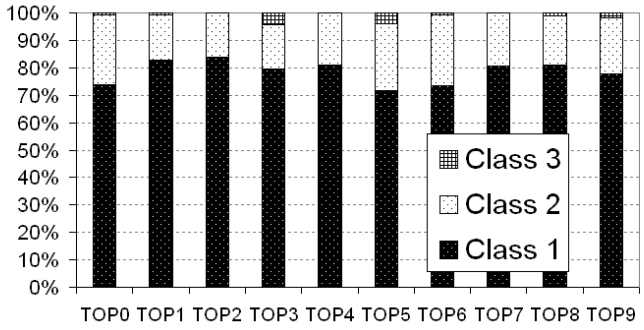
Once generated the Pareto front approximation, we have to choose a solution. In our experiments, we verified that the variation interval of CF for the solutions on the final Pareto front approximation is quite small. Thus, we can assume that each solution on the Pareto front can be acceptable with respect to the CF objective. In order to validate this assumption, we have performed a two–sided rank sum test (Wilcoxon test) by selecting from each final archive the solutions characterized by the minimum value of CV and the minimum value of CF. Figs. 10-12 show the boxplots of the NLE values obtained for the 10 network topologies and for the three connectivity ranges. Here, the lowest and the largest values of NLE are represented as



(a) $R = 0.13$



(b) $R = 0.15$



(c) $R = 0.17$

Figure 8: Average distribution of the non–anchor nodes in the three classes introduced in Section 2.3.

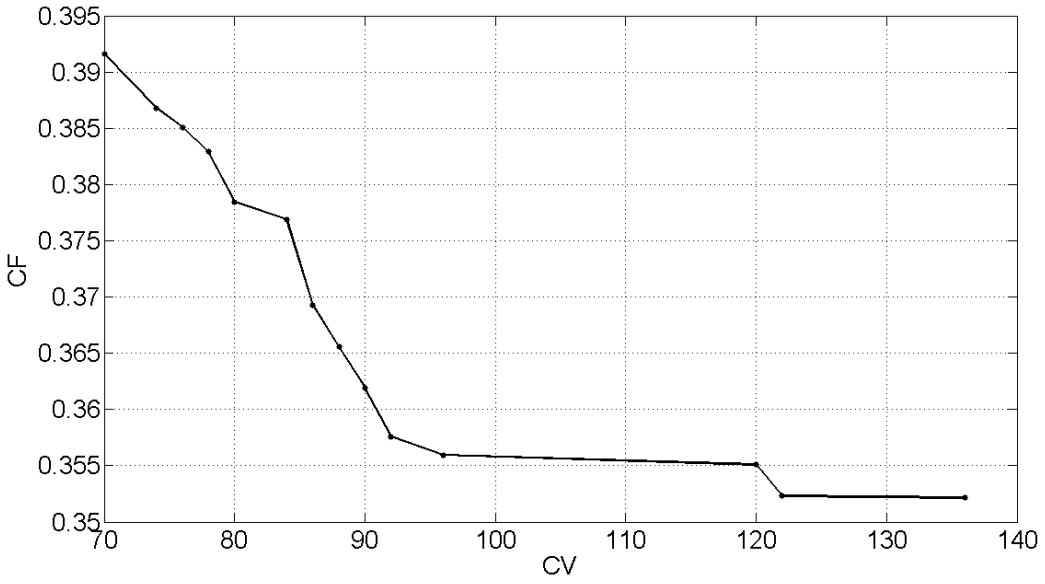


Figure 9: Final front of non–dominated solutions.

whiskers, the lower and upper quartiles are shown with a box, the median is represented by a line and observations that may be considered outliers are possibly marked with asterisks [30]. Further, mCV and mCF stand for the sample distribution of the individuals characterized by the lowest value of CV and CF in the final archive, respectively. Finally, we exploit the background color of the labels used to identify the specific network topology to indicate whether the null hypothesis is rejected or not: if the background color is gray, the null hypothesis cannot be rejected; otherwise, if the background color is white, the null hypothesis can be rejected. We observe that, for each network topology and for each connectivity range, the null hypothesis cannot be rejected. Thus, we can conclude that the distributions are statistically equivalent with a confidence level of 95%. Further, we can also note that the value of NLE decreases as R increases since, as shown in Fig. 8, a larger number of non–anchor nodes have anchor nodes as neighbors. Finally, we can also observe that the results achieved by our algorithm are quite stable. Indeed, the number of outliers is rather low, the median is rather well–balanced between the upper and lower quartiles and the whiskers are not too far from the upper and lower quartiles.

Since no statistical difference exists in terms of NLE among the solutions

in the final Pareto front approximation, each solution can be actually selected in order to perform a comparison with the SAL algorithm. For the sake of brevity, we have decided to use the solution characterized by the lowest value of CV.

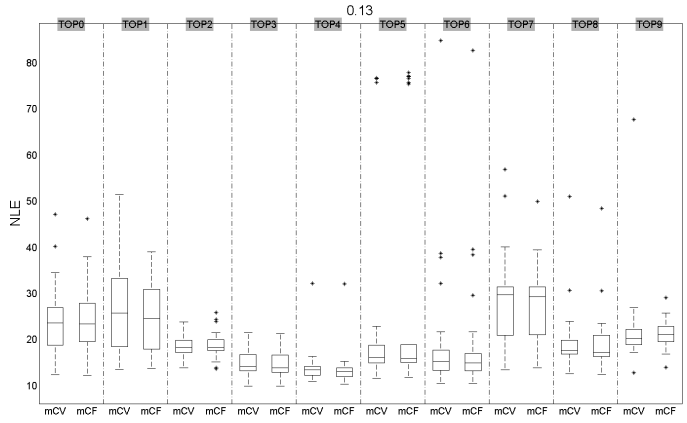


Figure 10: Boxplots of the NLE values obtained for the ten network topologies with connectivity range $R = 0.13$.

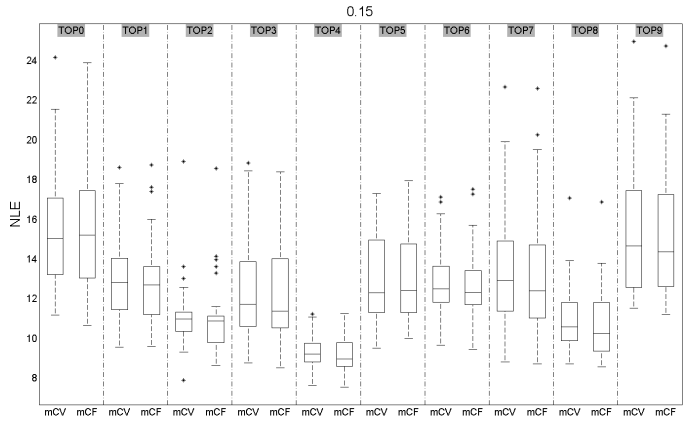


Figure 11: Boxplots of the NLE values obtained for the ten network topologies with connectivity range $R = 0.15$.

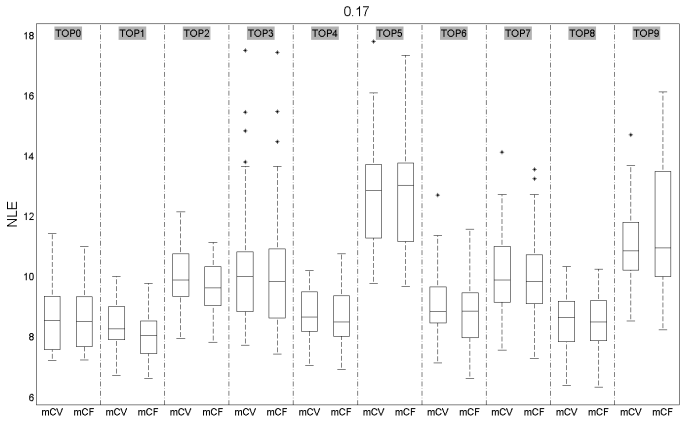


Figure 12: Boxplots of the NLE values obtained for the ten network topologies with connectivity range $R = 0.17$.

4.3. Comparison with the SAL algorithm

In this section, we compare the results obtained by the solution characterized by the lowest value of CV among the solutions in the final Pareto front approximation with those obtained by the SAL algorithm.

SAL is a stochastic optimization scheme which has proved to be very effective in solving the WSN localization problem [9]. We will not perform comparisons with non–stochastic methods since these methods are designed for providing fast, though not very accurate, solutions. Thus, these methods relax the original nonconvex problem so as to speed up the computation, but generally cannot achieve values of NLE comparable to the ones of the solutions generated by our approach. On the other hand, in [9] the authors have already experimentally proved that SAL considerably outperforms a non–stochastic method, namely SDP.

The SAL algorithm tackles the fine–grained localization problem in WSNs using a 2–phase optimization task. In the first phase, a simulated annealing approach is applied to estimate the non–anchor node positions so as to minimize the cost function defined in eq. (4). At the end of the first phase (after a maximum number of iterations, or when the control temperature has gone over a threshold value), the following check is performed: if the neighborhood of a sensor node is correct then it is elevated to an anchor node, otherwise it is identified as non–uniquely localizable node and placed in the set of nodes to be re–localized using the refinement phase. In this phase, another simu-

lated annealing is performed on the non–uniquely localizable nodes in order to minimize a new cost function (CF_{REF}) defined as:

$$CF_{REF} = \sum_{i=m+1}^{n} \left(\sum_{j \in N_i} \left(\hat{d}_{ij} - d_{ij} \right)^2 + \sum_{\substack{j \in \overline{N}_i \\ \hat{d}_{ij} < R}} \left(\hat{d}_{ij} - R \right)^2 \right) \tag{8}$$

where N_i and $\overline{N}_i$ are defined in formulas 2 and 3. If a node $j \in \overline{N}_i$ has been estimated such that $\hat{d}_{ij} < R$, then it is assumed to be placed in the wrong neighborhood; the minimum error due to the misplacement is computed as $(\hat{d}_{ij} - R)$ and included in the cost computation as an extra additive term.

In Fig. 13, we show the pseudocode of the algorithm executed in the two phases of the SAL approach: the unique difference between the two phases is the fitness function used in line 13. Indeed, the first phase employs eq. (4) as fitness function while the refinement phase uses eq. (8). After the initialization (lines 2-5), SAL executes the main loop (lines 6-30). At each iteration, the control temperature T and the perturbation entity ΔD are slowly decreased according to the rules in lines (28-29) and $N * P * Q$ function evaluations are computed (where P and Q are given parameters, and N is the number of non–anchor nodes to be localized) by systematically perturbing the estimated non–anchor nodes locations (lines 7-27). If the perturbed position induces a lower value in the fitness function, then the solution is accepted (*downhill*, lines 15-17), otherwise the solution with the increased cost is accepted with a certain probability (*uphill*, lines 18-24). It can be noticed that the SAL algorithm requires a fine tuning of the several simulation parameters, in order to meet a good trade–off between the accuracy of the results and the execution time. Since in [9] the complete parameter setup was not provided, we have used the simulation setup provided in [17], after having verified its effectiveness. Table 2 summarizes the values of the parameters used in SAL.

We observe that, knowing the total number of non–anchor nodes to be localized, after fixing the values of T_0, T_f and α, one can derive the total number of iterations that SAL will execute during the optimization phase. Since for each iteration, $N*P*Q$ fitness evaluations are performed, one could also compute the total number of fitness evaluations carried out during the optimization process. However, when SAL enters the second optimization phase, an unpredictable number of non–anchor nodes will be elevated to

```
1: procedure SAL(#nonAnchorNodes, T_0, T_f, ΔD_0, α, β)
2:     N ← #nonAnchorNodes
3:     T ← T_0
4:     ΔD ← ΔD_0
5:     CF_old ← ∞
6:     while T ≥ T_f do
7:         for q ← 1 to Q do
8:             aPermutation ← randPerm(N)
9:             for i ← 1 to N do
10:                curIndex = aPermutation[i]
11:                for p ← 1 to P do
12:                    perturbPosition(curIndex, ΔD)
13:                    CF_new ← evaluateFitness()
14:                    ΔCF ← CF_new − CF_old
15:                    if ΔCF ≤ 0 then
16:                        acceptPerturbation(i)
17:                        CF_old = CF_new
18:                    else
19:                        r = uniformRandom(0, 1)
20:                        if r ≤ exp(−ΔCF/T) then
21:                            acceptPerturbation(i)
22:                            CF_old = CF_new
23:                        end if
24:                    end if
25:                end for
26:            end for
27:        end for
28:        T ← α · T
29:        ΔD ← β · ΔD
30:    end while
31: end procedure
```

Figure 13: The algorithm executed in the two phases of the SAL approach.

the status of anchor–nodes and not relocated during the refinement phase. Thus, the value of N cannot be determined *a priori* for the refinement phase. Using the parameters in Table 2, in the 30 trials executed for each network topology and for each connectivity range, SAL computed on average 710000

Parameter	Value
T_0	0.1
T_f	10^{-11}
P	10
Q	2
D_0	0.1
α	0.80
β	0.94

Table 2: parameter setup of SAL.

fitness evaluations. By analyzing the values of the parameters used in PAES and shown in Table 1, we realize that the number of fitness evaluations is higher for SAL and for PAES.

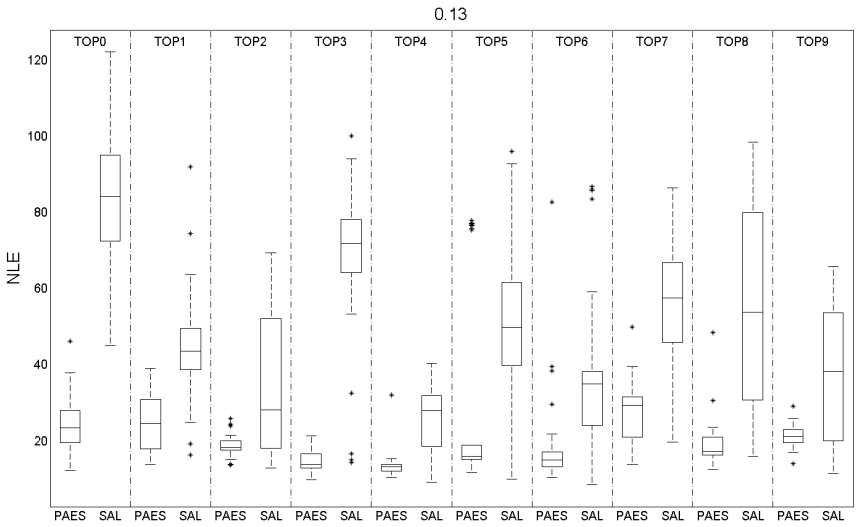


Figure 14: Boxplots of the NLE values obtained by PAES and SAL for the 10 network topologies using a connectivity range $R = 0.13$.

Figs. 14-16 show the boxplots of the NLE values obtained by PAES and SAL on the 10 network topologies for the three connectivity ranges. We can observe that the results achieved by the SAL algorithm are less stable than the ones achieved by PAES. Indeed, the ranges of variation of the values of NLE are much wider. For verifying whether the distributions are statistically different, we have performed a two–sided rank sum test (Wilcoxon

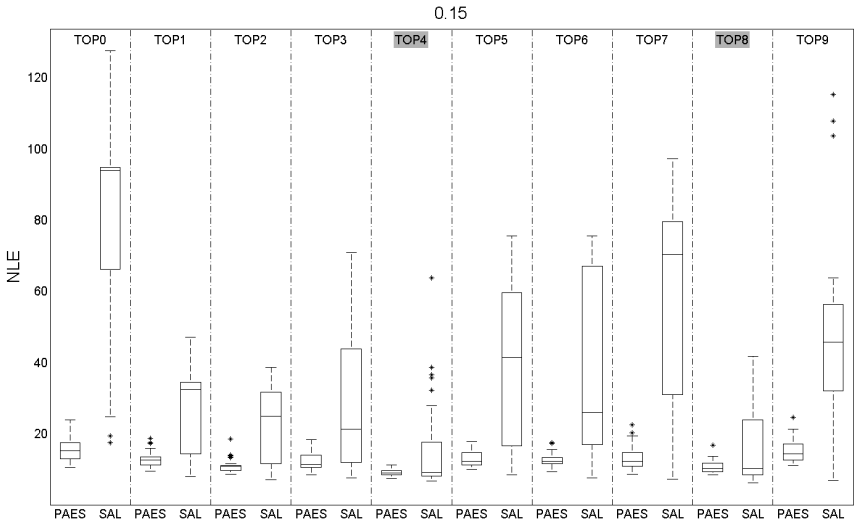


Figure 15: Boxplots of the NLE values obtained by PAES and SAL for the 10 network topologies using a connectivity range $R = 0.15$.

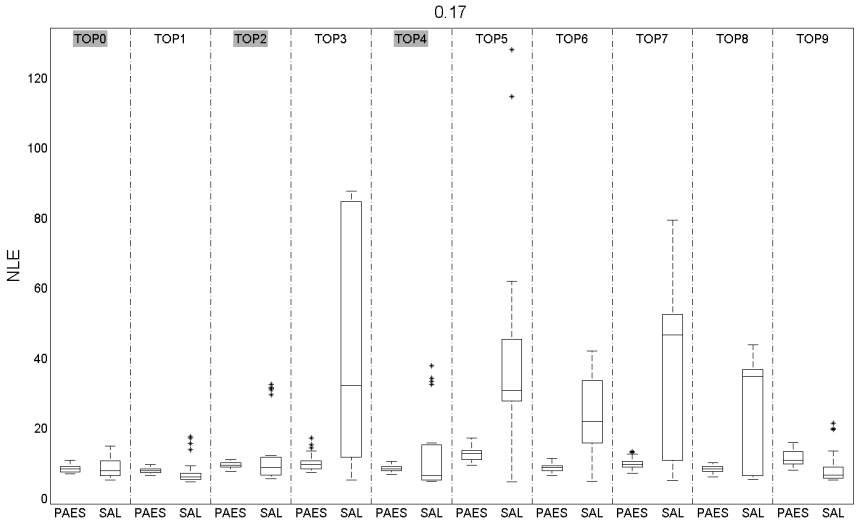


Figure 16: Boxplots of the NLE values obtained by PAES and SAL for the 10 network topologies using a connectivity range $R = 0.17$.

test). Again, we exploit the background color of the labels used to identify the specific network topology to indicate whether the null hypothesis is rejected or not: if the background color is gray, the null hypothesis cannot be rejected; otherwise, if the background color is white, the null hypothesis can be rejected. We observe that only for topologies TOP1 and TOP9 with connectivity range $R = 0.17$ SAL outperforms PAES. For all the remaining topologies PAES outperforms SAL, except for TOP4 and TOP8 with $R = 0.15$ and TOP0, TOP2 and TOP4 with $R = 0.17$ where the null hypothesis cannot be rejected. Since PAES uses a lower number of fitness evaluations, this result testifies the effectiveness of our approach.

5. Conclusions

In this paper we have proposed a two–objective evolutionary algorithm able to accurately solve the fine–grained localization problem in WSNs. The problem is not new in the literature, since several techniques have been proposed in the last decade. The novelty of the approach relies on a better exploitation of the connectivity graph so as to define topological constraints to be used as a second objective function in a multi–objective optimization framework. The topological constraints define zones of the space where each sensor can or cannot be located, thus reducing the search space of the evolutionary algorithm and contextually the chance of ambiguously flipping nodes' locations. We have shown that the proposed approach is able to solve the localization problem with high accuracy for a number of different topologies and different connectivity ranges. Further, we have discussed how our approach considerably outperforms a recently proposed localization algorithm based on simulated annealing.

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393–422.

[2] F. Marcelloni, M. Vecchio, Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization, Information Sciences 180 (10) (2010) 1924–1941.

[3] S. Croce, F. Marcelloni, M. Vecchio, Reducing power consumption in wireless sensor networks using a novel approach to data aggregation, The Computer Journal 51 (2) (2008) 227–239.

[4] L. Hu, D. Evans, Localization for mobile sensor networks, in: MobiCom '04: Proc. of the 10th Int. Conf. on Mobile Computing and Networking, 2004, pp. 45–57.

[5] L. M. R. Peralta, Collaborative localization in wireless sensor networks, in: SENSORCOMM 07: Proc. of the 2007 Int. Conf. on Sensor Technologies and Applications, 2007, pp. 94–100.

[6] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, III, R. L. Moses, N. S. Correal, Locating the nodes: cooperative localization in wireless sensor networks, IEEE Signal Processing Mag. 22 (4) (2005) 54–69.

[7] G. Mao, B. Fidan, B. D. O. Anderson, Wireless sensor network localization techniques, Computer Networks 51 (10) (2007) 2529–2553.

[8] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, T.-C. Wang, Semidefinite programming approaches for sensor network localization with noisy distance measurements, IEEE Trans. Autom. Sci. Eng. 3 (4) (2006) 360–371.

[9] A. A. Kannan, G. Mao, B. Vucetic, Simulated annealing based wireless sensor network localization with flip ambiguity mitigation, in: Proc. of the 63-rd IEEE Vehicular Technology Conference, 2006, pp. 1022–1026.

[10] X. Ji, H. Zha, Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling, in: Proc. of IEEE INFOCOM, Vol. 4, 2004, pp. 2652–2661.

[11] J. A. Costa, N. Patwari, A. O. Hero, III, Distributed weighted–multidimensional scaling for node localization in sensor networks, ACM Trans. on Sensor Networks 2 (1) (2006) 39–64.

[12] P. Tseng, Second–order cone programming relaxation of sensor network localization, SIAM J. on Optimization 18 (1) (2007) 156–185.

[13] Z. Wang, S. Zheng, Y. Ye, S. Boyd, Further relaxations of the semidefinite programming approach to sensor network localization, SIAM J. Optim. 19 (2) (2008) 655–673.

[14] R. Connelly, Generic global rigidity, Discrete Comput. Geometry 33 (4) (2005) 549–563.

[15] A. A. Kannan, B. Fidan, G. Mao, Analysis of flip ambiguities for robust sensor network localization, IEEE Trans. Vehicular Technology 59 (4) (2010) 2057–2070.

[16] S. Severi, G. Abreu, G. Destino, D. Dardari, Understanding and solving flip-ambiguity in network localization via semidefinite programming, in: GLOBECOM'09: Proc. of the 28th IEEE Conf. on Global Telecommunications, 2009, pp. 3910–3915.

[17] E. Niewiadomska-Szynkiewicz, M. Marks, Optimization schemes for wireless sensor network localization, Applied Mathematics and Computer Science 19 (2) (2009) 291–302.

[18] A. Savvides, C.-C. Han, M. B. Srivastava, Dynamic fine–grained localization in ad-hoc networks of sensors, in: MobiCom '01: Proc. of the 7th Int. Conf. on Mobile Computing and Networking, 2001, pp. 166–179.

[19] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, IEEE Trans. Evol. Comput. 8 (2) (2000) 173–195.

[20] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.

[21] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization, in: K. Giannakoglou, et al. (Eds.), Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, 2002, pp. 95–100.

[22] J. Horn, N. Nafpliotis, D. E. Goldberg, A Niched Pareto Genetic Algorithm for Multiobjective Optimization, in: Proc. of the 1st IEEE Conference on Evolutionary Computation, Vol. 1, 1994, pp. 82–87.

[23] J. D. Knowles, D. W. Corne, Approximating the nondominated front using the Pareto Archived Evolution Strategy, IEEE Trans. Evol. Comput. 8 (2) (2000) 149–172.

[24] N. Srinivas, K. Deb, Multiobjective optimization using Nondominated Sorting in Genetic Algorithms, IEEE Trans. Evol. Comput. 2 (3) (1994) 221–248.

[25] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[26] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by Simulated Annealing, Science 220 (4598) (1983) 671–680.

[27] J. J. Durillo, A. J. Nebro, E. Alba, The jMetal framework for multiobjective optimization: Design and architecture, in: CEC '10: Proc. of the IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.

[28] Z. Michalewicz, Genetic algorithms + data structures = evolution programs, 2nd Edition, Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[29] C. A. C. Coello, G. B. Lamont, D. A. V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[30] J. W. Tukey, Exploratory Data Analysis, Addison-Wesley, 1977.