Applied Soft Computing 11 (2011) 1223–1240

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Location discovery in Wireless Sensor Networks using metaheuristics

Guillermo Molina*, Enrique Alba

Dept. de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Málaga 29071, Spain

ARTICLE INFO

Article history:
Received 8 June 2009
Accepted 27 February 2010
Available online 7 March 2010

Keywords:
Wireless Sensor Networks
Location discovery
Metaheuristics

ABSTRACT

Wireless Sensor Networks (WSN) monitor the physical world using small wireless devices known as sensor nodes, with high precision and in real time, without the intervention of a human operator. Location information plays a critical role in many of the applications where WSN are used. Though a simple and effective solution could be to equip every node with self-locating hardware such as a GPS, the resulting cost renders such a solution unefficient. A widely used self-locating mechanism consists in equipping a small subset of the nodes with some GPS-like hardware, while the rest of the nodes employ reference estimations (received signal strength, time-of-arrival, etc.) in order to determine their locations. The task of determining the node locations using node-to-node distances combined with a set of known node locations is referred to as location discovery (LD). The main difficulty found in LD is the presence of distance estimation errors, which result in node positioning errors. We describe in this work an error model for the estimations, and propose a two-stage search procedure that combines minimization of an error norm function with maximization of a maximum likelihood function to solve the problem. We perform an empirical study of the performance of several variants of the guiding functions, and several metaheuristics used to solve real LD problem instances. Finally, we test our proposed technique against the single phase techniques in order to evaluate its performance.

1. Introduction

Wireless Sensor Networks (WSN) have become a hot topic in research [5]. Their capabilities for monitoring large areas, accessing remote places, real-time reacting, and relative ease of use have brought scientists a whole new horizon of possibilities. WSN have so far been employed in military activities such as reconnaissance, surveillance [1], and target acquisition [19], environmental activities such as forest fire prevention [17], or civil engineering such as structural health measurement [23]. Their uses increase by the day and their potential applications seem boundless. The wide variety of applications results in a wide variety of networks bearing different constraints and having different features, yet most of these networks share some common issues that allow them to be treated homogenously.

One of the key features in a WSN is the location awareness in the information. When a WSN is deployed to monitor a wide area (for instance fire prevention or surveillance) the value of the data retrieved requires associated geographical information. If action is to be taken in response to a detection (send a helicopter, send a squad), it is necessary to know *where* the detected event is located. Therefore, we need the sensor nodes to know their location information.

A simple way to ensure this is to equip every node in the WSN with GPS. This solution is very likely to be unfeasible due to economical issues. The GPS hardware is rather expensive, WSN typically contain very high amounts of nodes, and are generally cost constrained. Therefore, only a small subset of the network can affordably be equipped with GPS; such nodes are called *anchor* nodes, or beacons. An automatic localization process is required for the rest of the nodes in the network, using the anchor nodes as reference points. This process is commonly known as location discovery (LD). For this, distance measurements between sensor nodes are employed, and a multilateration system can be defined. There exist several techniques to estimate the distance between two sensor nodes, some of which will be described in higher detail further on.

However, the distances measured among the nodes in a WSN usually contain errors. These errors range from slight errors to large ones, and are difficult to characterise by a standard model such as Gaussian. These distance measurement errors may result in severe location errors that degrade the WSN performance, and therefore should be taken into account during the LD. We will analyze and apply two of the most popular fitness functions used for LD: the L_1 error norm function and the Likelihood function. We will propose a new search process based on combining the two fitness functions, and compare it against the results obtained with either

* Corresponding author. Tel.: +34 952 13 33 03.
E-mail addresses: guillermo@lcc.uma.es (G. Molina), eat@lcc.uma.es (E. Alba).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.02.021

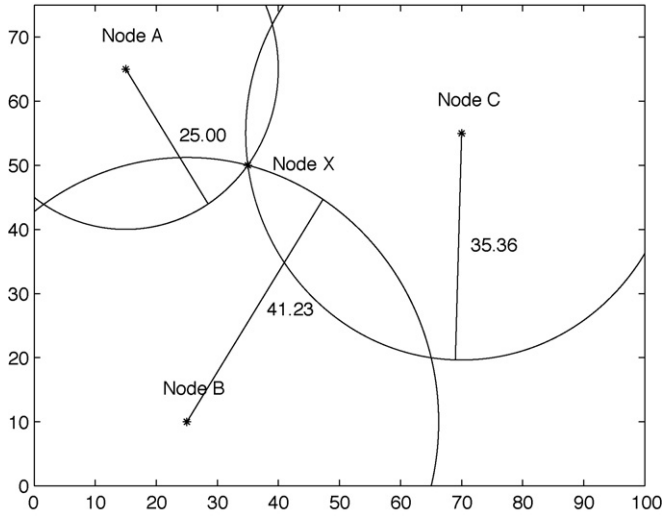

Fig. 1. Simple trilateration system: nodes A, B, and C have positioning knowledge, node X can then calculate its position by measuring the distances to nodes A, B and C.

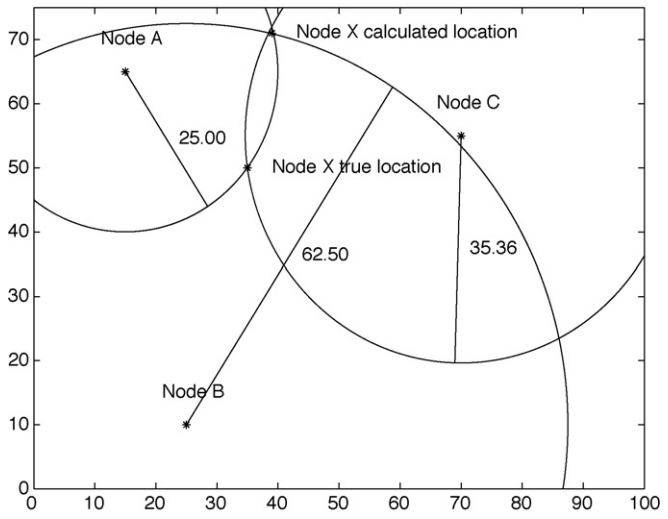

Fig. 2. An error in the distance measured to node B causes node X to calculate a wrong location.

single fitness function. We will propose some heuristic additions to the fitness function and test their impact on the results produced. We will compare three popular metaheuristic algorithms, Simulated Annealing, Genetic Algorithm and Particle Swarm Optimization, and compare their results. We will also study the effect of the number of beacons in the WSN.

The rest of the paper is organized as follows. In Section 2 the location discovery problem is explained and formulated. The existing work on the field is commented in Section 3. Section 4 will focus on the measurement model. The fitness function will be discussed in Section 5, and the proposed optimization techniques will be presented in Section 6. Then in Section 7 the experiments performed and the results obtained are shown. Finally some conclusions are drawn in Section 8.

2. Location discovery

The LD problem is an optimization problem where a set of element locations has to be determined in a manner such that the location error is minimized. It is widely considered one of the fundamental tasks in a WSN [11]. The LD affects many aspects of a sensor networks, from application (context-aware monitoring) to communication protocols (geographical routing [8]). The location discovery problem has been proven to be NP-complete [18].

A simple trilateration system is shown in Fig. 1. Assume nodes A, B and C know their position coordinates, and node X measures the distances separating itself from each of them. Then, node X can determine its own location as is shown in Fig. 1.

We will first give some short definitions. We can assume that locations are given in 2D or 3D. For the sake of simplicity we will assume 2D in the following.

Let a and b be two points whose – presumably unknown – coordinates are (x_a, y_a) and (x_b, y_b) respectively, and (x'_a, y'_a) and (x'_b, y'_b) their estimated coordinates (a.k.a. estimated locations, obtained through LD). We define the following:

- Real distance $d_{a,b} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$
- Measured or estimated distance $\delta_{a,b}$ (obtained by some measuring technique)
- Calculated distance$c_{a,b} = \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2}$
- Measured distance error $\epsilon_{a,b} = \delta_{a,b} - d_{a,b}$
- Calculated distance error $\epsilon'_{a,b} = c_{a,b} - d_{a,b}$
- Location errors $\epsilon_a = \sqrt{(x_a - x'_a)^2 + (y_a - y'_a)^2}$, $\epsilon_b = \sqrt{(x_b - x'_b)^2 + (y_b - y'_b)^2}$

Note that the final objective of LD is to minimize the *location errors* (ϵ_a and ϵ_b). Fig. 2 shows a case of LD where errors exist in the measurements: the distance from node B to node X is measured as 62.5 when its real value is 41.23. This causes node X to calculate an erroneous location. The location error is the distance from node X's real location to its calculated location.

Our formulation of the problem is as follows: given a set S of N nodes s_i, $1 \le i \le N$, a subset of nodes s_j, $1 \le j \le K < N$ with previously known locations (anchor nodes), and a set of distance estimations $\delta_{i,j}$, $1 \le i, j \le N$, $i \ne j$, we have to determine the locations of every node s_l, $K < l \le N$ such that the location error is minimized.

When the distance estimations equal the real distances $\delta_{i,j} = d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (assuming 2D), the location discovery problem can be formulated as an ideal trilateration problem. In this case, as long as the scenario is well-defined, the location discovery can be solved to optimality. Since the estimated distances are the real distances, finding the locations that produce calculated distances equal to the measured distances will solve the problem.

However, in a general scenario, estimated distances do not equal real distances, and the distance errors have significant values. This causes a number of problems for location discovery. The main one is that we are trying to minimize a value (the location error) we are unable to calculate, since we do not know the real locations (otherwise we would know the solution to the problem). Moreover, we cannot estimate this value, or even ensure that we are improving it, since the estimated distances are error prone, thus approximating the estimated distances with the calculated distances does not necessarily mean that we are approaching the real distances.

There are several possible ways to deal with this issue. We can classify these methods into two categories regarding the philosophy behind the technique: methods that minimize an estimated error committed by the candidate solution, and methods that maximize the likelihood (probability) that the candidate solution is correct.

Belonging in the first category are the methods that assume measured distances are real distances and solve the optimiza-

tion problem by minimizing the calculated distances error norm $L(c_{ij} - \delta_{ij})$. Typical norms are L_1, L_2 or L_∞ [12]. This method can produce good results when the measurements are accurate (errors are small), but will lead to large positioning errors if the measurement errors are large. Although these methods do not require any specific problem knowledge besides the instance data, there are several ways in which it could be introduced in order to tune the method. Among them, we could mention a simple discrimination norm minimization, in which measured values are weighted according to their expected reliability, which in turn is obtained from some problem model.

Methods in the second category require previous knowledge on the measurement errors, for they use a bidimensional probability density function (pdf) for the real distance vs. measured distance. Using calculated distances in the place of the real distances (in order to use the pdf), we maximize the product of the probability for every c_{ij} given δ_{ij}.

3. Related work

Location discovery is a fairly popular research topic, and has been solved for many scenarios using a variety of techniques. Therefore, a wide body of knowledge exists. We point out in this section several representative examples related to location discovery, errors in location and their modeling, and other issues that are related to LD.

A theoretical study on the Crámer-Rao Lower Bound on positioning error was performed in [20]. In that work, the authors consider the range-free location system based on hops from the landmarks (anchor nodes), and the Distance Vector-position method, in which both distance and angle estimates are available by nodes. Finally, some conditions are given under which DV-position outperforms range-free location.

Some early works on the subject employ the norm functions L_1, L_2 and L_∞ as the fitness functions for the optimization procedure. In [12], all three functions are alternatively used in addition to a location error minimization function $\sum_{j=1}^{n_B} \sqrt{(x_{AIj} - x_{AFj})^2 + (y_{AIj} - y_{AFj})^2}$, where *AIj* represents the original GPS-determined position of the beacon *j*, and *AFj* represents the location determined for that beacon. The paper contemplates the existence of Gaussian error both in the GPS locations of the beacon nodes, and in the distance measurements (with standard deviation proportional to the real distance). Surprisingly enough, L_∞ outperforms the other two norm functions in the experiments performed.

A study on the location errors in several applications for WSN is done in [21]. The work focuses on exposure, best- and worst-case coverage, and shortest path routing. The norm functions are used in this work as the objective functions for location discovery. The process is incremental: at each step, all nodes that can triangulate their locations using distance measurements from beacons determine their locations, and become beacons themselves for the rest of the nodes. The sources of errors are identified and modeled, and the propagation and effect of the errors are studied.

A different focus is adopted in [18], where the main stress is put on the robustness of the localization. A robust localization is defined as one that avoids flip ambiguities. A distributed algorithm for beaconless network localization is proposed, in which nodes use noisy distance measurements only. Thus, locations are determined up to a global rotation and translation. The concept of *robust quadrilaterals* is introduced, representing quads of nodes that can be unambiguously located even in the presence of measurement noise; when a node cannot be included in such a quad, its location is not determined and is considered unknown. The algorithm is implemented on a physical WSN, and supports localization of mobile nodes.

Some security issues related to LD are also considered. Possible security threats related to LD, such as Sybil or wormhole attacks, are described in [13] and [15]. Some techniques for attack resistant location discovery are described in [14], which are based on defining data-driven bounds on the accepted mean square errors (as in the L_1 norm) for every single node positioning, and based on voting. A similar method combined with the use of difectional antennae is the method proposed in [13]. The focus adopted in [15] is different, where the authors suggest a method for detecting malicious beacon nodes based on a combination of wormhole detectors with "detective" beacon nodes.

In [4], the LD problem is formulated as a Multidimensional Scaling problem (MDS). Formally speaking the MDS problem uses a number of dissimilarities (i.e. distances) in order to determine the multidimensional values (i.e. coordinates) of a set of objects. The authors propose a distributed iterative algorithm in which every node refines its location by using location information and measured distances from its neighbors. Special stress is put on the neighbor selection as a two-stage resolution process is used to eliminate the induced bias.

In this paper we use real measurement errors for the LD instance. Our starting point is the work in [7], from which we borrow the problem concept. We also employ a non-parametric model for these errors instead of assuming Gaussian models. The discrimination of measurement distances is based upon error analysis, which gives further insight than robustness criteria. We use a distance-based link selection in a first stage resolution to obtain an initial solution, then every link is considered in the second-stage refinement process by applying the model.

4. Distance measurements

The basis for location discovery is the distance measurement performed by the nodes. We will introduce some popular techniques employed in WSN to measure node-to-node distance, and present the measurement model developed in this work, based on real distance measurement data.

4.1. Distance measuring techniques

Sensor nodes can measure the distances separating themselves in several ways. This is not a novel feature, since some cellular systems already had this kind of technology. The most widely used techniques are [22]: Time of Arrival (ToA), Angle of Arrival (AoA), Time Difference of Arrival (TDoA), and Received Signal Strength (RSS). We will now briefly describe them.

In ToA, a signal is sent from a transmitter to a receiver. When the signal arrives to the receiver, it in return sends a signal to the transmitter, who can then measure the time lapse between the first signal was sent, and the second signal was received. Typically an ultrasound is used as the traveling signal and the distance between transmitter and receiver can be estimated as (Eq. (1)):

$$D = \frac{T \cdot V}{2}, \tag{1}$$

where V is the signal velocity and T is the total estimated signal traveling time. This is the technique that was employed to generate the data used for this work (Section 4.2).

The TDoA uses two signals traveling at different speeds, such as radio frequency (RF) and ultrasound. The distance can then be calculated from the time lapse between the first signal was received

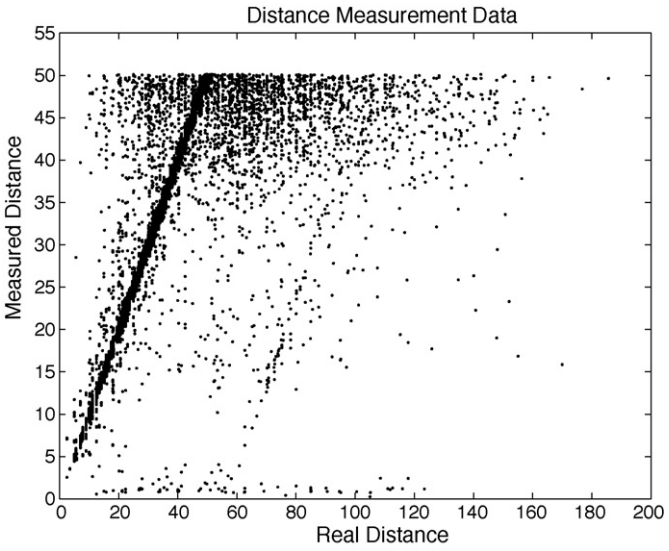

Fig. 3. Distance measurements plot.

and the second time was received as follows (Eq. (2)):

$$D = \Delta T \cdot \frac{V_{RF} \cdot V_{US}}{V_{RF} - V_{US}}, \tag{2}$$

where V_{RF} and V_{US} are the traveling speeds of RF and ultrasound signals, respectively. Note that in this case there is no need to divide it in half since signals were only sent from transmitter to receiver, and not back.

The AoA approach requires an array of receivers, which can determine the direction of the incoming signal.

Finally, RSS uses, instead of the signal traveling time, the signal propagation loss as the indicator to estimate the distance separating the two nodes. A signal traveling through space will typically reduce its energy following some law, which can be mathematically modeled. A widely spread model for the path loss is the following (Eq. (3)):

$$PL(d) = PL(d_0) + 10n \log\left(\frac{d}{d_0}\right), \tag{3}$$

where $PL(\,)$ is the path loss exponent function measured in decibels, d_0 is a reference distance, and n is an exponent that depends on the environment (generally ranging from 2 to 4).

It is widely assumed that RSS is the method that incurs the most significant errors, since path loss is subject to quick and large variations such as shadows or fading [21]. However, RSS is an interesting method since it can be easily implemented in sensor nodes without requiring any additional hardware, using the communications subsystem, which results in a virtually zero-cost method.

4.2. Measurement errors

We employ data gathered from several experiments performed at the Fort Leonard Wood Self Healing Minefield Test Facility [16]. Those experiments deployed WSN containing from 79 to 93 sensor nodes, which are custom design on an SH4 microprocessor running at 200 MHz. The nodes are equipped with four independent speakers and microphones and use ToA on the acoustic signal to determine the distance between themselves [20]. The WSN was deployed on an area of 200 m × 50 m.

In total, there are 33 sets of distance measurements collected over the course of a few days. Each set consists of a single round of acoustic signal transmission by all the nodes. Fig. 3 shows a graphical representation containing all the data from the 33 sets.

In general, this kind of knowledge can be acquired in two ways. The first is to do as explained here: perform some previous experiments from which the measurement error model can be compiled. This is not always feasible, therefore a second, on-the-fly approach can alternatively be adopted. In this approach we assume some beacon nodes are within measurement range. In that scenario, the combination of GPS-known locations and measured distances can be used to establish the measurement error model.

As can be seen in Fig. 3, the measurement errors are sometimes large. The measured distance ranges from 1 m to 50 m approximately, whereas the real distances can go as high as 200 m. This has an immediate effect: any measure link with real distance beyond 50 m will produce a very large error. In fact, we can distinguish three sections in the graph.

- *Measured* < 5 m: no visible correlation between the measured and real distances.
- 5 m < *Measured* < 40 m: real and measured distances are mostly correlated.
- 40 m < *Measured*: degraded correlation.

As a side note, we can argue these results come as no surprise. On the one hand, there is a wide consensus that measurement errors increase with the value of the distance measured, which explains measurements over 40 m are not reliable. On the other hand, the nodes used to establish the model are separated by at least 5 m (this is purely circumstantial in the benchmark), therefore any measured distance between 0 and 5 m is forcefully wrong.

5. Fitness function design

In this section we present and discuss two of the most popular fitness functions used for LD: the error norm function (that estimates the error committed by the current solution), and the likelihood function (that determines the probability that the current solution is correct). After that, we compare both fitness functions on a consistency basis for different considered scenarios, and finally propose a new method based on the comparisons results.

5.1. Error norm function

This section presents the base error norm function (which does not include problem knowledge). Then a simple link weighting approach that can be used to incorporate problem knowledge into the function is described.

5.1.1. Base function

The simplest fitness functions for location discovery are error norm functions, which assume we have no knowledge about the error model. The norm functions are a family of $\mathbb{R}^n \to \mathbb{R}$ functions that serve as indicators of *how much error* a given candidate solution incurs. When an error norm function is employed the location discovery becomes a *minimization* problem.

Let l_i, $1 \le i \le L$ be the measured link distances ($l_i = \delta_{i1,i2}$ corresponds to the link between nodes $i1$ and $i2$). We define the measured distance error for link l_i as $\epsilon_i = c_{i1,i2} - \delta_{i1,i2}$.

The most popular norm functions are L_1, L_2, and L_∞, [12,21], which are shown in Eqs. (4)–(6) respectively.

$$L_1 = |\epsilon_1| + |\epsilon_2| + |\epsilon_3| + \cdots + |\epsilon_L| \tag{4}$$

$$L_2 = \sqrt{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \cdots + \epsilon_L^2} \tag{5}$$

$$L_\infty = max\{\epsilon_1, \epsilon_2, \epsilon_3, \ldots, \epsilon_L\} \tag{6}$$

The LD problem can be solved optimally using any of these norm functions as fitness function in the absence of measurement errors. However, in the presence of significant measurement errors, the performance of any search algorithm that uses some error norm function is severely degraded. When the errors are highly varying (as is usual in WSN) the L_1 norm produces the best results, while L_∞ produces the worst ones. In [7], all three norm functions were tested on a small instance where 1 node was located using 9 anchor nodes as references, with node to node distances ranging from 7 m to 45 m. The location errors obtained were 1.272 m, 5.737 m and 8.365 m for L_1, L_2 and L_∞ respectively.

5.1.2. Incorporating problem knowledge

Other techniques use information retrieved from a previous analysis of the measurement errors. These techniques incorporate that information in order to obtain more robust and accurate fitness functions. There are a number of such techniques with varying complexities.

The simplest approach to the use of a measurement error model is to incorporate a weighting function to the norm operator. This means we will still use the norm function, but we will multiply every link distance error ϵ_i by a weight w_i that indicates how reliable is the measured value. The weighting function is developed based on some previous knowledge on the measurement errors and can be arbitrarily complex. A distance measurement that is believed to be correct will have a large weight, whereas one that is suspected to be incorrect will have a low weight. Thus, the weighting function should help reduce the impact of measurement errors.

We employ in this work a weighting function, that discriminates measurements according to their reliability. The associated norm function is the L_1 norm, since it has shown the smallest location errors in the previous tests. The weighting function is compiled from the measurement data available (see Section 4.2), but never using information related to the currently solved problem instance. The weighting function employed is calculated as follows:

$$Weight(d) = \exp\left(-\frac{average\ \ location\ \ error\ \ (d)}{NORM}\right) \quad (7)$$

where d is the measured link distance in m, the average location error is calculated for all links in the data base ranging from $(d - margin)$ to $(d + margin)$ with a *margin* of 50 cm, and *NORM* is a normalization value. In this work the value of *NORM* was empirically chosen to be 5. Fig. 4 illustrates the weighting function for the bank of measurements available. We can see how the function assigns higher weights to links with measured distances between 5 and 35 m, there is a transition zone for distances from 35 to 45 m, and distances below 5 or over 45 m are heavily discriminated against.

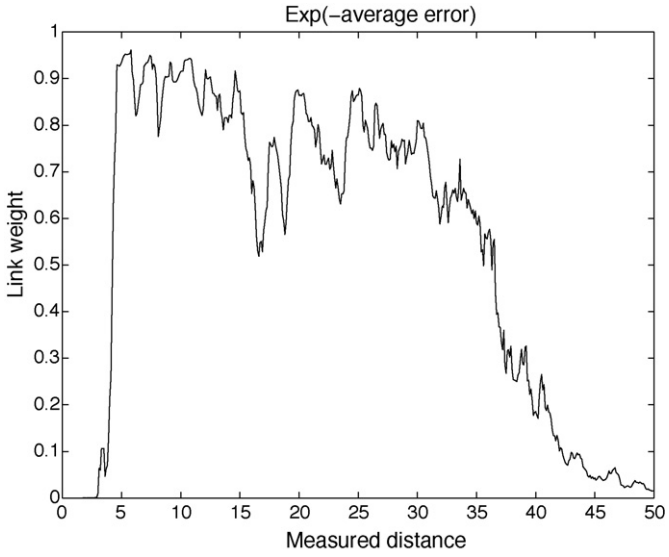

Fig. 4. Weighting function.

5.2. Likelihood function

The second solving approach considered, which natively incorporates the use of measurement error knowledge, is to solve the LD as a maximum likelihood optimization problem (ML). In this case, a full measurement error model has to be developed such that for any pair of real and estimated (measured) distances (d, δ), the model will provide the likelihood (probability) that for estimated distance δ, the corresponding real distance is d, noted $P(d, \delta)$. We say that every link has an associated probability, or likelihood. In this case the LD becomes a *maximization* problem, where the value to be maximized is the global likelihood L (Eq. (8)).

$$L = P(d_1, \delta_1) \times P(d_2, \delta_2) \times P(d_3, \delta_3) \times \cdots \times P(d_L, \delta_L) \quad (8)$$

There are many possible manners to create a statistical model for the measurement error. One can assume the measurement error follows some probability distribution (Gaussian, beta, gamma, etc.) and adjust the corresponding parameters to best fit the available data. These are called parametric models.

Rather than the previous, we use a non-parametric kernel error model similar to the one in [7]. We select 10 data sets that will generate the problem instances, and the remaining 23 will serve as the data to establish the model. For each link we know the real distance and the measured distance, we employ a pyramidal smoothing kernel function with a base diagonal of 1 m. Fig. 5 shows the probability density functions obtained for five different values of measured link distance: 10, 20, 30, 40 and 50 m. For simplicity we show all five pdf functions superimposed.

However, an analysis of this likelihood function showed that with the data available for this work (Section 4.2), for almost any given candidate solution there is always some link producing a probability of zero (even for optimal solutions). This renders the likelihood function virtually useless, since a single zero turns the global product into zero. In order to avoid this, we established a minimum probability floor, that ensures that unprobable links will not produce a zero likelihood, but rather a very low value. After

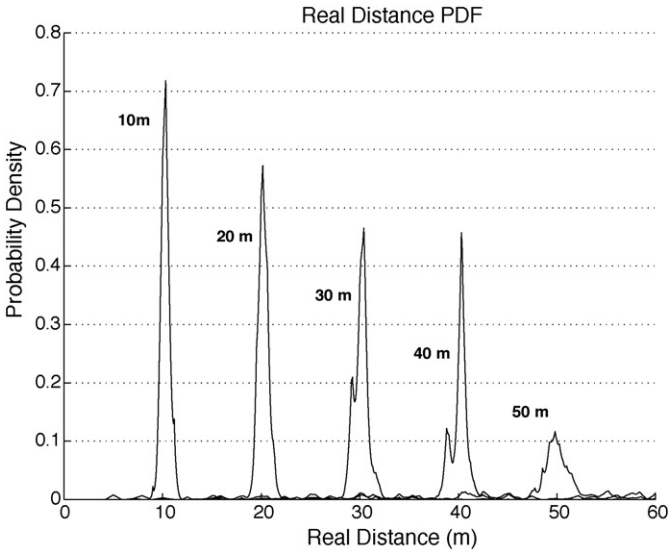

Fig. 5. Kernel error model.

some experimentation, this ground value vas set to 10^{-6}. Additionally, in order to cope with the enormous range of values of this fitness evaluator, we use a logarithmic scale rather than the linear one.

5.3. Fitness function consistency

It has already been said that the main objective of location discovery is to reduce the location error, that is, the distance between the real positions and the position estimations. However, we do not employ this parameter as the guiding criterion to our optimization technique, since it is unrealistic to assume we already know the real sensor nodes locations. Instead, a fitness function like the ones described above is employed. It is only natural thus to ask oneself whether the fitness function selected is correctly guiding the algorithm towards better solutions, that is, whether the fitness value and the location error of a given solution are correlated.

In order to provide some insight onto this issue, we will use a simple yet effective criterion: let s_a and s_b be two possible solutions for a given LD instance, and $LE()$ the location error function; if $LE(s_a) < LE(s_b)$ then the fitness function should favor s_a over s_b. If this is the case, then we say the fitness evaluator is *consistent* for this pair of nodes.

We define three different scenarios in order to apply our criterion. The first scenario consists of a pool of randomly generated pairs of solutions (low quality solutions). The second scenario consists of pairs of random solutions with low average location error (high quality solutions); for this scenario the solutions are generated by adding low power white Gaussian noise to the real locations of the nodes. The third scenario consists of pairs of consecutive best-found-so-far solutions during an execution with Simulated Annealing using a sampling rate of 1.000 iterations.

Algorithm 1 shows the pseudocode of the consistency check performed; the *initialize* function (lines 3 and 4) depends on the considered scenario, the *evaluate* function (lines 5 and 6) is the corresponding fitness function (error norm or likelihood), and the fitness values are considered *better* (line 8) when they are *higher* if the fitness function is the likelihood, or *lower* if the fitness function is an error norm. Finally, the returned value is the total number of consistent solution pairs (line 13) divided by 100 in order to get the *percentage* of consistency.

Algorithm 1. Fitness consistency check

```
1:  consistency = 0
2:  for 1.000 do
3:     initialize(S_a)
4:     initialize(S_b)
5:     evaluate(S_a)
6:     evaluate(S_b)
7:     if locationError(S_a) ≤ locationError(S_b) then
8:        if S_a has better fitness than S_b then
9:           consistency++
10:       end if
11:    end if
12: end for
13: return consistency/100
```

For each scenario we generate 1.000 random pairs of solutions, and compare the maximum likelihood function, the L_1 norm function and the location error for the two solutions. For the second scenario we have used three different power levels for the noise: 0 db, −10 dB and −20 dB corresponding the average location errors of 1 m, 10 cm and 1 cm respectively. The percentage of pairs of solutions where each function is consistent (the solution with the lowest location error gets the best fitness value) are shown in Table 1; we also include the consistency of the L_∞ norm – which is not used in this work – as a reference.

Table 1
Consistency of the maximum likelihood, L_1 and L_∞ norm functions for different location errors (%).

Scenario	*ML*	L_1	L_∞
Pure random	55.8	62.1	54.7
WGN 100 cm	69.1	66.8	49.4
WGN 10 cm	68.5	55.8	50.4
WGN 1 cm	71.7	51.9	50.1

For the second scenario we produce 1.000 pairs of solutions by adding white Gaussian noise, with 0 dB power relative to the location in meters (1 m on average), to the coordinates of the optimal solution. In this scenario the consistency is of 68.9% and 66.3% for ML and norm fitness functions respectively. If the noise power is reduced to −10 dB (10 cm on average) the consistency values become 68.0% and 56.1%, and for −20 dB (1 cm) they become 71.6% and 51.6% respectively.

From Table 1 it can be appreciated that ML acts only slightly better than blind search (50% consistency) when the solution is far from the optimum. The same holds true for the L_1 norm function when the solution approaches the optimum. We can state that the norm function is preferable when the solution is far away from the optimum, as in the beginning of the search process, and the ML function is preferable when the solution is close to the optimum, as in the end of the search process. Therefore, we propose a new approach for solving this problem, that is described in the following section.

5.4. Two-stage resolution

If we use the L_1 norm function alone, the obtained accuracy is expected to be limited, but it provides good guidance when the current solution is far away from the optimum, and the local optima are not extremely sharp. If we use the likelihood function, it provides a highly improved accuracy in the neighborhood of the optimum, but its guidance is poor in regions far away from it, and the local optima can be very strong. Therefore, using the L_1 norm seems a good idea when starting from a randomly generated solution, since it is likely to guide the search towards the neighborhood of the optimum; once the search process has entered that neighborhood, it is convenient to switch to a likelihood estimation, since it will produce much more accurate results in that narrow region.

Therefore, as in [4], we propose a two-stage solving process that combines two search processes. Fig. 6 shows the basic configuration. The basic intuition is to use an initial phase to generate a rough initial guess by using L_1 starting from a random initial solution, then a second phase to refine the initial guess by using ML. The key feature for the first phase is robustness, we want to obtain a solution that has an upper bounded location error. The key feature for the second phase is accuracy, we want to minimize the location error as much as possible.

5.5. Beacon reinforcement factor

Finally, we noticed that one of the main problems that arise during LD is the link-clustering of the WSN. This happens when a set of nearby nodes has many measurements for pairs of nodes contained in the set, and very few between any node in the set and a node outside of the set. In this situation, a translation, rotation or flipping of the complete set may result in a very slight difference in the fitness function, and is thus very difficult to detect by the search algorithm, but the location error will suffer a large increase [18]. An example of rotation error is shown in Fig. 7, where real locations

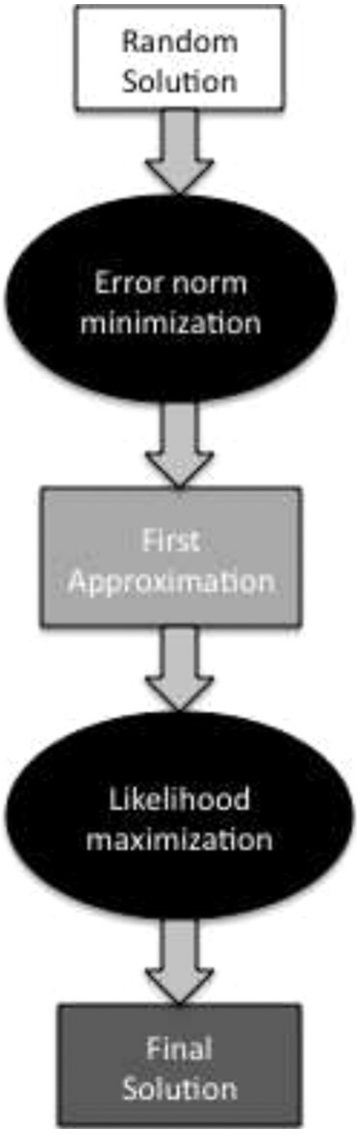

Fig. 6. Two-stage resolution process combining the first stage using error norm function and the second stage with a likelihood function.

are indicated with dots, estimated locations with asterisks and a dotted line links every estimated location with its corresponding real location. We can see that almost every dotted line intersects at a single point: the rotation center.

When many nodes belonging to one such cluster are displaced, they produce an *attraction* effect on the remaining nodes of the cluster, inducing them away from their real locations towards the cluster rotated/translated position. Speaking in optimization terms, a cluster displacement constitutes a local optima, and can in some cases be a very strong one. Therefore, this calls for some mechanism that helps escape this kind of trap. There are some heuristic factors that can be incorporated to the fitness function and can help improve its performance.

Since a beacon cannot be moved away from its location, this issue is usually solved when one or more nodes in the cluster set is a beacon. However, due to the reduced number of beacons, this is generally not the case.

We propose to reinforce the effect of the already existing beacons in the network as a way to avoid translation, rotation of flipping errors. To do this, we assign a higher weight to those links in which one the end nodes is a beacon node. This will force those

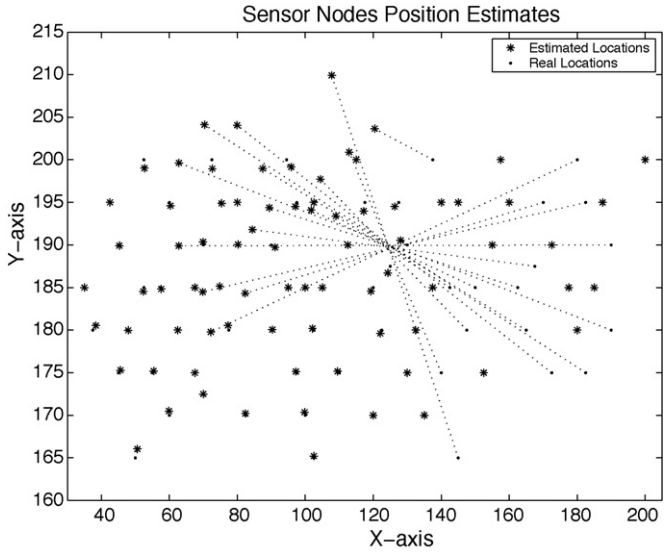

Fig. 7. Example clustering error: a cluster of nodes has their location estimations reflected through a point; this error is hard to detect when there are few distance measurements from nodes in the cluster to nodes outside the cluster.

nodes that have link distances measured with respect to some beacon to stay at this distance, specially the beacon's close neighbors. For this work we have chosen a weight of 2 for the links containing a beacon; this weight is used both in the L_1 norm and the ML functions.

6. Optimization algorithms

In this section we present the search algorithms we use to solve location discovery. These algorithms handle one solution or a population of solutions, and employ a fitness function (described in the previous section) to guide the search.

For these algorithms to be able to tackle an optimization problem, a suitable solution coding must be chosen. First, every node in a WSN is numbered. Then, we use a straightforward encoding: an array of real numbers with length double the number of sensor nodes in the WSN. The first two elements are respectively the *x* and *y* coordinates of the first node, from this point the next two values represent *x* and *y* coordinates of the following nodes respectively.

6.1. Simulated annealing

Simulated annealing is a trajectory based optimization technique [3]. It was first proposed by Kirkpatrick et al. [10]. The pseudocode for this algorithm is shown in Algorithm 2.

Algorithm 2. Pseudocode of SA

```
1: t ← 0;
2: initialize(T,S_a)
3: evaluate(S_a)
4: while not endCondition(t,S_a) do
5:    while not coolingCondition(t) do
6:       S_n ← chooseNeighbor(S_a)
7:       evaluate(S_n)
8:       if accept(S_a,S_n,T) then
9:          S_a ← S_n
10:      end if
11:      t ← t + 1
12:   end while
13:   cooldown(T)
14: end while
```

Table 2
Test instances features.

Instance	3-19A	3-19B	3-19C	3-19D	3-19E	3-19F	3-20A	3-20B	3-25A	3-25B
Number of nodes	79	93	93	94	94	94	94	93	93	94
Number of links	677	673	394	644	378	622	978	1026	992	1279
Avg. link error (m)	4.55	3.89	2.05	2.99	1.70	2.52	3.51	2.92	2.55	4.58

The algorithm works iteratively keeping a single tentative solution S_a at any time. In every iteration, a new solution S_n is generated from the previous one, S_a (line 6), and either replaces it or not depending on an acceptance criterion (lines 8 and 9). The acceptance criterion works as follows: both the old (S_a) and the new (S_n) solutions have an associated quality value, determined by an objective function (also called *fitness* function). If the new solution is better than the old one, then it will replace it. If it is worse, it replaces it with probability P. This probability depends on the difference between their quality values and a control parameter T named *temperature*. This acceptance criterion provides a way of escaping from local optima. The mathematical expression for the probability P is:

$$P = \frac{2}{1 + e^{(fitness(S_a) - fitness(S_n))/T}} \tag{9}$$

As iterations go on, the value of the temperature parameter is reduced following a cooling schedule (line 10), thus biasing SA towards accepting only better solutions. In this work we employ the geometric rule $T(n+1) = \alpha \cdot T(n)$, where $0 < \alpha < 1$, and the cooling if performed every k iterations (k is the *Markov chain length*).

For the neighbor selection we use a mutation process. In this mutation, we select each of the coordinates with a given probability p. Then, each selected coordinate is modified by adding a displacement d, which is a random value between $-R_{\max}$ and $+R_{\max}$. The value for $R_{\max}$ is selected as the average error per link measurement (considering L_1 norm) multiplied by a scaling factor we refer to as *mutation intensity*, regardless of the chosen fitness function. Furthermore, this value is weighted by a value representing the algorithm's execution progress: $1 - evaluations/max\ evaluations$. The idea behind this is to have an increasing fine grain precision even when the error norm is lower bounded.

The initial value for the temperature T is automatically generated in such a way that any movement from the initial (random) solution will be accepted with probability 80%.

6.2. Genetic algorithm

Genetic Algorithms belong to the wide family of evolutionary algorithms [2]. They appear for the first time as a widely recognized optimization method as a result of the work of John Holland in the early 1970s, and particularly his 1975 book. The pseudocode for this algorithm is shown in Algorithm 3. A standard genetic algorithm is a population based technique [3] that uses a selection operator to pick solutions from the population (line 4), a crossover and a mutation operators to produce new solutions from them (lines 5 and 6), and a replacement operator to choose the individuals for the next population (line 8).

Algorithm 3. Pseudocode of GA

```
1:  t ← 0
2:  initialize(P_a)
3:  while not endingCondition(t,P_a) do
4:      Parents ← selectionParents(P_a)
5:      Offspring ← crossover(Parents)
6:      mutate(Offspring)
7:      evaluate(Offspring)
8:      P_n ← replacement(Offspring,P_a)
9:      t ← t + 1
10:     P_a ← P_n
11: end while
```

Our implementation of the genetic algorithm uses a ranking method for parent selection and elitist replacement for the next population, that is, the best individual of the current population is included in the next one. The crossover method is the *sbx* crossover defined for continuous variables (as is the case here) [6]. The mutation method is the same employed for SA and described in Section 6.1.

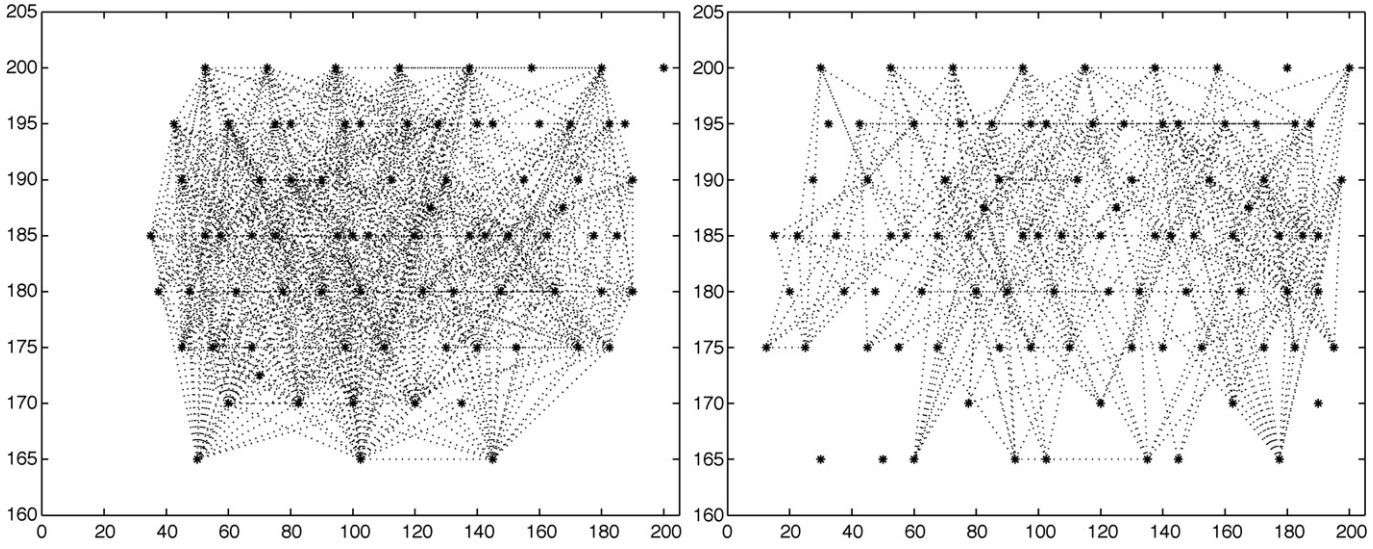

Fig. 8. Two test instances: 3-19A (left) with 79 nodes and 677 link distances, and 3-19C (right) with 93 nodes and 394 link distances.

6.3. Particle Swarm Optimization (PSO)

Particle Swarm Optimization [9] is a population based metaheuristic inspired in the social behavior of birds within a flock, and initially designed for continuous optimization problems. In PSO, each potential solution to the problem is called *particle* and the population of particles is called *swarm*. In this algorithm, each particle position x^i is updated each generation g by means of the Eq. (10).

$$x^i_{g+1} \leftarrow x^i_g + v^i_{g+1} \tag{10}$$

where factor v^i_{g+1} is the velocity of the particle and is given by

$$v^i_{g+1} \leftarrow w \cdot v^i_g + \varphi_1 \cdot (p^i_g - x^i_g) + \varphi_2 \cdot (b_g - x^i_g) \tag{11}$$

In this formula, p^i_g is the best solution that the particle i has stored so far, b_g is the best particle (also known as the *leader*) that the entire swarm has ever created, and w is the inertia weight of the particle (it controls the trade-off between global and local experience). Finally, φ_1 and φ_2 are specific parameters which control the relative effect of the personal and global best particles ($\varphi_1 = \varphi_2 = 2 \cdot UN(0, 1)$).

Table 3
Parameter values for the basic SA configuration.

Parameter	Value
Evaluations	5.000,000
Mutation rate (%)	1.5
Mutation Intensity	15
Initial T	*auto*
Markov chain length	50
α	0.9995

Algorithm 4 describes the pseudo-code of PSO. The algorithm starts by initializing the swarm (line 1), which includes both the positions and velocities of the particles. The corresponding p^i of each particle is randomly initialized, as well as the leader g (line 2). Then, during a maximum number of iterations, each particle *flies* through the search space updating its velocity and position (lines 5 and 6), it is then evaluated (line 7), and its p^i is also calculated (line 8). At the end of each iteration, the leader b is updated. Besides, as the execution progresses, the inertia weight linearly evolves from an initial value to a final value (which is generally lower).

Algorithm 4. Pseudocode of PSO

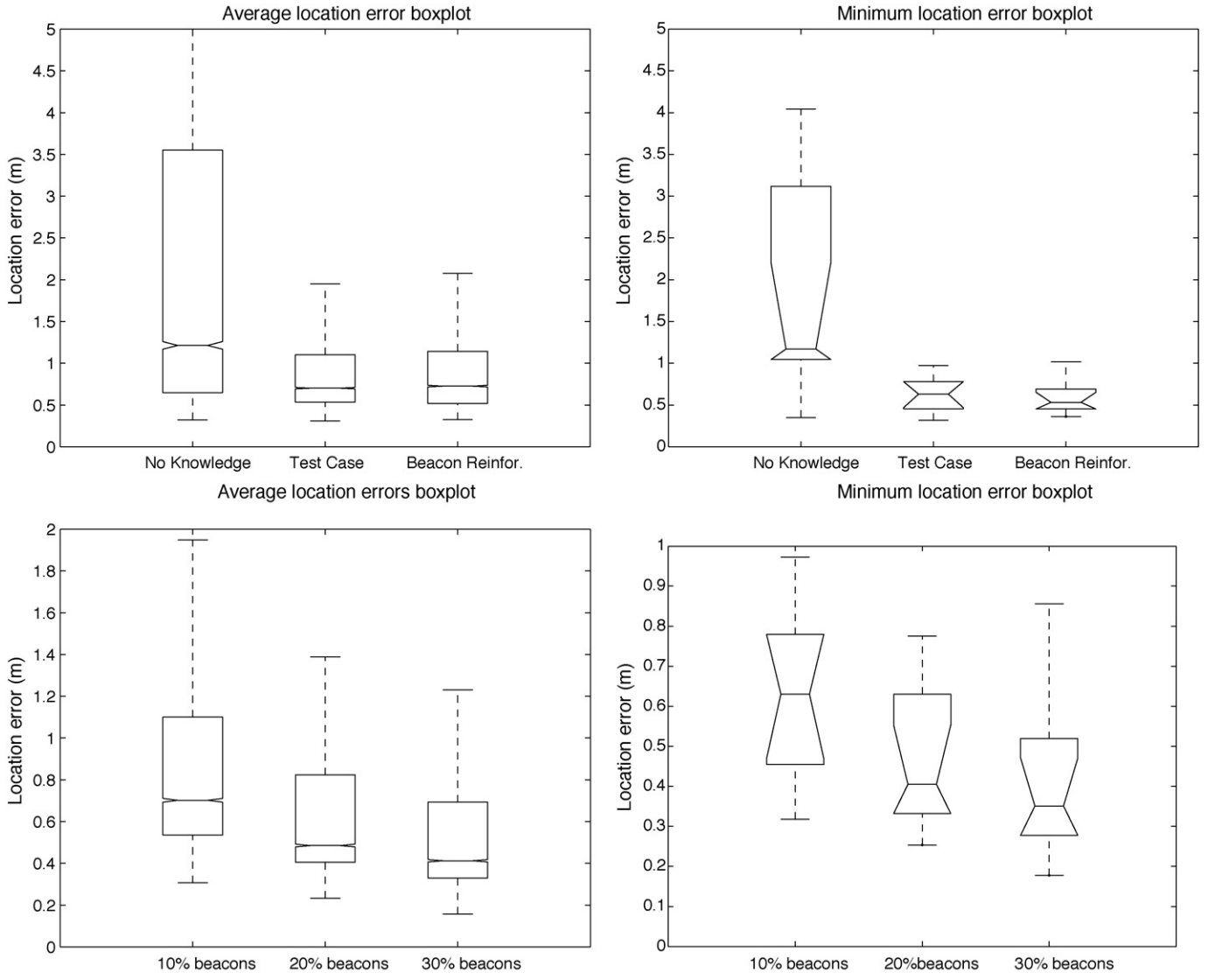

Fig. 9. Boxplot comparisons of the average (left) and minimum (right) location errors of the different experiments: Test case, without problem knowledge and adding beacon reinforcement (top); different beacon densities (bottom).

```
initializeSwarm()
locateLeader(b)
while g < maxGenerations do
  for each particle x_g^i do
    updateVelocity(v_g^i) //Equation 11
    updatePosition(x_g^i)// Equation 10
    evaluate(x_g^i)
    update(p_g^i)
  end for
  updateLeader(b_g)
end while
```

7. Experiments and results

For our experiments we select 10 measurement sets as test problem instances (see Section 4.2 and Table 2), and compile an error model from the 23 remaining sets, so that no information from an instance is used in the process of resolving it. Fig. 8 shows two of the solved instances, named "3-19A" and "3-19C". The former consists of 79 sensor nodes, with a total of 677 node-to-node distance measurements available. The latter consists of 93 nodes, with 394 node-to-node measurements.

In the following, we will take Simulated Annealing using the plain L_1 norm function with link weighting (see Section 5.1.2) – but no beacon reinforcement factor – as the base configuration for solving this problem, then establish a comparison-based analysis of the other techniques described against it.

We describe our experimental methodology in the next section. We then present the results obtained with our base configuration in Section 7.2. We then check the effect of the using problem knowledge (through link weights), and adding the beacon reinforcement heuristic factor (see Section 5.5) in Section 7.3, the analyze the impact of the number of beacon nodes in the base case from 10% to 20% and 30% in Section 7.4. We then compare the results obtained with the Genetic Algorithm and the Particle Swarm Optimization against our basic configuration in Section 7.5. We finally test the effectiveness of the likelihood fitness function and our proposed two-stage solving process in Section 7.6.

7.1. Experimental methodology

We want to test our resolution process robustness by starting from random generated initial solutions. Note that it is possible to use available knowledge (beacon positions plus measured distances) to compile an initial solution with bounded error, in that case the results obtained by the optimization algorithms would be arguably better. In our experiments we use a predefined bounding box that defines the feasible locations for the sensors; such a bounding box can be defined on the fly using the combined information from distance measurements, network topology and beacon locations.

Every problem instance will be solved using information from a single round of distance measurements. Once again, the results are expected to improve if several measurement rounds are performed. We consider an average location error below 1.5 m to be *acceptable*, and above it to be *poor*.

Since the beacons nodes have a big effect on the obtained results, we generate ten random beacon configurations for each instance, therefore producing a total of 100 scenarios. For each scenario 100 independent runs are performed in order to get statistical confidence on the results. This means that for every experiment, a total of 10,000 independent runs are performed (100 scenarios × 100 independent runs). A single run is stopped after visiting 5,000,000 candidate solutions; this takes between four and ten minutes of computation in a Pentium IV@3 GHz with 512 M RAM running Linux Fedora 7. This makes a total of 50 days computation time per experiment on average. In order to cope with the enormous amount of computation required, we employ the high-throughput platform Condor with a grid of up to 400 non-dedicated computers, which allowed us to perform up to that many independent runs in parallel. Thanks to this we managed to reduce the total time required for a single experiment to 1 day.

Table 4
Minimum and average location errors(m) obtained using the test case: Simulated Annealing with L_1 error norm function and weighting function. The highest and lowest values are highlighted.

Test case: SA with problem knowledge										
BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.6626	2.8657	0.5983	0.8484	0.7791	2.3655	0.4908	0.6118	0.9728	5.8903
Config 2	0.7321	3.1985	0.5556	0.6556	0.7493	1.8338	0.4757	0.5839	0.9021	1.0842
Config 3	0.4857	0.7199	0.4645	0.5882	0.7793	1.2224	0.5316	0.7317	0.9392	1.3399
Config 4	0.4907	0.5772	0.6285	0.8607	0.7624	1.0814	0.5564	0.6554	0.9432	1.4985
Config 5	0.6178	1.2821	0.6302	0.7242	0.8084	1.1277	0.5352	0.6603	1.2057	4.3144
Config 6	0.5701	1.2616	0.6368	0.8047	0.8512	5.9806	0.5217	0.7617	0.9227	3.9059
Config 7	0.5419	0.5867	0.5474	1.0535	0.7998	1.2721	0.4941	1.2015	1.2296	5.5411
Config 8	0.5895	0.6563	0.6462	0.9372	0.6033	4.1068	0.5185	1.1612	0.9477	2.9734
Config 9	0.5529	0.7319	0.4454	0.5654	0.8123	1.0724	0.5234	0.5987	0.9355	2.0305
Config 10	0.5315	0.6485	0.6191	0.7771	0.8255	1.7596	0.5774	0.7396	1.1309	3.0067
BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.7473	1.0891	0.4066	0.4575	0.4545	0.5133	0.3189	0.4455	6.3661	7.2795
Config 2	0.8502	0.9869	0.3765	0.6211	0.4193	0.4721	0.3381	0.4223	0.4239	0.4712
Config 3	0.4256	0.6365	0.4394	0.7867	0.4565	0.5533	0.3388	0.4115	0.4152	0.5293
Config 4	0.8287	1.0234	0.3999	0.6392	0.4749	0.5934	0.3225	**0.3999**	0.4288	0.6139
Config 5	0.7701	1.0419	0.4876	3.5575	0.5444	8.9061	0.3479	0.6707	**17.4838**	**28.8802**
Config 6	0.7752	1.1592	0.3634	1.8911	0.4293	0.4715	0.3557	0.4455	0.4546	5.0313
Config 7	0.7567	3.7864	0.4006	1.0498	0.4996	0.6079	**0.3066**	0.6341	0.4235	0.8147
Config 8	0.7571	1.0251	0.4351	1.2708	0.4972	0.5584	0.3159	1.8009	16.0939	17.0134
Config 9	0.7252	1.1207	0.4362	2.2256	0.5039	0.5826	0.3708	0.4732	0.4309	0.5629
Config 10	0.8119	1.0811	0.3637	0.5871	0.3987	0.4523	0.3379	0.4148	5.7559	7.9845

Table 5
Results using Simulated Annealing: without problem knowledge (top), and adding beacon reinforcement to the test case (bottom).

SA without problem knowledge

BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	11.2303	18.8675	1.0439	1.3103	1.1769	8.0828	3.1171	3.8911	1.1747	1.5641
Config 2	**14.4672**	21.4007	0.8992	0.9706	0.4372	2.3338	0.9767	1.8711	1.1062	1.4061
Config 3	0.5654	0.6482	0.8528	1.8961	1.1001	1.8956	4.2689	4.5113	1.2194	1.4402
Config 4	0.5456	0.5816	0.7137	0.9063	1.0843	1.5855	3.1752	4.0621	1.1769	1.4494
Config 5	0.6377	1.2153	0.7269	1.0581	1.0886	1.7635	1.1525	2.7075	1.2696	3.7217
Config 6	0.5808	4.9765	1.0173	4.5999	1.2834	8.7244	1.0436	4.0927	1.2251	4.4017
Config 7	0.5839	1.0067	0.6261	1.1708	1.0952	2.4952	0.9387	1.1585	1.9156	6.4623
Config 8	0.6166	0.6953	1.0153	1.9838	1.1539	5.7915	4.3957	8.3389	1.1441	2.1449
Config 9	0.6015	0.8329	0.8064	0.8664	1.1048	1.4334	0.9413	3.9675	1.1206	1.9605
Config 10	0.6452	6.9427	0.9959	2.0196	1.1082	6.5457	1.0917	1.2682	1.4528	3.8971

BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.4271	0.8379	1.1316	1.2437	1.1662	1.2362	0.3505	0.4241	4.0435	8.4201
Config 2	0.3771	0.5158	1.1239	1.1546	1.1339	1.1812	0.3606	0.4148	0.4733	0.5197
Config 3	0.4524	0.6662	1.1846	1.2872	1.1743	1.2609	0.3215	0.5576	0.4534	0.5686
Config 4	0.4394	0.6065	1.1378	1.1955	1.1288	1.2768	0.3413	0.3901	0.4807	0.6034
Config 5	0.4188	0.6057	1.2159	8.2288	1.2317	1.6348	0.3762	0.4942	0.6281	**32.8157**
Config 6	0.4198	5.1663	1.1246	4.4196	1.1341	1.1971	0.3704	0.4188	4.2424	11.5843
Config 7	0.3502	5.4619	0.4404	0.6312	0.5521	2.1002	**0.3213**	**0.3792**	0.4235	0.8921
Config 8	0.4565	0.6751	1.1616	6.686	1.1919	1.2652	0.3428	8.6623	0.5324	15.9664
Config 9	0.4269	0.6368	1.1935	4.8177	1.1815	1.2586	0.4012	0.4679	0.4924	0.6405
Config 10	0.4977	0.7903	1.0961	1.1657	1.1197	1.2019	0.3486	0.4039	0.4956	8.251

SA with beacon reinforcement

BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.6929	4.8566	0.6404	1.1646	0.8103	1.2171	0.5021	0.5814	1.0152	3.0445
Config 2	0.7393	2.2779	0.5765	0.6883	0.8027	1.4167	0.5346	0.8279	1.5802	1.7094
Config 3	0.4171	0.4485	0.4386	0.5784	0.7728	1.3156	0.4967	0.5718	0.9334	1.7477
Config 4	0.4132	0.4508	0.6314	0.9287	0.8095	1.233	0.4881	0.5986	0.9754	1.4846
Config 5	0.5889	0.6889	0.6504	0.8931	0.9866	1.3564	0.5342	0.6438	1.0869	2.7899
Config 6	0.5291	0.8889	0.6321	0.7812	0.8799	5.0206	0.5114	2.8215	0.8119	4.3977
Config 7	0.5238	0.5686	0.5996	0.7058	0.7894	1.0814	0.4757	0.6651	0.9434	5.9786
Config 8	0.6027	0.6706	0.6472	2.4393	0.6842	5.3925	0.5032	0.6003	0.8452	2.6328
Config 9	0.5647	0.6008	0.4746	0.5874	0.8608	1.1783	0.5332	0.6123	1.0233	2.3169
Config 10	0.5246	0.6287	0.6666	0.8304	1.0237	1.7361	0.5812	0.6656	1.3152	2.0783

BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.5614	1.0787	0.4128	0.4765	0.4528	0.5134	0.3615	0.4287	0.4724	4.8811
Config 2	0.7653	1.0098	0.3869	0.4135	0.3992	0.4404	0.3306	0.7257	0.4035	0.4434
Config 3	0.6766	0.8276	0.4503	0.8636	0.4462	0.5121	0.3666	0.4359	1.0372	1.1008
Config 4	0.8313	0.9906	0.3975	0.5238	0.4371	0.5067	0.3284	0.4012	0.4247	0.4893
Config 5	0.7672	1.0033	0.5087	4.3328	0.5139	7.1002	0.3404	1.5179	0.4621	1.9872
Config 6	0.4874	0.8406	**4.7028**	5.3759	0.4014	0.4432	0.3722	0.4884	4.2752	4.4159
Config 7	0.7562	1.7187	0.3975	0.9768	0.4077	0.5466	0.3325	**0.4008**	0.5827	0.7866
Config 8	0.6085	0.9094	0.4507	1.331	0.4708	1.6041	**0.3264**	1.6112	0.5339	**16.1109**
Config 9	0.7652	1.1716	1.0636	4.1773	0.4747	0.7022	0.4122	0.4808	0.4516	0.4891
Config 10	0.7902	1.1075	0.4085	0.9908	0.3948	0.4362	0.3517	0.4197	0.4302	0.4847

All the results obtained in our experiments are subject to a statistical analysis for their comparison. The statistical analysis consists of a one-way ANOVA test is the data sets are normally distributed and homocedastic (checked with Kolmogorov–Smirnoff and Levene tests respectively), or a non-parametric Kruskal–Wallis test otherwise. The results of these tests will be displayed when necessary in a *versus* table confronting pairs of algorithms in the form 'A vs. B'. In this case a ▲ symbol indicates that the values in the data from algorithm A are significantly larger than the values in data from B, a ▽ symbol indicates that values in data from A are significantly lower than values in data from B, and a blank space indicates that no statistically significant difference could be found.

7.2. Test case results

Table 4 shows the location error results obtained with 100 independent runs of SA (Table 3 shows the parametric configuration of the algorithm) with L_1 error norm as the fitness function in 100 scenarios (10 problem instances with 10 beacon configurations each).

As can be seen from the table, the results obtained can be considered reasonably good. Despite the beacon configuration seems to have an important impact on the obtained results (the average location error varies from 0.40 m in instance 3-25A to 28.9 m in instance 3-25B), unefficient solving behaviour seems to happen sparsely: only 24 of the 100 solved scenarios produce an average location error above 1.5 m, which is reduced to just 4 if

Table 6
Minimum and average location errors(m) with: 20% of beacon nodes (top), 30% of beacon nodes (bottom).

Results using 20% beacon nodes										
BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.4396	0.6648	0.3317	0.4775	0.6303	0.8976	0.4168	0.4782	0.7756	0.9182
Config 2	0.4577	0.7128	0.3327	0.4768	0.6286	0.9664	0.4036	0.4727	0.7867	0.9841
Config 3	0.4671	0.5517	0.3289	0.4636	0.6225	0.8613	0.3948	0.4861	0.7789	0.9369
Config 4	0.4535	0.6562	0.3297	0.5063	0.6066	0.8881	0.3973	0.4711	0.7924	0.9483
Config 5	0.4569	0.6044	0.3266	0.4527	0.6475	0.8715	0.4178	0.4676	0.7822	0.9462
Config 6	0.4616	0.6568	0.3525	0.5143	0.6285	0.9345	0.3901	0.4639	**0.8004**	0.9286
Config 7	0.4677	0.6614	0.3396	0.4446	0.6376	0.9031	0.4153	0.4717	0.7761	0.9178
Config 8	0.4534	0.6032	0.3324	0.5127	0.6313	0.9487	0.4121	0.4753	0.7908	0.9434
Config 9	0.4535	0.6558	0.3184	0.4462	0.6308	**1.0351**	0.4045	0.4894	0.7922	0.9364
Config 10	0.4651	0.6025	0.3466	0.4506	0.6369	0.8789	0.4073	0.4676	0.7758	0.9724
BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.6552	0.9354	0.3587	0.6116	0.3934	0.4598	0.2536	0.2881	0.2944	0.3718
Config 2	0.6332	0.9299	0.3746	0.6671	0.4176	0.4656	0.2536	0.2925	0.2999	0.4111
Config 3	0.7054	0.9096	0.3602	0.6187	0.4103	0.4642	**0.2333**	**0.2868**	0.2982	0.4024
Config 4	0.6748	0.9148	0.3595	0.6492	0.4008	0.4604	0.2501	0.2902	0.3022	0.3686
Config 5	0.6937	0.9076	0.3595	0.6626	0.4002	0.4633	0.2404	0.2895	0.3094	0.3729
Config 6	0.7734	0.9178	0.3629	0.6307	0.3866	0.4601	0.2577	0.2911	0.3104	0.3896
Config 7	0.6336	0.9108	0.3706	0.6113	0.4065	0.4602	0.2525	0.2861	0.3035	0.3873
Config 8	0.6562	0.9181	0.3797	0.6158	0.4131	0.4625	0.2491	0.2911	0.3083	0.3954
Config 9	0.6497	0.9197	0.3583	0.6389	0.4191	0.4636	0.2488	0.2888	0.3059	0.4007
Config 10	0.6341	0.9123	0.3634	0.6578	0.3952	0.4612	0.2406	0.2866	0.2975	0.3987
Results using 30% beacon nodes										
BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.3486	0.3878	0.5375	0.6221	0.3376	0.5121	0.4147	0.4583	0.8599	0.9414
Config 2	0.3586	0.3885	0.5196	0.6188	0.3435	0.5121	0.4107	0.4571	0.8555	0.9803
Config 3	0.3566	0.3895	0.5248	0.6245	0.3511	0.4985	0.4086	0.4579	**0.8713**	0.9736
Config 4	0.3567	0.4106	0.5342	0.6187	0.3511	0.5075	0.4011	0.4585	0.8587	0.9628
Config 5	0.3517	0.4098	0.5329	0.6259	0.3556	0.4933	0.4028	0.4599	0.8435	0.9577
Config 6	0.3505	0.3868	0.5256	0.6199	0.3489	0.5178	0.4046	0.4603	0.8698	0.9562
Config 7	0.3611	0.3918	0.5267	0.6195	0.3527	0.4766	0.3979	0.4582	0.8677	0.9622
Config 8	0.3548	0.4127	0.5242	0.6165	0.3499	0.4989	0.4004	0.4584	0.8591	**0.9946**
Config 9	0.3596	0.3916	0.5375	0.6252	0.3547	0.5007	0.4101	0.4579	0.8658	0.9545
Config 10	0.3623	0.3902	0.5187	0.6004	0.3457	0.5111	0.4035	0.4599	0.8688	0.9611
BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.6594	0.9009	0.1714	0.3498	0.1781	0.2149	0.2647	0.3165	0.3344	0.3633
Config 2	0.6482	0.9116	0.1774	0.4258	0.1841	**0.2129**	0.2776	0.3113	0.3367	0.3625
Config 3	0.6361	0.8962	0.1731	0.3798	0.1868	0.2134	0.2696	0.3127	0.3146	0.3599
Config 4	0.6787	0.9025	**0.1574**	0.3994	0.1828	0.2135	0.2753	0.3168	0.3324	0.3651
Config 5	0.7918	0.9031	0.1725	0.3658	0.1845	0.2157	0.2744	0.3114	0.3296	0.3652
Config 6	0.7924	0.9143	0.1771	0.3889	0.1854	0.2143	0.2719	0.3143	0.3262	0.3634
Config 7	0.6605	0.8934	0.1725	0.3752	0.1886	0.2128	0.2673	0.3186	0.3221	0.3592
Config 8	0.6878	0.9064	0.1789	0.3405	0.1949	0.2142	0.2655	0.3218	0.3263	0.3631
Config 9	0.8054	0.9256	0.1756	0.3579	0.1841	0.2137	0.2641	0.3156	0.3261	0.3664
Config 10	0.6791	0.8974	0.1758	0.4261	0.1858	0.2132	0.2676	0.3104	0.3211	0.3609

Table 7
Parameter values for the GA.

Parameter	Value
Evaluations	5, 000, 000
Population size	100
Parent selection	*Roulette*
Offspring selection	*8-Tournament*
Crossover prob.	0.80
Mutation rate (%)	1.5
Mutation Intensity	15

Table 8
Parameter values for PSO.

Parameter	Value
Evaluations	5,000,000
Swarm size	50
$C1$	2
$C2$	2
Starting inertia	0.5
Final inertia	0.1

Table 9
Minimum and average location errors(m) obtained with: Genetic Algorithm (top), Particle Swarm Optimization (bottom).

Results with Genetic Algorithm

BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	5.9827	19.8387	0.8769	6.4971	5.2331	13.0268	1.6172	8.1077	4.5286	14.2532
Config 2	5.972	**22.0188**	0.6211	2.4887	2.4646	7.9038	0.6978	3.9925	2.1969	8.1091
Config 3	0.4946	4.9096	0.5357	7.2418	2.7589	9.8686	1.1992	4.5081	2.8818	6.1159
Config 4	0.4369	2.7703	0.6893	4.1008	2.7423	8.4936	1.4368	5.6387	3.9571	10.0566
Config 5	0.4911	7.4032	0.7954	7.4053	2.4309	9.0038	0.8132	9.0807	5.0479	16.4963
Config 6	2.0359	10.3731	0.7984	7.8259	7.3377	14.8285	1.5122	8.2448	2.1912	8.8964
Config 7	0.6109	6.0715	0.9071	6.6465	3.7897	10.4497	1.3852	11.7378	7.1284	14.9416
Config 8	0.7825	11.1164	2.2923	10.3181	7.6569	18.3324	3.5139	14.4739	4.9276	11.8834
Config 9	0.6168	10.6745	0.4584	2.9978	2.4207	7.4979	0.5503	3.2374	3.2897	7.0472
Config 10	3.3282	13.5453	0.9035	5.3971	2.4584	10.3287	0.8029	6.5319	7.0939	16.2653

BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	2.0229	8.5702	0.4552	7.2288	0.3891	5.9592	0.6682	7.937	4.9965	12.2315
Config 2	0.7013	3.5413	0.3813	3.3024	0.4067	2.8424	0.3677	4.5394	0.3923	3.7415
Config 3	0.9254	6.5843	0.7165	6.3739	0.7383	5.5826	0.6327	5.3921	1.2109	6.0116
Config 4	0.9169	4.8898	0.3979	3.8246	0.4307	3.2809	**0.3311**	2.4229	0.4019	2.9565
Config 5	1.4528	6.0034	**8.2931**	22.8591	4.9428	21.3669	1.6576	11.1839	1.0977	15.9262
Config 6	1.7646	13.0029	0.9634	6.7157	0.4237	3.9867	0.7181	11.6841	3.768	11.0229
Config 7	1.3786	10.5486	0.3654	9.1219	0.5889	9.5807	0.3713	5.9569	0.5881	5.4853
Config 8	0.9757	6.3597	1.1468	14.1739	1.9327	13.6648	1.3014	11.0747	1.3058	12.3737
Config 9	1.7313	5.5973	1.0512	10.0986	0.5798	10.3686	0.4995	5.4779	0.3961	4.7363
Config 10	2.2456	7.0669	0.4077	**1.9349**	0.3972	1.8575	0.3867	5.0026	0.4487	7.5859

Results with Particle Swarm Optimization

BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	2.7891	15.1347	0.6484	4.1813	3.9535	11.3705	1.0008	5.7746	4.218	13.5141
Config 2	2.4903	**22.4516**	0.5705	2.2911	0.9841	6.1834	0.6308	2.7004	3.5381	7.2288
Config 3	0.4485	5.0691	0.4674	5.8661	3.2797	16.0281	0.5282	3.9503	2.8673	8.2748
Config 4	0.4305	2.4821	0.6723	3.9198	2.3771	9.3084	0.8096	5.0753	3.4372	9.6144
Config 5	0.5601	5.8519	0.6539	5.0535	2.5038	9.3653	0.9269	7.6245	4.6487	15.7682
Config 6	0.8961	8.6816	0.7495	5.5361	3.4227	14.5862	0.8797	6.1662	2.3396	7.4059
Config 7	0.5592	4.3164	0.6644	5.1652	3.9273	10.8965	1.193	10.1843	4.4465	15.6244
Config 8	0.5214	10.7128	1.3383	7.4438	5.4583	16.107	1.7146	10.1535	3.0407	9.2273
Config 9	0.5595	10.132	0.4512	2.4253	1.6372	7.8785	0.5584	2.5775	1.5261	6.5987
Config 10	1.1351	9.3587	0.6062	4.1654	2.8958	8.6782	0.6593	4.4995	4.5452	15.1681

BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.9704	5.5585	0.4101	5.1747	0.4745	4.7208	0.3772	6.0773	0.9821	9.0104
Config 2	0.7075	2.8349	0.3835	1.7378	0.4119	**1.1634**	**0.3218**	2.6789	0.3915	1.7002
Config 3	1.0503	6.1974	0.4335	4.8842	0.4278	4.0497	0.3781	3.7915	0.6913	3.7381
Config 4	0.5942	3.5296	0.3883	4.1689	0.4072	3.3667	0.3302	1.7536	0.4224	2.5592
Config 5	1.1611	5.7341	**5.7021**	18.1622	2.0864	20.7531	0.4017	10.2647	1.0281	15.0996
Config 6	1.1277	10.6272	0.7579	6.1896	0.4095	2.4955	0.4179	7.3162	1.9704	8.2133
Config 7	2.5071	9.5511	0.7446	11.4921	0.4174	11.0013	0.3353	2.5632	0.4659	2.7317
Config 8	0.9085	4.7098	1.2242	12.0076	1.0678	12.4008	0.7181	9.1922	1.3287	11.5202
Config 9	1.4714	4.5927	0.5858	8.9011	0.5158	9.9894	0.3767	3.4694	0.4052	2.9283
Config 10	1.2195	4.8817	0.4622	2.1388	0.3994	1.8116	0.3279	3.0534	0.4497	4.4949

we pick the minimum location error instead (96% effectiveness). This seems quite reasonable considering that the average distance measurement error ranges from 1.70 to 4.58 m, which means the average location error is significantly lower. Given that this test case can be seen as a technique for obtaining only rough initial approximations to the real locations (according to our two-stage proposed technique in Section 5.4), these results suggest that it indeed constitutes a robust approach.

7.3. Impact of the link weighting and the beacon reinforcement

In this section we analyze the relative performance of the different flavors of L_1 error norm fitness function (Section 5.1). For this, we compare the test case against SA using the L_1 error norm fitness function without any knowledge from the problem (i.e. without link weight – see Section 5.1.2), and the SA with L_1 error norm with beacon reinforcement (see Section 5.5). All solutions are obtained from 100 independent executions performing 5,000,000 evaluations. The parametric configuration for the two new approaches is the same as the test case (Table 3).

The detailed results obtained with SA without problem knowledge are shown in the top part of Table 5, and those obtained using beacon reinforcement are shown in the bottom part of Table 5. Fig. 9 (top) shows the boxplot representation of the global average location errors (left) and the minimum location errors (right).

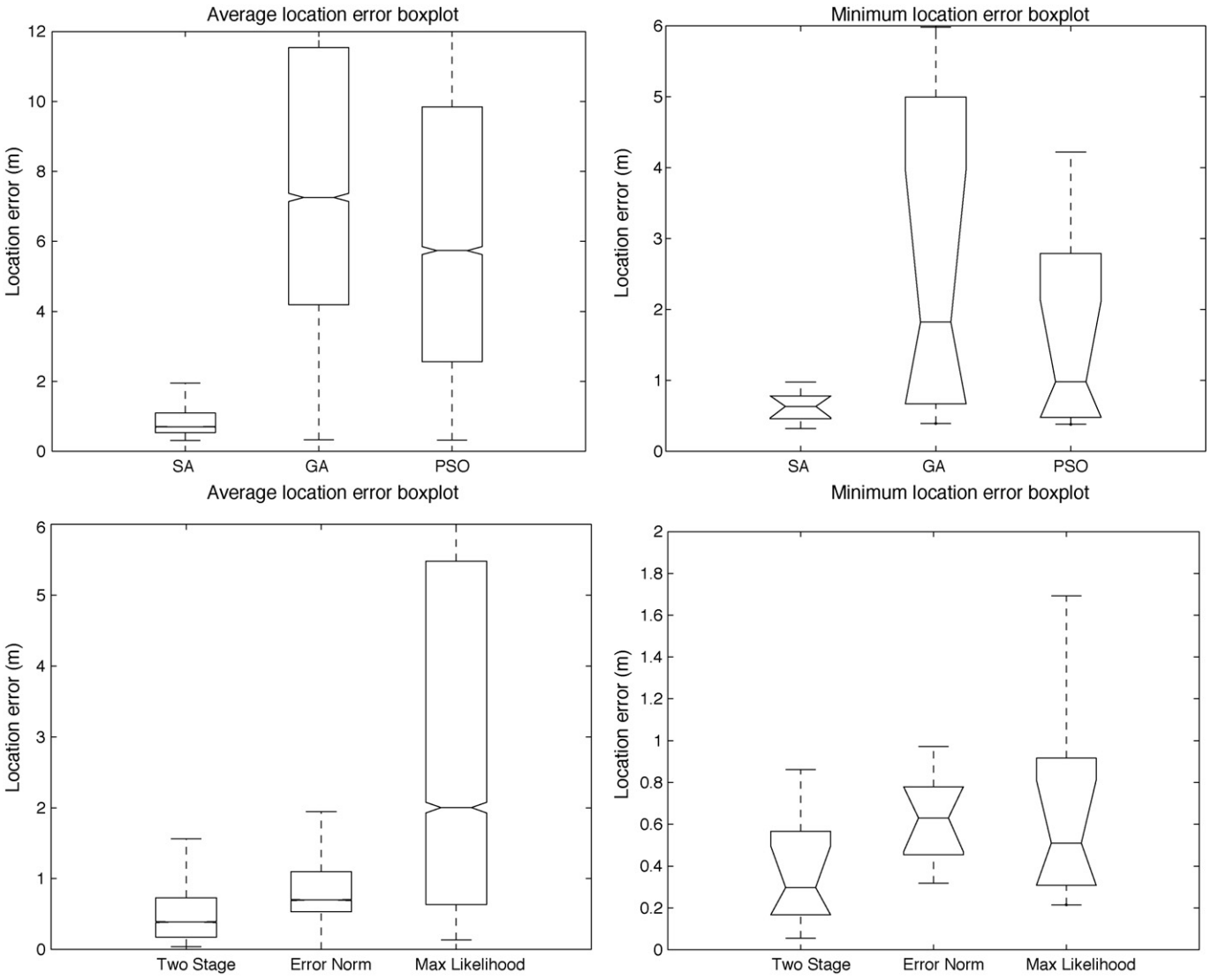

Fig. 10. Boxplot comparisons of the average (left) and minimum (right) location errors of the different experiments: SA, GA and PSO (top); two-stage, error norm and maximum likelihood (bottom).

Finally, the statistically assessed comparison between the test case and these two techniques is shown in Table 11.

SA without problem knowledge gets an average location error that ranges from 0.38 m (instance 3-25A) to 32.82 m (instance 3-25B). There are 46 scenarios with average location error over 1.5 m (24 in the test case), and 9 scenarios with minimum location error over 1.5 m (4 in the test case). SA with beacon reinforcement produces an average location error ranging from 0.40 m (instance 3-25A) to 16.11 m (instance 3-25B). There are 27 scenarios with average location error over 1.5 m, and 3 scenarios with minimum location error 1.5 m.

The results obtained with SA not incorporating problem knowledge have noticeably higher location errors than the two other cases considered (see Tables 4 and 5, and Fig. 9 up): while the former typically ranges from 0.5 to 3.5 m, with its median above 1 m, the two latter range from 0.5 to 1 m, with their medians around 0.65 m. In fact, although the results obtained using beacon reinforcement are slightly better than those without it – the test case – (at least for the minimum location boxplot, Fig. 9 upper-right), surprisingly the differences do not seem significant.

These impressions are confirmed be the statistical analysis results shown in Table 11. The location error obtained in the test case is significantly better (i.e., lower) than with SA without knowledge in 68 scenarios, and significantly worse in 14, out of 100. Looking at the problem instances, the test case gets the best results in 7, and is outperformed in only 1 (3-19F). Comparing the test case with the beacon reinforcement SA, the former is significantly better in 25 scenarios, whereas the latter is better in 36. Therefore, we can state that there is a noticeable improvement by incorporating problem knowledge (in our case under the shape of a link weight function that acts as a quality measure), and only a slight improvement by adding then a beacon reinforcement factor.

7.4. Influence of the number of beacons

In this section we study how the number of beacons affects the overall location error. Our intuition says that when the beacon density augments, the expected resulting location error should be reduced. Table 6 (top) shows the results obtained in instances containing 20% of beacon nodes, and Table 6 (bottom) shows the results in instances containing 30% of beacon nodes. Fig. 9 (bottom) shows the boxplot representation of the average location errors (left) and the minimum location errors (right).

Table 10
Results obtained with the maximum likelihood (top), and the two-stage search process (bottom).

Maximum likelihood

BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.0542	2.7735	0.4056	0.6125	0.5154	3.5184	0.2932	0.8991	0.6182	3.7347
Config 2	0.0563	3.5106	0.3926	0.9822	0.3371	2.6653	0.3747	0.9917	0.6166	2.3817
Config 3	0.0514	0.3912	0.2616	0.5953	0.4424	3.1997	0.3626	1.2055	0.7387	2.9963
Config 4	0.0461	0.2682	0.3921	0.4867	0.3557	2.8712	0.4409	1.4563	0.7893	3.5234
Config 5	0.0538	0.8051	0.3767	0.6286	0.4744	3.3616	0.4059	1.3839	**1.8321**	**12.2105**
Config 6	0.0428	0.3518	0.3871	0.7175	0.3375	7.0983	0.4815	1.3998	0.6061	4.8121
Config 7	0.0513	0.2777	0.3674	0.7274	0.3774	6.1396	0.4194	3.5493	1.3992	9.2969
Config 8	0.0531	2.9155	0.4211	0.6523	0.5667	5.7738	0.5138	1.3402	0.7076	4.0268
Config 9	**0.0433**	0.9664	0.2228	0.3339	0.2883	2.3395	0.3903	1.3425	0.4607	5.0763
Config 10	0.0497	0.3599	0.4173	0.7637	0.4227	2.4264	0.3402	1.0215	0.6258	9.3448

BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.2021	0.5131	0.1668	0.3622	0.1726	0.4214	0.1398	0.2652	0.1455	0.1791
Config 2	0.1827	0.5686	0.1716	0.3222	0.1713	0.3033	0.1143	0.1782	0.1496	0.2962
Config 3	0.1913	1.2389	0.1943	0.2753	0.1745	0.2045	0.1235	0.1733	0.1433	0.1668
Config 4	0.1931	0.5567	0.1742	0.2669	0.1747	0.4429	0.1126	0.1654	0.1506	0.1748
Config 5	0.2636	1.0706	0.1963	1.6077	0.1744	2.0171	0.0985	0.3838	0.1504	0.1773
Config 6	0.1984	0.5948	0.1672	0.6014	0.1739	0.2716	0.1326	0.3712	0.1508	0.1783
Config 7	0.2882	5.0131	0.1958	0.6225	0.1685	0.3376	0.0774	2.0649	0.0838	0.5825
Config 8	0.1963	0.5082	0.2012	0.5186	0.1791	0.7284	0.0753	0.2874	0.1019	0.2334
Config 9	0.2033	0.6652	0.1975	0.3157	0.1677	0.7796	0.1334	0.5867	0.1458	0.3745
Config 10	0.2057	0.5163	0.1754	0.5297	0.1699	0.1954	0.1304	0.1805	0.1446	**0.1628**

Two Stage Process

BEACON	3-19A		3-19B		3-19C		3-19D		3-19E	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.0552	0.3285	0.3949	0.4638	0.5668	0.9224	0.2533	0.3369	0.8612	0.9997
Config 2	0.0558	0.4582	0.3412	0.4132	0.6179	0.8584	0.4164	0.5319	1.4346	1.5608
Config 3	0.0507	0.0621	0.2011	0.2769	0.7136	0.9202	0.2407	0.3442	0.7329	0.8965
Config 4	0.0506	**0.0566**	0.3862	0.4218	0.5778	0.6484	0.2617	0.3989	0.9072	1.0715
Config 5	0.0514	0.0582	0.3544	0.3984	0.7096	0.8248	0.3624	0.4858	2.1906	2.7578
Config 6	0.0525	0.0657	0.3743	0.4395	0.5434	0.8262	0.3327	0.4201	0.7485	0.8331
Config 7	0.0499	0.0565	0.3705	0.4465	0.5094	0.6587	0.2691	0.4283	2.8296	3.0227
Config 8	0.0507	0.0642	0.4088	0.4571	0.6699	0.9425	0.2713	0.3567	0.7242	0.8849
Config 9	**0.0423**	0.0593	0.2092	0.2398	0.5635	0.6558	0.2969	0.3774	0.8641	1.0013
Config 10	0.0527	0.0658	0.3797	0.4746	0.8048	1.0159	0.2984	0.4194	1.0194	1.1805

BEACON	3-19F		3-20A		3-20B		3-25A		3-25B	
CONFIG.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.	Min.	Avg.
Config 1	0.3428	0.4471	0.1678	0.1731	0.1692	0.1848	0.1218	0.1516	4.5309	4.6424
Config 2	0.5376	0.6387	0.1726	0.1969	0.1678	0.1936	0.1015	0.1351	0.1344	0.1508
Config 3	0.4165	0.4617	0.1697	0.2056	0.1694	0.1904	0.1019	0.1231	0.7432	0.7528
Config 4	0.5652	0.6254	0.1578	0.2026	0.1697	0.1866	0.0871	0.1321	0.1423	0.1577
Config 5	0.5288	0.6378	2.8404	2.9082	0.1686	0.1945	0.0853	0.1264	0.1355	0.1516
Config 6	0.4801	0.5537	**4.5587**	**4.7168**	0.1691	0.1819	0.1267	0.1592	4.0015	4.0909
Config 7	0.6078	0.8853	0.7302	0.7649	0.1598	0.1752	0.0653	0.0853	0.2818	0.3504
Config 8	0.2883	0.3809	0.1715	0.2157	0.1767	0.2466	0.0599	0.1004	0.0933	0.1482
Config 9	0.4978	0.5622	3.2149	3.3434	0.162	0.2443	0.1021	0.1739	0.1349	0.1878
Config 10	0.4964	0.5729	0.1759	0.2012	0.1672	0.1756	0.1183	0.1461	0.1384	0.1448

In Table 12 we show the results of the statistical analysis of the comparisons between the location errors obtained with 10% of beacons against 20 and 30% of beacons. Note that the beacon configurations in these two scenarios cannot match the ones used in out test case (since different numbers of beacons are used), therefore the comparisons are not made on a scenario basis, but rather on whole problems (by aggregating all the solutions for all the beacon configurations for each problem).

The average location error obtained ranges from 0.29 to 1.04 m for instances containing 20% beacon nodes, and from 0.21 to 0.99 m for instances with 30% beacon nodes. The minimum location error ranges from 0.23 to 0.80 m and from 0.16 to 0.87 m for instances containing 20% and 30% beacon nodes respectively. Neither beacon density contains any scenario in which the average or the minimum location error surpasses 1.5 m.

The boxplot representation of the average and minimum location errors (Fig. 9 bottom) shows that the error values for 20% beacon nodes are lower than for 10%, and even lower for 30%; it is hard to tell whether the differences are statistically significant though, since there exist an important overlap among the boxes. The results of the statistical analysis (Table 12) state that the location errors in instances containing either 20% or 30% beacon nodes are significantly lower than with 10% for any problem instance. However, moving from 20% to 30% beacon nodes only brings significantly lower errors in 6 problem instances, whereas in the remaining 4 instances the error is significantly higher. There-

Table 11
Statistical comparisons of the effect of problem knowledge and beacon reinforcement.

Problem	Test case SA vs. SA - without knowledge										Test case SA vs. SA - Beacon reinforcement									
	Beacon configuration										Beacon configuration									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
3-19A	▽	▽	▽	▽		▽	▽	▽	▽	▽	▽		▲	▲	▲		▲	▽	▲	▲
3-19B	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽							▽	▽		
3-19C	▽		▽	▽	▽	▽	▽	▽	▽	▽				▽	▽			▽	▽	▽
3-19D	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▲	▽	▲	▲		▽	▲	▲		▲
3-19E	▲	▽						▲	▽		▲	▽	▽		▲			▲	▽	
3-19F	▲	▲		▲	▲	▲	▲	▲	▲	▲	▲		▽		▲	▲		▲		
3-20A	▽	▽	▽	▽	▽	▽	▲	▽	▽	▽	▽	▲				▽	▽		▽	▽
3-20B	▽	▽	▽	▽	▲	▽	▽	▽	▽	▽		▲	▲	▲	▲	▲	▲		▲	▲
3-25A	▽					▲		▽			▽	▲	▽				▽			
3-25B	▽	▽	▽	▽	▽	▽		▽	▽		▲	▲	▽	▲	▲	▲	▽		▲	▲

Table 12
Statistical analysis of the influence of the number of beacons (using SA).

Problem	10% beacons vs. 20% beacons	10% beacons vs. 30% beacons	20% beacons vs. 30% beacons
3-19A	▲	▲	▲
3-19B	▲	▲	▽
3-19C	▲	▲	▲
3-19D	▲	▲	▲
3-19E	▲	▲	▽
3-19F	▲	▲	▲
3-20A	▲	▲	▲
3-20B	▲	▲	▲
3-25A	▲	▲	▽
3-25B	▲	▲	▽

Table 13
Statistical comparison of SA against GA and PSO.

Problem	SA vs. GA										SA vs. PSO									
	Beacon configuration										Beacon configuration									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
3-19A	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19B	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19C	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19D	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19E	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19F	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-20A	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽		▽	▽	▽	▽	▽	▽	▽	▽
3-20B	▽		▽	▽	▽	▽	▽	▽	▽		▽	▲	▽	▽	▽	▽	▽	▽	▽	
3-25A	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-25B	▽	▽	▽	▽	▲	▽	▽	▲	▽		▽	▽	▽	▽	▲	▽	▽	▲	▽	▲

Table 14
Statistical significance of the improvement obtained using the two-stage process over single stages with L_1 error norm and maximum likelihood.

Problem	SA - Two Stage vs. SA - Test Case										SA - Two Stage vs. SA - Maxlikelihood									
	Beacon configuration										Beacon configuration									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
3-19A	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19B	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19C	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19D	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19E	▽	▲	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-19F	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-20A	▽	▽	▽	▽	▽	▲	▲	▽	▲	▽	▽	▽	▽	▽	▲	▲	▽	▽	▲	▽
3-20B	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-25A	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
3-25B	▽	▽	▲	▽	▽	▽	▽	▽	▽	▽	▲	▽		▽	▽	▲	▽	▽	▽	▽

fore, we recommend 20% beacon nodes as a good tradeoff value between price and accuracy.

7.5. Results of the different search techniques

In this section we compare the three algorithms described in Section 6: SA, GA and PSO. All three algorithms use with the L_1 error norm function with problem knowledge as their fitness function, and are tested on the same test scenarios (10 instances with ten beacon configurations). The values of the parameters used for SA were shown in Table 3, those of GA and PSO are shown in Tables 7 and 8 respectively. We display the detailed results obtained by GA in Table 9 (top) and by PSO in Table 9 (bottom). Fig. 10 (top) shows the boxplot representation of the global average location error (left) and minimum location error (right) for the three compared techniques. Finally, the statistically assessed comparison between the test case and the two other techniques is shown in Table 13.

The average location errors range from 1.93 to 22.02 m and from 1.16 to 22.45 m for the Genetic Algorithm and Particle Swarm Optimization respectively. For the minimum location errors the values range from 0.33 to 8.29 m and from 0.32 to 5.70 m respectively. GA produces average and minimum location errors above 1.5 m in 100 and 38 scenarios respectively; PSO in 99 and 26 (with SA it is 24 and 4).

The representations in Fig. 10 (top) highlight that SA produces solutions with significantly lower location errors (both average and minimum) than either GA or PSO. In effect, the statistical analysis of the results (Table 13) points out that the solutions obtained with SA have significantly lower location errors than those produced by GA in 95 scenarios (versus 2 scenarios in which their error is higher), and significantly lower location errors than PSO in 94 scenarios (versus 4). Therefore, we can state that SA clearly outperformed both GA and PSO.

7.6. Improvements of the two-stage search process

We finally want to check how much improvement can be attained by using our proposed two-stage search process. For this, we compare the results obtained using a two-stage SA (Table 10, bottom) against the test case results (Table 4) and the maximum likelihood results (Table 10, top). Once again the testbed will be the 100 scenarios described in Section 7.1. For the sake of fairness, we set the total number of solution evaluations in the two-stage process to be 5,000, 000; after an empirical study, the configuration is set as follows: initial stage (using L_1 norm function) 4,000, 000 evaluations, final stage (using likelihood) 1,000, 000 evaluations.

Fig. 10 (bottom) displays the boxplot representation of the average (left) and minimum(right) location errors produced by the two-stage solving process compared to the error norm and the maximum likelihood. Table 14 shows the results of the statistical analysis of the results obtained by our proposed technique compared to the two others.

With the maximum likelihood criterion the average location error obtained ranges from 0.16 to 12.21 m, and the minimum location error ranges from 0.04 to 1.83 m. With the two-stage search process, the average location error obtained ranges from 0.06 to 4.71 m, and the minimum location error ranges from 0.04 to 4.55 m. A special trait that can be noticed is that the location errors can be separated into two distinct groups: the *low* error group (values below 1 m) and the *high* error group (values above 2 m); with hardly any element in between. In fact, when the final solution from the initial phase has a high location error (especially when its location error is above 1.5 m), the second phase does not perform well and the final location error is very close to the one at the end of the initial phase; only when the solution from the initial phase has a low error (below 1 m) does the second phase noticeably improve the accuracy. This can be explained by the fact that the guiding function in the second phase, the likelihood function, is consistent only in the neighborhood of the optimum (see Section 5.3).

From Fig. 10, both the average and minimum location errors obtained by the two-stage search process seem clearly lower than those from the other search processes. The maximum likelihood is clearly behind in terms of average location errors. The results from the statistical analysis state that the two-stage search process is significantly better than the test case in 95% of the test scenarios, and worse in only 5%, and better than the likelihood in 94%, being worse in just 5%. Therefore, we conclude that the two-stage search process produces a real improvement over single phase solving processes; obtaining significantly lower location errors in over 94% of the tested cases.

8. Conclusions

In this paper we have presented and described the location discovery problem (LD). This problem amounts to finding the real locations of a set of sensor nodes, given a set of node-to-node measurements and a set of node coordinates (a subset of the nodes carry GPS-like positioning equipment). Two of the most popular guiding functions, the L_1 error norm and the likelihood functions, are presented and their consistencies are checked for several sets of real data. It turned out that the error norm function outperforms the likelihood function when the current solution has a high location error, but the tide turns when the solution has a low location error; therefore, we propose a solving approach consisting of an initial phase using the error norm followed by a second phase using the likelihood function. Ten instances are selected from the available dataset with ten beacon configurations for each as the test scenarios for our experimental study. We have first studied the effect of the use of problem knowledge in the error norm function, finding that the link weighting produced a significant improvement in the solutions produced. We then tried different search techniques, and found that Simulated Annealing produced significantly better results than Genetic Algorithm and Particle Swarm Optimization. A study on the influence of the number of beacons showed that 20% of beacon nodes configuration provided a good balance between location accuracy and equipment cost. Finally, our proposed two-stage solving process proved to significantly outperform the error norm approach in 95% of the test scenarios, and the maximum likelihood in 94%, therefore proving itself as the most robust and accurate solving procedure. Future work will lead towards an automatic switching procedure from the first to the second stages of the two-stage search process, the use of new search techniques, and improvements in the likelihood estimation.

Acknowledgements

Authors acknowledge funds from the Spanish Ministry of Sciences and Innovation European FEDER under contract TIN2008.0641-C04-01 (M* project http://mstar.lcc.uma.es) and CICE, Junta de Andalucía under contract P07-TIC-0.044 (DIRICOM project http://diricom.lcc.uma.es). Guillermo Molina receives grant AP2005.0914 from the Spanish government.

The authors would also like to acknowledge Dr. Miodrag Potkonjak from the UCLA for all the help and data provided.

References

[1] A. Ákos Lédeczi, P. Nádas, G. Völgyesi, B. Balogh, J. Kusy, G. Sallai, S. Pap, K. Dóra, M. ároly Molnár, G. Maróti, Simon, Countersniper system for urban warfare, ACM Trans. Sen. Netw. 1 (2) (2005) 153–177.

[2] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, New York, 1996.

[3] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[4] J.A. Costa, N. Patwari, O. Alfred, I. Hero, Distributed weighted-multidimensional scaling for node localization in sensor networks, ACM Trans. Sen. Netw. 2 (1) (2006) 39–64.
[5] D. Culler, D. Estrin, M. Srivastava, Overview of sensor networks, IEEE Comput. 37 (8) (2004) 41–49.
[6] K. Deb, R. Agrawal, Simulated binary crossover for continuous search space, Complex Syst. 9 (1995) 115–148.
[7] J. Feng, L. Girod, M. Potkonjak, Location discovery using data-driven statistical error modeling, in: INFOCOM 2006, 25th IEEE International Conference on Computer Communications, Proceedings, April, 2006, pp. 1–14.
[8] B. Karp, H.T. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: MobiCom'00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, ACM Press, New York, NY, USA, pp. 243–254, http://dx.doi.org/10.1145/345910.345953.
[9] J. Kennedy, R. Eberhart, Particle swarm optimization, vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.
[10] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 4598 (220) (1983) 671–680, May.
[11] F. Koushanfar, S. Slijepcevic, M. Potkonjak, A. Sangiovanni-Vincentelli, Location discovery in ad-hoc wireless sensor networks, in: X. Cheng, X. Huang, D.-Z. Du (Eds.), Ad Hoc Wireless Networking, Kluwer Academic Publishers, 2003, pp. 137–173.
[12] F. Koushanfar, S. Slijepcevic, J. Wong, M. Potkonjak, Global error-tolerant algorithms for location discovery in ad-hoc wireless networks, in: Acoustics, Speech, and Signal Processing, 2002, Proceedings (ICASSP'02). IEEE International Conference on 4, IV-4186, 2002.
[13] L. Lazos, R. Poovendran, Serloc: robust localization for wireless sensor networks, ACM Trans. Sen. Netw. 1 (1) (2005) 73–100.
[14] D. Liu, P. Ning, W. Du, Attack-resistant location estimation in sensor networks, in: Fourth International Symposium on Information Processing in Sensor Networks, 2005, IPSN 2005, April 2005, pp. 99–106.
[15] D. Liu, P. Ning, W. Du, Detecting malicious beacon nodes for secure location discovery in wireless sensor networks, in: Proceedings. 25th IEEE International Conference on Distributed Computing Systems, 2005, ICDCS 2005, June 2005, pp. 609–619.
[16] W. Merrill, F. Newberg, L. Girod, K. Sohrabi, Battlefield ad-hoc lans: a distributed processing perspective, GOMACTech, 2004.
[17] N. Mladineo, S. Knezic, Optimisation of forest fire sensor network using GIS technology, in: Proceedings of the 22nd International Conference on Information Technology Interfaces, 2000, ITI 2000, June 13–16, 2000, pp. 391–396.
[18] D. Moore, J. Leonard, D. Rus, S. Teller, Robust distributed network localization with noisy range measurements, in: SenSys'04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, 2004, pp. 50–61.
[19] J. Nemeroff, L. Garcia, D. Hampel, S. DiPierro, Application of sensor network communications, in: Military Communications Conference, 2001, MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force, vol. 1, IEEE 1, 2001, pp. 336–341.
[20] D. Niculescu, B. Nath, Error characteristics of ad hoc positioning systems (aps), in: MobiHoc'04: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, New York, NY, USA, 2004, pp. 20–30.
[21] S. Slijepcevic, S. Megerian, M. Potkonjak, Location errors in wireless embedded sensor networks: sources, models, and effects on applications, SIGMOBILE Mob. Comput. Commun. Rev. 6 (3) (2002) 67–78.
[22] Y.-C. Tseng, C.-F. Huang, S.-P. Kuo, Positioning and Location Tracking in Wireless Sensor Networks, in: M. Ilyas, I. Mahgoub (Eds.), Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, Kluwer Academic Publishers, 2005.
[23] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, D. Estrin, A wireless sensor network for structural monitoring., in: SenSys'04: Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM, New York, NY, USA, 2004, pp. 13–24.