Mobile-Beacon Assisted Sensor Localization with Dynamic Beacon Mobility Scheduling

Xu Li, Nathalie Mitton, Isabelle Simplot-Ryl, David Simplot-Ryl

Mobile-Beacon Assisted Sensor Localization with Dynamic Beacon Mobility Scheduling

Xu Li, Nathalie Mitton, Isabelle Simplot-Ryl, and David Simplot-Ryl
INRIA Lille - Nord Europe, Univ Lille Nord de France, USTL, CNRS UMR 8022, LIFL, France
{firstname.lastname}@inria.fr

***Abstract*—In mobile-beacon assisted sensor localization, beacon mobility scheduling aims to determine the best beacon trajectory so that each sensor receives sufficient beacon signals with minimum delay. We propose a novel *DeteRministic bEAcon Mobility Scheduling* (DREAMS) algorithm, without requiring any prior knowledge of the sensory field. In this algorithm, beacon trajectory is defined as the track of depth-first traversal (DFT) of the network graph, thus deterministic. The mobile beacon performs DFT under the instruction of nearby sensors on the fly. It moves from sensor to sensor in an intelligent heuristic manner according to RSS (Received Signal Strength)-based distance measurements. We prove that DREAMS guarantees full localization (every sensor is localized) when the measurements are noise-free. Then we suggest to apply node elimination and topology control (Local Minimum Spanning Tree) to shorten beacon tour and reduce delay. Through simulation we show that DREAMS guarantees full localization even with noisy distance measurements. We evaluate its performance on localization delay and communication overhead in comparison with a previously proposed static path based scheduling method.**

***Index Terms*—Mobility management, Mobile beacon, Sensor localization, Wireless sensor networks**

I. Introduction

A wireless sensor network (WSN) is a collection of low-cost sensing devices, i.e., sensors, connected by wireless communication links in an ad-hoc fashion. Usually, it is densely deployed in a region of interest for object monitoring and target tracking. In many high-level WSN applications, a reported event, e.g., a fire in a woody area, an enemy tank in a battle field and a survivor in a natural disaster, just to mention a few, is meaningful and can be responded to only if the event position is known. Location awareness also plays an important role in some low-level network functionalities such as geographic routing and data centric storage. Indeed, sensors are expected to know their positions for effective and efficient use of WSNs.

Localization is a fundamental problem dealing with how a sensor determines its spatial coordinates (i.e., position). A straightforward solution is to equip each sensor with a GPS receiver that can provide the sensor with its exact location. But this is not a cost-effective solution because WSN normally contains a massive number of sensors (thus requires a large amount of extra monetary investment). It has limited applicability because GPS works only in open areas with no obstruction to satellite signal (for example, they do not function in underwater or indoor environment), and in some hostile scenario such as battlefield surveillance, GPS signals may be jammed by adversaries and become unavailable to individual sensors unless the sensors are equipped with expensive GPS anti-jam protection techniques [5].

Existing non-GPS-based sensor localization algorithms can be classified as *range-based* or *range-free* [15]. Range-based localization relies on signal features such as Angle of Arrival (AoA), Time of Arrival (ToA), Time Difference of Arrival (TDoA), or Received Signal Strength (RSS) for ranging (i.e., finding nodal relative distance or angle). Unlocalized nodes use reference locations and the range information to estimate their own position by triangulation, trilateration or multilateration, depending on the signal feature used. The reference locations are called *beacons*. They are implemented by location-aware nodes emitting location signals. Range-free localization uses topological information (e.g., hop count) rather than range information. It may or may not use beacons.

Localization that relies on AoA, ToA, or TDoA requires additional complex hardware and/or synchronized transmitters for the signal feature measurement and dramatically increases deployment cost. On the other hand, measuring RSS is free of cost because RSS Indicator (RSSI) is available with regular radio module. In range-free localization without use of beacons (thus any signal feature), nodes are localized in a virtual coordinate system that does not necessarily have direct relation with the ground truth. While virtual coordinates may be an acceptable replacement of physical coordinates for networking, they are less helpful to real life applications where true physical position is required for event response and decision making (e.g., emergency rescue and environment control).

Hence, considerable research attention has been attracted to beacon-assisted localization (whether range-based or range-free) using RSS, for accuracy, efficiency and applicability concerns. Generally speaking, sensors infer their position from their spatial relations with the beacons in range, which are in turn estimated using RSS. To limit positioning possibilities in a minimal region (i.e., to increase localization accuracy), they should have direct contact with a sufficient number of beacons. The number of beacons and their distribution thus has direct impact on the localization performance. A large number of uniformly distributed beacons will lead to better performance than a few crawdedly or linearly deployed ones (which may result in symmetry in localization).

On the other hand, manual beacon placement may not be possible due to operational factors, and meanwhile the number of beacons should be significantly smaller than the number of

sensors because, otherwise, beacon deployment cost will offset the savings on sensor side. Localized sensors can become new beacons and help other sensors self-localize. This iterative method reduces the initial number of beacons required but results in aggregated localization error. Under these circumstances, mobile beacons (wireless communicable devices such as mobile robots and unmanned vehicles) are introduced as alternative cost-saving technique in delay-tolerant scenarios. As we will see in this paper, if one or a few mobile beacons are used properly, full sensor localization guarantee can be achieved.

A. Objective and motivation

In this paper, we address mobile-beacon assisted sensor localization: a location-aware beacon travels in the sensory field, transmitting signals conveying their latest location on a periodic basis; on individual sensors, a localization algorithm is engaged during beacon visit. Some experimental work [2] has already demonstrated the feasibility and applicability of this approach in practice. In this approach, beacon transmission locations correspond to traditional fixed beacons. It is advantageous in terms of deployment cost (only a few beacon nodes are required) and communication overhead (only local communication is involved). These advantages however do not come for free, but in exchange for increased localization delay. It is because sensors can be localized only when they are in direct contact with the mobile beacon and receive sufficient signals from it. Beacon trajectory thus have to be properly planned so as to be shortest in length and meanwhile well cover every sensor for quick, full and accurate localization.

Our focus here is not on individual sensor localization but on *beacon mobility scheduling*, i.e., finding optimal beacon trajectory, which is a variant of the NP-complete Traveling Salesman Problem (TSP). A poor trajectory may result in not only large localization delay but also low localization ratio and high localization error. Due to random node dropping, sensors placement pattern is not known a priori. In a dynamic environment, even if the initial pattern was known, the final node distribution may be different (e.g., moved by wind or animals). Beacon trajectory thus should be planned on the fly rather than beforehand. In the literature, this problem is often purposefully overlooked, or oversimplified by unrealistic assumptions, due to its difficulty stemming from the fact that the position of unknown nodes is not known a priori [9]. To advance state of the art, we study dynamic beacon mobility scheduling in realistic settings. At individual sensor level may any existing localization procedure be adopted as long as it utilizes beacon signals for localization. Below is detailed problem statement.

B. Problem statement

We consider a connected static WSN randomly deployed in the Euclidean plane. The sensory field boundary is not known a priori. Sensors are initially unlocalized. A mobile beacons is stochastically placed in the network to localize sensors. It knows about its own coordinates in a coordination system (not necessarily geographic location) as it moves.

Nodes, whether sensors or beacon, are equipped with an omni-directional antenna. Sensors have the same transmission range; the beacon node has a transmission range not smaller than sensors'. Each node is associated with a unique identifier (ID), e.g., manufacturer serial number, by which it can be distinguished from others. Each sensor node (and beacon) periodically transmits 'hello' (resp., beacon) message carrying its ID (resp., current local position coordinates) and other information necessary for localization. 'Hello' frequency and beacon frequency may be different. Each sensor is equipped with a localization procedure, which can be any algorithm that relies on beacon signals.

The goal is to develop a beacon mobility scheduling algorithm, by which the mobile beacon is able to dynamically determine its trajectory and coordinate with one another such that every sensor receives sufficient beacon signals and becomes localized. The overall beacon mobility scheduling should yield minimum localization delay and minimum communication overhead. Note that here we consider only single-beacon scenario; multi-beacon scenario is part of our future work.

C. Our contributions

We propose a novel localized solution to the above stated problem, named *DeteRministic bEAcon Mobility Scheduling* (DREAMS). To our knowledge, DREAMS is the first algorithm of its kind. In the following we introduce its working principle and summarize our simulation study.

In DREAMS, the beacon node first visits a sensor by random movement, and then performs a depth-first traversal (DFT) on the network graph under currently visited sensor's instruction (a visited sensor, after being localized, recommends an unvisited neighbor for the beacon to visit next). It stops moving when it finds that it returns to the first sensor and the sensor has no unvisited neighbors. During DFT, the beacon node performs intelligent distance-based *heuristic movement* from sensor to sensor following RSS, and sensors run the built-in localization procedure to self-localize using received beacon signals. To shorten the beacon trajectory, DFT may be performed on a local minimum spanning tree (LMST) subgraph, where edges are weighted by RSS, and in addition, unvisited, but localized, sensors may be excluded from DFT if the exclusion does not affect discovery of unlocalized sensors.

We prove that DREAMS guarantees full localization with noise-free pair-wise distance measurement (which is used by beacons for heuristic inter-sensor movement). For the practical scenarios with measurement noise, we conducted extensive simulation study, in comparison with the fixed S-shape beacon trajectory proposal (referred to as StaticPath) [19]. Because both DREAMS and StaticPath are independent of the localization procedure on individual sensors, in simulation we chose to use trilateration for simplicity. We comparatively evaluate their performances on localization quality and localization delay with various configuration parameters. Our simulation results

indicate that DREAMS is alway able to offer 100% localization ratio (full localization), while StaticPath has failure rate at 1.6%. For localized sensors, DREAMS has more than 4 times better localization accuracy than StaticPath. As expected, DREAMS however has larger localization delay, increasing as the network size goes up.

The remainder of the paper is organized as follows. We review previous work on mobile-beacon assisted sensor localization in Sec. II. We present our new algorithm DREAMS in Sec. III and evaluate its performance in Sec. IV. We concludes the paper and discuss our future work in Sec. V.

II. Related work

There exists a rich body of research on sensor localization. Due to space limit, we only review the previous works on mobile-beacon assisted approaches. A comprehensive survey covering other localization methods can be found in [15].

In [13], a mobile beacon periodically transmits a beacon message containing its latest location. Any sensor receiving the message concludes that it must be somewhere around the beacon node with certain probability. For each received beacon message, the sensor obtains the RSSI and uses it together with the beacon location (embedded in the message) to construct a constraint on its own position estimate. Then it applies Bayesian inference to compute its new position estimate from its old position estimate and the new constraint. Its initial position estimate is set to an arbitrary point in the sensory field. Having received all beacon messages, the sensor is localized to a weighted average of these position estimates. In [9], the sensor localization problem is modeled as a non-linear optimization problem, where node locations are to be optimized. The nonlinear objective functions are the differences between pair-wise distances computed from sensor location estimates and range measurements (ToA based). This function-fitting problem is solved using the least square method, which requires an initial guess for the sensor positions. The quality of the initial guess has direct impact on the (convergence) performance of the algorithm. To mitigate this dependence, an iterative incremental approach is suggested. Node location estimates are improved in each iteration. Mobile beacons are used to transform relative positions to absolute locations.

Filtering is a signal processing technique that removes from a signal unwanted component (noise). It has been widely applied in different domains such as image processing and robotics, and is recently adopted for mobile-beacon assisted sensor localization. The idea is to model localization as an online estimation in a non-linear dynamic system and estimate the system state (i.e., sensor location) using range information obtained from filtered wireless signal features. In [11], [18], robust Extended Kalman Filter and unscented Kalman filter are respectively used with RSS as system measurements. In [11], the filter is installed on the beacon node that receives periodical signal from sensors, and localization results are passed to sensors by local communication. In [18], the mobile beacon periodically transmits signal to sensors, and the latter run the filter to localize themselves. In [6], Monte Carlo sampling-based Bayesian filtering is employed with various types of observations such as ranging, AoA, connectivity and their combinations. The filter is run on the beacon node. In [2], [14], a particle filter solution is presented. In [14], the authors proposed to increase filtering efficiency and accuracy by varying the size of sample sets and the parameter of the dynamic model in the estimation process.

In all above mentioned algorithms, beacon mobility scheduling was not studied. Few existing works [8], [17] addressed this problem under strong assumptions. In [8], accurate (noise-free) pair-wise distance measurements are assumed, and localization is by trilateration (requiring three non-collinear reference locations). The idea of mobility scheduling is to provide sensors with reference locations in preferred regions upon their request. It is not clear how the beacon will move when no request is received, while some remote area remains unlocalized. Localization is thus not guaranteed. In [17], the sensory field is known a priori, and unrealistic binary communication model is implicitly used. Two algorithms were presented, respectively using beacon departure/arrival and variance of RSS. Pre-defined 'S' shape trajectory spanning the entire sensory field is suggested to support the two algorithms. In [19], [20], other types of fixed trajectories are suggested under the assumption of known sensory field boundary. In [7], [12], the beacon node first collects the topology information of the network, then with this global view, it identifies the areas where additional distance measures are needed so that in the end the whole network becomes a globally rigid region and is thus localizable by a range-free algorithm. The authors did not address how to physically identify a target area and move the beacon node to that area without position information.

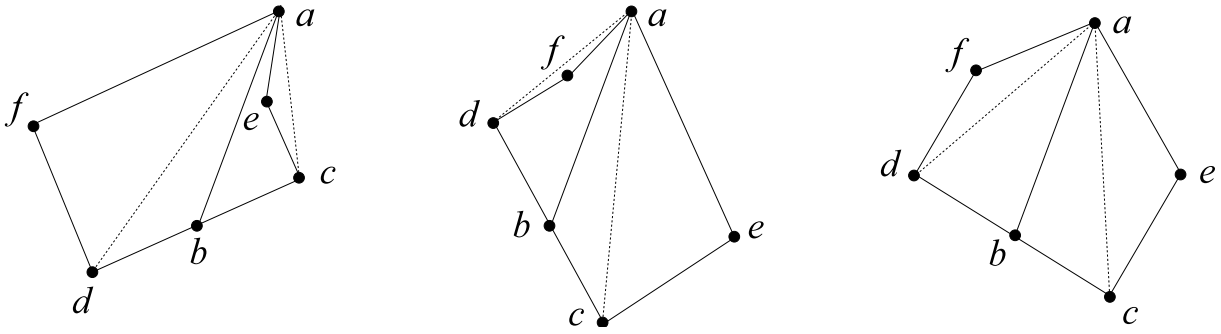
Summarizing, mobile-beacon assisted sensor localization is not well studied. In particular, there is no real solution to the integral beacon mobility scheduling problem, which is our focus here. We propose a novel localized mobility scheduling algorithm, called DREAMS, advancing the literature one step forward. DREAMS dynamically determines beacon trajectory and guarantees full localization, without any prior knowledge of the sensory field boundary. It is a generic framework that can be incorporated with any existing localization algorithm that relies on beacon signals to solve the mobile-beacon assisted sensor localization problem in a complete sense.

III. Beacon mobility scheduling

In this section, we propose a novel *DeteRministic bEAcon Mobility Scheduling* (DREAMS) algorithm, by which a mobile beacon R is able to visit every sensor and provide it with sufficient beacon signals and as a result a fully localized network is guaranteed. This algorithm is deterministic in the sense that sensors' visiting order is fixed provided R starts from the same sensor.

A. The framework (depth-first traversal)

The framework of DREAMS is a process of depth-first traversal (DFT) of the network graph, where R follows instructions (embedded in 'hello' message) from neighboring sensors



(a) $\cos(\widehat{cba}) > \frac{1}{4}$ (b) $\cos(\widehat{cba}) < -\frac{1}{4}$ (c) $-\frac{1}{4} \leq \cos(\widehat{cba}) \leq \frac{1}{4}$

Fig. 1. Theorem 1

and corresponding RSS. Throughout the DFT, R periodically transmits beacon message, while sensors self-localize using received beacon signals.

At initiation, R moves randomly and tries to find an adjacent sensor by listening to 'hello' message. The first sensor discovered is called *root*. It is the sensor that R starts the DFT from, and will be the root of the resultant traversal tree. When multiple sensors are discovered together, sensor ID or RSS can be used as tie breaker. A *target sensor* is a sensor that R is currently moving to. A sensor is *visited* if it has been taken by R as target, and *unvisited* otherwise. The first target of R is the root sensor. R notifies its current target sensor by beacon message so that the latter is able to identify its own status (whether serving as target or not).

When a sensor S is serving as the target of R, it selects as *child* an unvisited neighboring node S' with strongest RSS. A random selection can be alternatively made, or preference can be given to a neighbor with high priority, which can be evaluated, for example, by the importance/intensity of its detected event (embedded in 'hello' message). By 'hello' message, S recommends S' to R as next target; by the same message, it informs S' about their parent-child relationship. If S has no unvisited neighbors, it will recommend its parent (which has already been localized) to R. In the case that S has no parent either (i.e., it is the root node), it informs R about this situation. R finishes the DFT, in other words, DREAMS terminates, when it realizes that its target sensor is localized and neighboring no unlocalized sensors and has no parent.

During the DFT, R proceeds toward each target sensor S by *heuristic movement* (see Sec. III-B for detail) if S is not yet localized. While R is performing this movement, S estimates its own location by the sensor localization procedure and informs R of the result. Once R is informed that S becomes localized, it stops the heuristic movement, moves straight toward S and becomes ready to change target. It changes target to the recommended S' immediately if S' is localized already (in this case, S' is the parent of S), and otherwise as soon as it receives 'hello' message from S'. Then it notifies S about the change via beacon message.

In the case that R has not received signal from S' after arriving at the estimated location of S, it knows that the location estimate of S is not accurate enough and informs S about the inaccuracy. After that, it resumes the heuristic movement toward S, which will lead it either infinitely closer to the true location of S or moving around S (as discussed in Sec. III-B), and eventually discover S'. In this recovery heuristic movement process, R notifies S to update location estimate directly with the reported beacon location if it reaches a position closer to S.

B. Distance-based heuristic movement

Central to the above DFT framework is the heuristic movement of R toward its target sensor S. The heuristic movement relies on the measurement of relative distance of R and S. It is implemented by a sequence of *effective movement procedures (EMP)*, each in turn consists of a few movement steps and moves R to a position closer to S, without resorting to the position information (whether exact or approximate) of S. In the following, we present EMP first with noise-free distance measurement and then with noisy measurement.

1) Noise-free distance measurement: Before we elaborate on EMP, we introduce the following theorem that serves as the foundation of EMP.

THEOREM 1: Given six different points a, b, c, d, e, f in the Euclidean plane such that $b \in cd$, a, e, f are located on the same side of cd, $|bc| = |bd| = \frac{1}{2}|ab|$, $ce \perp cd$, $|ce| = \frac{1}{2}|ac|$, $df \perp cd$ and $|df| = \frac{1}{2}|ad|$, as illustrated in Fig. 1.

1) If $\cos(\angle cba) > \frac{1}{4}$, then $|ac| < |ab|$.
2) If $\cos(\angle cba) < -\frac{1}{4}$, then $|ad| < |ab|$.
3) If $-\frac{1}{4} \leq \cos(\angle cba) \leq \frac{1}{4}$, then $|ae|, |af| < |ab|$.

Proof: For ease of presentation, let $x = |ab|$, $\alpha_1 = \angle cba$, $\beta_1 = \angle acb$, $\vartheta_1 = \angle eca$ ($\angle ace$ in Fig. 1(a)), $y_1 = |ac|$, $z_1 = |ae|$, $\alpha_2 = \angle abd$, $\beta_2 = \angle bda$, $\vartheta_2 = \angle adf$ ($\angle fda$ in Fig. 1(b)), $y_2 = |ad|$, $z_2 = |af|$. Then $\alpha_1 + \alpha_2 = \pi$, $\beta_1 + \vartheta_1 = \frac{1}{2}\pi$ and $\beta_2 + \vartheta_2 = \frac{1}{2}\pi$. Further, define

$$t_1 = \frac{5}{4} - \cos\alpha_1 \text{ and } t_2 = \frac{5}{4} - \cos\alpha_2.$$

In Δabc and Δabd, by the law of cosines we respectively have

$$y_1^2 = x^2 + (\frac{x}{2})^2 - 2x(\frac{x}{2})\cos\alpha_1 = x^2 t_1 \qquad (1)$$

$$y_2^2 = x^2 + (\frac{x}{2})^2 - 2x(\frac{x}{2})\cos\alpha_2 = x^2 t_2 \quad (2)$$

Apparently, if $\cos\alpha_1 > \frac{1}{4}$, then $0 \le t_1 < 1$ and $y_1 < x$; if $\cos\alpha_1 < -\frac{1}{4}$, then $\cos\alpha_2 > \frac{1}{4}$ and $0 \le t_2 < 1$ and thus $y_2 < x$. We remain to show $y_1 < x$ and $y_2 < x$ for $-\frac{1}{4} \le \cos\alpha_1 \le \frac{1}{4}$. Notice that under this condition we have

$$1 \le t_1, t_2 \le \frac{3}{2}.$$

In Δabc we have $x^2 = y_1^2 + (\frac{x}{2})^2 - 2y_1(\frac{x}{2})\cos\beta_1 = y_1^2 + \frac{x^2}{4} - xy_1\cos\beta_1$ by the law of cosines. Then through simple transformation, $\cos\beta_1$ is expressed as

$$\cos\beta_1 = \frac{y_1}{x} - \frac{3x}{4y_1}. \quad (3)$$

Take square root at both sides of Eqn. (1) and plug it in Eqn. (3). We obtain

$$\cos\beta_1 = \sqrt{t_1} - \frac{3}{4\sqrt{t_1}}. \quad (4)$$

We know $\cos\vartheta_1 = \sqrt{1-(\sin\vartheta_1)^2}$ and $\sin\vartheta_1 = \sin(\frac{\pi}{2} - \beta_1) = \cos\beta_1$. Using Eqn. (4), we have

$$\cos\vartheta_1 = \sqrt{1 - (\sqrt{t_1} - \frac{3}{4\sqrt{t_1}})^2} \quad (5)$$

In Δabe, according to the law of cosines and by Eqn. (1) and (5) we have

$$z_1^2 = x^2 t_1 (\frac{5}{4} - \sqrt{1 - (\sqrt{t_1} - \frac{3}{4\sqrt{t_1}})^2}). \quad (6)$$

Our goal is to prove $z_1 < x$. It is equivalent to prove $z_1^2 - x^2 = x^2(\frac{5}{4}t_1 - 1 - t_1\sqrt{1-(\sqrt{t_1} - \frac{3}{4\sqrt{t_1}})^2}) < 0$, which holds if and only if

$$\frac{5}{4}t_1 - 1 < t_1\sqrt{1 - (\sqrt{t_1} - \frac{3}{4\sqrt{t_1}})^2}. \quad (7)$$

Because $\frac{5}{4}t_1 - 1 > 0$, the above inequality can be safely converted to the following inequality by taking square at both sides and then through trivial manipulation.

$$t_1^3 - \frac{15}{16}t_1^2 - \frac{31}{16}t_1 + 1 < 0. \quad (8)$$

Let λ be the left side of this inequality. We have $\lambda' = \lambda - 1 + t_1 = t_1(t_1^2 - \frac{15}{16}t_1 - \frac{15}{16}) > \lambda$. Because the parabola $f(x) = x^2 - \frac{15}{16}x - \frac{15}{16}$ opens up and intersects the X axis at $\frac{15 \pm \sqrt{960}}{32}$ which contains the closed range $[1, \frac{3}{2}]$, we conclude $t_1^2 - \frac{15}{16}t_1 - \frac{15}{16} < 0$, and consequently, $\lambda' < 0$ and $\lambda < 0$. This finally proves that $y_1 < x$ for $-\frac{1}{4} \le \cos\alpha_1 \le \frac{1}{4}$. By the same technique, we can easily prove that $y_2 < x$ for $-\frac{1}{4} \le \cos\alpha_1 \le \frac{1}{4}$. This completes the proof. ■

Now we are ready to present the *effective movement procedure (EMP)*. Suppose that S is located at a and R is initially located at b in Fig. 1. The idea of EMP is to move R to position c in a random direction, and if c is not closer to a than b, further move R to either e or f (depending on $\angle abc$). From Eqn. (1), $|ac|$ is a monotonically decreasing function of $\cos(\angle abc)$. Plug the threshold values $\pm\frac{1}{4}$ of $\cos(\angle abc)$ in the equation, we obtain the corresponding threshold values (in terms of $|ab|$) of $|ac|$, i.e., $|ab|$ and $\frac{\sqrt{6}}{2}|ab|$, which is used in EMP by R to make movement decision as it is not able to measure angle. The pseudo codes of EMP are given below.

```
Procedure: Effective_Movement_Procedure
Input    : S, target sensor; dir, direction of 1st step
Output   : direction of the last movement step

/*****all distances used are error-free measurement*****/
p = current position of R;
move a distance of 1/2|pS| to q1 in dir;
if |q1S| < |pS| then
   return dir;
else if |q1S| > sqrt(6)/2|pS| then
   opp_dir = the opposite of dir;
   move a distance of |pS| to q2 in opp_dir;
   return opp_dir;
else
   per_dir = a direction perpendicular to dir;
   move a distance of 1/2|q1S| to q2 in per_dir;
   if |q2S| < |qS| then
      return per_dir;
   else
      opp_per_dir = the opposite of per_dir;
      move a distance of |q2S| in opp_per_dir;
      return opp_per_dir;
   endif
endif
```

As we see, EMP takes target sensor S and an arbitrary direction as input. It involves at most three movement steps. The first step is performed toward the inputed direction. It returns the direction of the last step. EMP guarantees progress (toward S) according to Theorem 1. In the course of heuristic movement to S, R runs EMP iteratively, each time with S and an arbitrary direction (or the output direction of previous EMP iteration or a direction perpendicular to it) as input. As the procedure iterates, R moves infinitely closer to S. It can be considered having reach S if its distance to S is sufficiently small (e.g., with respect to a given threshold).

THEOREM 2: With accurate pair-wise distance measurement, DREAMS guarantee full localization.

Proof: The heuristic movement enables the mobile beacon to reach a target sensor by Theorem 1, which is localized in the worst case when the beacon arrives at its position. The DFT ensures every sensor to be taken as localization target and therefore localized by the mobile beacon. ■

2) Noisy distance measurement: The above introduced EMP relies on accurate distance measurement. It can not be applied directly in reality, as the measurement is always inaccurate/nosily. Measurement noise may lead R to a wrong direction, away from S, or contradictionary situations that are impossible in theory. In the following, we are going to modify EMP to accommodate measurement noise. We begin with how to measure pair-wise distance.

R estimates its distance to S simply using the RSS of 'hello' message from S and the log-distance signal propagation model [16] reversely. According to this propagation model, the average power at distance d from a wireless transmitter is

$$\mu(d) = \beta - 10\alpha\log d, \quad (9)$$

where α is a slope describing signal strength change rate,

(a) Heuristic movement with distance noise

(b) Dead end

Fig. 2. Noisy distance measurments

and β is a constant determined by the transmitted power, wavelength, and antenna gain of the transmitter. RSS is a combination of transmission power and noise due to various factors such as shadow fading and interference. Because of the noisy nature of RSS, multiple RSS samples are necessary for smoothing the readings and reducing estimation error. The average or mean RSS is finally used for distance estimation.

We shall now modify EMP to mitigate its dependence on the accuracy of distance measurement and reduce the impact of measurement noise on the performance. The modification includes removal of precise distance thresholds and addition of exception handling (by moving R back to its initial position). The new EMP takes the same input parameters and returns the same output as the old one. But unlike the old EMP, it may return NULL, implying that no progress is made and R stays at the original position. As the old EMP, this new procedure is run in iterations by R to approach S. If one iteration returns NULL, the next iteration will take a random direction as input. In each EMP, some sojourn time is needed between steps for distance estimation. Below are the revised pseudo codes.

```
Procedure: New_Effective_Movement_Procedure
Input    : S, target sensor; dir, direction of 1st step
Output   : direction of the last movement step

/*****all distances used are error-prone estimate*****/
p = current position of R;
move a distance of (1/2)|pS| to q1 in dir;
if |q1S| < |pS| then
  return dir;
else
  opp_dir = the opposite of dir;
  move a distance of |pS| to q2 in opp_dir;
  if |q2S| < |pS| then
    return opp_dir;
  else if |q2S| = ∞ then
    return Exception_Handling(p);
  else
    per_dir = a direction perpendicular to dir;
    move a distance of (1/2)|q2S| to q3 in per_dir;
    if |q3S| < |pS| then
      return per_dir;
    else
      opp_per_dir = the opposite of per_dir;
      move a distance of |q2S| to q4 in opp_per_dir;
      if |q4S| < |pS| then
        return opp_per_dir;
      else
        return Exception_Handling(p);
      endif
    endif
  endif
endif

Subroutine: Exception_Handling
Input     : dest, movement destination
Return    : direction of the movement

move to dest;
return NULL;
```

In Fig. 2(a), the RSS gradient lines of S are shown by dashed curves. R starts heuristic movement toward S from position p. In the first iteration of EMP, it visits $q_1, \cdots, q_4$ in order, as indicated by the arrowed lines, and finally returns to p (no progress is made). In the second iteration, it visits q_5 and then q_6, which finally appears closer to S than p. From q_6, taking a direction perpendicular to the output direction of the second EMP iteration, R advances to q_7 through an third iteration of EMP. From there R continues EMP iteratively, which are now shown in the figure. Due to distance measurement noise it is possible that R is stuck at a *dead end* and never able to reach S, as discussed below.

Figure 2(b) illustrates a dead-end situation. The dashed curve enclosing S closely stands for the RSS gradient line through R's current position p. The dashed circle centered at S is the corresponding RSS gradient line in the deal case. R estimate its distance to S to be d, while the true distance is less than $\frac{1}{2}d$. In any EMP iteration started at p, R always moves to a position on the solid circle of radius $\frac{1}{2}d$ and centered at p at the first step, and at the last step it always return to p. Notice the fact that R is actually attempting to move from p to an area of high-RSS, which is however enclosed in the solid circle. Because its step destinations are never inside the circle, EMP fails always. But nevertheless, R may pass through the high-RSS area in the course of EMP, depending on the input direction. Thus we modify the exception-handling subroutine as follows to recover from dead end.

```
Subroutine: Exception_Handling
Input     : rss, reference RSS;
            S, target sensor; dest, destination
Output    : direction of the movement

move to dest;
step_dir = direction of the movement;
while moving to dest
  cur_rss = RSS of the 'hello' message from S;
```

```
    if cur_rss > rss then
      stop;
      return step_dir;
    endif
  endwhile
  return NULL;
```

The new exception handling subroutine has three input parameters, i.e., target sensor S, beacon initial position p and RSS of S at p. With this subroutine, after a number of trial EMP, if not at the first EMP, R will pass through the high-RSS (relative to rss) area and get out of the dead end. Because of the opportunistic nature of this dead end recovery method, large delay may occur. However, recall that in DREAMS the ultimate goal of the heuristic movement (iterative EMP) is not to move R to the position of S but provide S with sufficient beacon signals or discover the recommended next target sensor around S. Once the goal is achieved, R will move away from S for a new target even if it has not yet recovered from the dead end. Later, in Sec. IV we show through simulation that DREAMS guarantees full localization despite the presence of measurement noise and evaluate its delay performance.

C. Optimization techniques

In the rest of this section, we will present two optimization techniques, *node elimination* and *topology control* for truncating beacon trajectory and reducing localization delay. They can be used in combination for the best performance.

1) Node elimination: While the beacon node R is localizing its target sensor (by iterative EMP), nearby sensors may overhear its signals and become localized as a side effect. This is very likely to happen since sensors are usually densely deployed. Those by-accident localized sensors remain unvisited. Intuitively, unvisited, but localized, sensors should not be included in the beacon trajectory. Thus we propose to improve the algorithm performance by *node elimination*. Eliminated nodes are not recommended to R as target and thus not visited. Notice that the network graph (or subgraph) that R is traversing is a navigation graph, and its connectivity is critical to full localization. Node elimination needs to be done with caution; otherwise, it may disconnect the navigation graph and causes localization failure.

A localized, but unvisited, sensor eliminates itself from the navigation graph if it has degree 1 (eliminated neighbors do not count) or all its neighbors are either visited or *eliminated*. The rationality is that, visiting such a node does not contribute to localization, and removing it does not affect the connectivity of the unvisited part of the graph. To perform self-elimination, a sensor changes its status to *eliminated*. Node elimination can cause chain effect: a sensor that does not satisfy the elimination conditions becomes qualified after (some of) its neighbors are eliminated and then performs self-elimination. As such, the size of the navigation graph is dynamically reduced, and beacon trajectory is consequently shortened.

It is often beneficial to have a spanning tree structure (e.g., for efficient communication) over the network. Eliminated nodes will apparently not be included in the DFT tree. But they can be easily linked to the tree through local operations as follows. When a node performs self-elimination, it attaches itself to one of its neighbors that are not yet eliminated by taking that neighbor as parent. It informs the selected parent node about their parent-child relation by 'hello' message. This parent node is later either eliminated (and attached to another node) or visited (thus attached to the DFT tree). Cascaded attachment ensures all eliminated nodes to be linked to the traversal tree without loop.

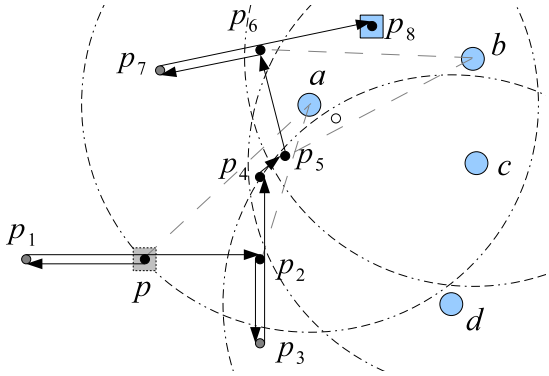
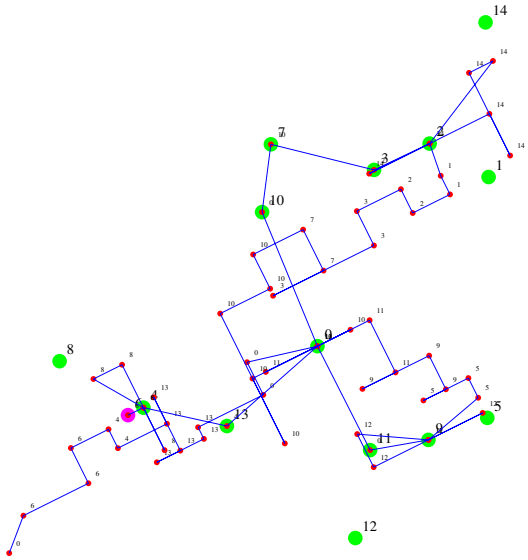


Fig. 3. Node elimination

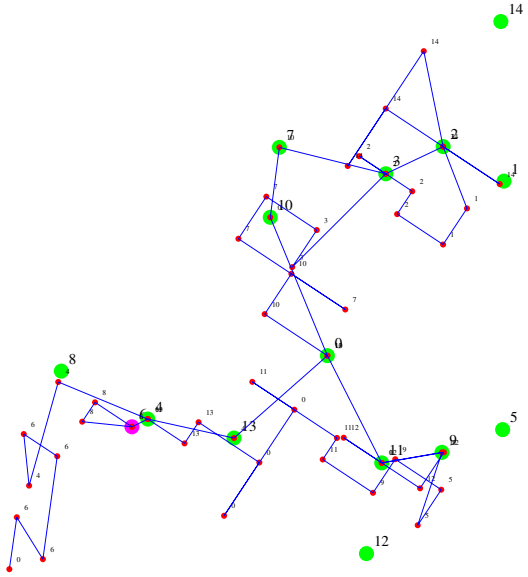
Figure 3 illustrates node elimination in a network of 4 nodes, with trilatertion as individual sensor localization procedure for simplify and beacon transmission only at step destinations. Their communication ranges are indicated by dashed circles. The beacon node R starts from p and visits $p_1, \cdots, p_8$ in order in 8 steps. After R reaches p_4, a becomes localized because there exist 3 non-collinear reference points p, p_2, and p_4 in its communication range. R keeps moving toward the position estimate of a (shown as a small hollow dot beside a) by the algorithm. At p_5, it receives 'hello' message from b, which is the node suggested by a to visit next.

Then R changes its target to b and decides to move to p_6 (a random decision). At the same time, a becomes localized due to the three non-collinear reference locations p_2, p_3 andp_5 and then performs self-elimination since it has only one neighbor c. When R arrives at p_8, both b and c become localized because of p_5, p_6, and p_8. c has three neighbors, a, b and c. Because c is eliminated and a and b are both visited, b eliminates itself. After c is eliminated, b has no unvisited neighbor and thus recommends to R its parent a, which is the root node, as next target. Because a now has no unvisited neighbors either, R stops moving and stays at p_8. The final tree structure is $a - b - c - d$ with a being the root. The edge ab is defined by DFT, while the rest are defined by node elimination.

2) Topology control: To minimize localization delay, R should travel a shortest tour that covers all the sensors in the network. This is in general the Euclidean *Traveling Salesman Problem* (TSP), which is known to be NP-complete. An informal definition of the TSP problem is the following: given a number of cities (i.e., sensors), find the shortest tour that visits each city exactly once and returns to the starting city. It is well known that DFT on Euclidean *minimum spanning tree* (MST) (with repeated visit being removed from the sequence) defines



(a) DFT on original graph



(b) DFT on LMST subgraph

Fig. 4. Beacon trajectory

a 2-approximate solution to Euclidean TSP [4]. Motivated by this result, we naturally consider to run DREAMS on Euclidean MST instead of the original network graph. In our context, the repeated visits skipped for TSP approximation have to be allowed because they may be necessary for R to receive signal from unvisited sensors and continue DFT.

An MST is a subgraph connecting all the network nodes with weighted edges that lead to minimum total weight. In Euclidean MST, each edge is weighted by Euclidean distance between the two end nodes. Sensors may distributedly (e.g., by [1], [3]) construct an MST according to the mutual agreement on RSS with each of its neighbors (which reflects their relative distance, here no RSS-to-distance conversion is needed). The resultant MST is thus called *RSS MST*. It is an approximation of the Euclidean MST. MST construction is an expensive global computation process. To preserve the localized nature of DREAMS, we consider using a localized graph that is similar to MST. A natural option is local MST (LMST) [10], which contains MST as subgraph. To compute LMST, each node computes the MST of its own neighborhood. An edge between two nodes belongs to the LMST if and only if the two nodes are in each other's local MST.

In Figure 4, a network of 15 nodes (numbered 0-14) are randomly deployed in a 100×100 area. Their transmission radii are 25. The two sub-figures show the complete beacon trajectory respectively over the original network graph and LMST subgraph. The mobile beacon starts from the lower-left corner and performs heuristic movement (all movement steps are shown). The mobile beacon starts from the lower-left corner; the first sensor discovered by it is shown as a dark node. The destination of beacon movement steps is shown indicated by a small dot. The number beside it is the ID of the corresponding target sensor. If we count the number of the small dots, we will find that LMST-based trajectory involves 55 beacon steps while the non-LMST-based has 63 steps. From the figure, it is no clear what is the saving in the length of beacon trajectory. A detailed performance evaluation will presented in Sec. IV.

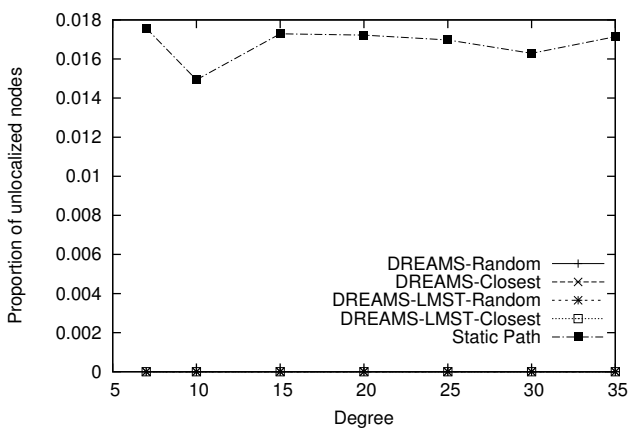
IV. Performance Evaluation

In this section, we evaluate the performance of DREAMS through simulation. Beacon mobility scheduling is not well studied in the literature. DREAMS is the first localized dynamic deterministic solution designed for realistic settings (without prior knowledge of sensory field and with RSS measurement noise). It is independent of individual sensor localization procedure; the static path method is the only existing work that has this property. Hence we decide to use fixed beacon trajectory for comparison. In particular, we chose S-shape StaticPath as it is reported to have most stable performance [19].

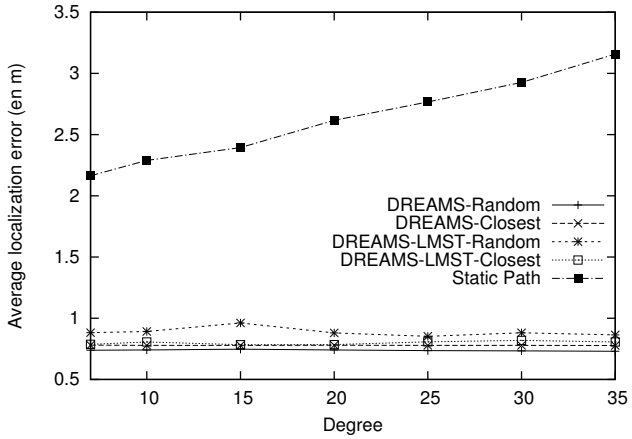
'Hello' message is regarded as built-in technique. Beacon message is a common tool of all beacon-assisted localization algorithm. Both are constant in size and propagate only one hop. Given fixed transmission frequency, their resultant communication cost is countable and thus out of our interest. Our simulation study focuses on the following two aspects.

Localization performance: It is measured by *failure rate* and *localization error*. The former is the average ratio of the number of unlocalized sensors (due to insufficient beacon signals) to the total number of sensors after the scheme terminates; the latter is the average distance between the estimated and the real positions of sensors.

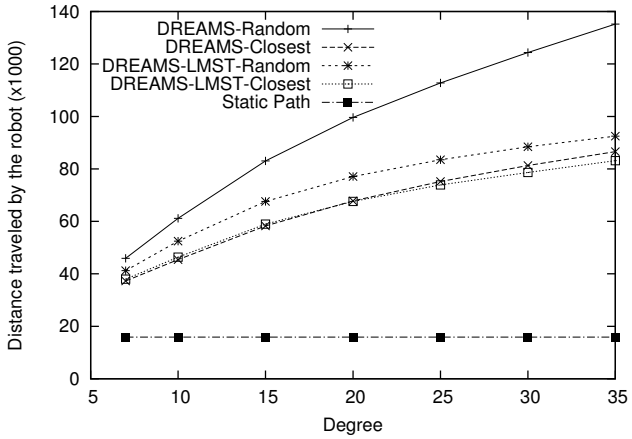
Delay performance: It is measured by *moving distance* and *number of movement steps*. The former is the total distance that the beacon travels during the course of localization. Movement steps are the steps in EMP. It is a metric only for DREAMS, where there is waiting time between two successive steps and thus additional delay.



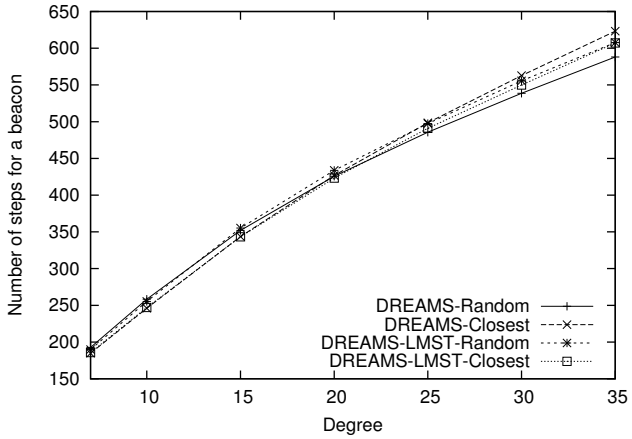
(a) Failure rate.



(b) Mean localization error.



(c) Moving distance.



(d) Movement steps.

Fig. 5. Simulation results.

A. Simulation setup

We implemented StaticPath [19] and DREAMS in a custom C simulator. We implemented the following four DREAMS variants, which differ in the way that a visited sensor chooses a neighbor to recommend to the beacon node as next target:

- DREAMS-Random:
  choose a unvisited neighbor randomly;
- DREAMS-Closest:
  choose a RSS-strongest unvisited neighbor;
- DREAMS-LMST-Random:
  choose a unvisited LMST neighbor randomly;
- DREAMS-LMST-Closet:
  choose a RSS-strongest unvisited LMST neighbor.

In heuristic movement, an EMP iteration takes a perpendicular direction of the output direction of previous iteration as input if the out direction is not NULL, and an random direction otherwise. Because StaticPath and DREAMS are independent of the individual sensor localization algorithm, we choose to use trilateration (which requires three non-collinear beacon locations and pair-wise distance) for simplify in both algorithms. Study with other localization algorithms is for future work.

For both StaticPath and DREAMS, a sliding-window technique is used in trilateration-based sensor localization for error removal. Each sensor applies a sliding window of size 3 on the received sequence of beacon signals. Using the signals in the window, it computes its own location (if computable), and compare the result with the average of previous results. If the difference is less than a threshold, set to 0.5 in our simulation, it localizes itself to the new average position. A sensor is localized until its position estimate is satisfactorily (with respect to the above threshold). In StaticPath, it is localized to the latest average regardless.

In our simulation, we deployed a connected WSN of size n randomly in a 1000×1000 square region. We varied n in such a way that the average node degree ranges from 7 to 35. A mobile beacon is randomly placed in the network. It has constant speed 0.08 per simulated time unit. The beacon and sensors have same communication range 100. RSS reading is the summation of the true value and a zero-mean Gaussian noise of variance equal to 20% of the true value. We use the same 'hello' frequency and beacon frequency: one message

per $1+\delta$ simulated time units, where δ is a random value in range $[-0.2, +0.2]$. For each setting, we conducted 1000 simulation runs, each with a randomly generated sensor and beacon distribution.

B. Simulation results

Fig. 5(a) plots the proportion of nodes that can not be localized, i.e., that know less than 3 non-collinear beacon locations, or that could not compute a reasonable result due to RSS noise. As expected, all DREAMS variants present a 0% failure rate (i.e., full localization) since the beacon visits every node. This is not the case for StaticPath for which 1.6% of nodes do not receive enough beacons and remain unlocalized. Fig. 5(b) plots the mean localization error noticed. One could observe that for every DREAMS variant, the error is independent of the network density (node degree) and remains very low (bellow 1, close to 0.5). This is due to the fact that a beacon will move around a target node as long as it is necessary to achieve this accuracy. This is not the case for the StaticPath scheme for which the error is much higher and increases with the number of nodes. This can be explained by the fact that in this latter scheme, the beacon trajectory depends neither of the node status (localized or not) nor of the number of nodes, and thus nodes are more likely to receive insufficient beacon messages for localization.

Fig. 5(c) shows the average moving distance. By StaticPath, the beacon always travels the same distance whatever the network size is. In DREAMS, the more sensors, the longer the distance to travel. For DREAMS-Random, the distance is the longest. This can be explained by the fact that a sensor may recommend for the beacon to visit a neighbor far from the current target. Since in LMST, edges are generally shorter than in the original graph, the mean travel distance of DREAMS-LMST-Random is smaller than DREAMS-Random. In DREAMS-Closest and DREAMS-LMST-Closest, it is yet smaller since a node always recommends its closest neighbor. All DREAMS curves tend to be flat as node degree increases because the higher the node density the more probably nodes are localized without being targeted by the beacon. Fig. 5(d) plots the total number of movement steps by the beacon, which has the same trend as moving distance for the same reasons.

This is worth noting that delay performance greatly depends on the localization accuracy required. The more accurate the computed position, the more movement steps and the larger moving distance involved. We chose to present only results for a accuracy below 1 due to space limit.

V. Conclusions

While beacon mobility scheduling is long considered a great difficulty in mobile-beacon assisted sensor localization, we have shown in this paper that it can be surprisingly accomplished by using the sensor network as a navigation graph. We proved that our solution, named DREAMS, guarantees full localization when each beacon is able to measure pair-wise distance accurately and showed through simulation that this property preserves even when measurement noise is present.

DREAMS was presented without considering sensor failures, which may disconnect the navigation graph and lead to localization failure. A simple fault-tolerant technique is random walk. That is, when the beacon node fails to receive instructions from sensors for its next movement step, it starts to move randomly to find a unlocalized sensor. Once it finds one, it will re-run DREAMS from there again. DREAMS is currently designed for scenarios that there is a single mobile beacon. In the future, we will extend it to multi-beacon scenarios, where the main challenge is localized beacon coordination.

Acknowledgment

This work was partially supported by a grant from CPER Nord-Pas-de-Calais/FEDER Campus Intelligence Ambiante and the ANR SensLAB project.

References

[1] B. Awerbuch. “Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election, and Related Problems”. In Proc. ACM STOC, 1987.

[2] F. Caballero, L. Merino, I. Maza, and A. Ollero. “A Particle Filtering method for Wireless Sensor Network Localization with an Aerial Robot Beacon”. In Proc. of IEEE ICRA, pp. 596-601, 2008.

[3] R. G. Gallager, P. A. Humblet, and P. M. Spira. “A distributed algorithm for minimum-weight spanning trees”. ACM TOPLAS, 5(1):66-77, 1983.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. Introduction to Algorithms. The MIT Press, 2002.

[5] GPS Anti-Jam Protection. http://defense-update.com/products/g/gps-aj.htm.

[6] R. Huang and G. V. Zaruba. “Monte Carlo localization of wireless sensor networks with a single mobile beacon”. Wireless Networks, 15(8):978-990, 2009.

[7] S. Kanchi and C. Wu. “Robot assisted localization of sensor networks guided by rigidity”. In Proc. of WICON, 2008.

[8] K. Kim and W. Lee. “MBAL: A Mobile Beacon-Assisted Localization Scheme for Wireless Sensor Networks”. In Proc. ICCCN, pp. 57-62, 2007.

[9] M. Kushwaha, K. Molnar, J. Sallai, P. Volgyesi, M. Maroti, A. Ledeczi. “Sensor Node Localization Using Mobile Acoustic Beacons”. In Proc. of IEEE MASS, 2005.

[10] N. Li, J. Hou, and L. Sha. “Design and analysis of an MST-based topology control algorithm”. IEEE Trans. on Wireless Communications, 4(3): 1195–1206, 2005.

[11] P.N. Pathirana, N. Bulusu, A.V. Savkin, and S. Jha. ”Node Localization Using Mobile Robots in Delay-Tolerant Sensor Networks”. IEEE Trans. on Mobile Computing, 4(3):285-295, 2005.

[12] N.B. Priyantha, H. Balackrishnan, E. Demaine, and S. Teller. “Mobile-Assisted Localization in Wireless Sensor Networks”. In Proc. of IEEE INFOCOM, 2005.

[13] M. Sichitiu and V. Ramadurai. “Localization of Wireless Sensor Networks with a Mobile Beacon”. In Proc. of IEEE MASS, 2004.

[14] G. Teng, K. Zheng, and W. Dong. “Adapting Mobile Beacon-Assisted Localization in Wireless Sensor Networks”. Sensors, 9:2760-2779, 2009.

[15] J. Wang, R.K. Ghosh, and S.K. Das. “A survey on sensor localization”. Journal of Control Theory and Applications, 8(1):2-11, 2010.

[16] H.H. Xia. “A simplified analytical model for predicting path loss in urban and suburban environments”. IEEE Trans. on Vehicular Technology, 46(4): 1040–1046, 1997.

[17] B. Xiao, H. Chen, and S. Zhou. “A Walking Beacon-Assisted Localization in Wireless Sensor Networks”. In Proc. of IEEE ICC, 2007.

[18] L. Zhang, Q. Cheng, Y. Wang, and S. Zeadally. “A Novel Distributed Sensor Positioning System Using the Dual of Target Tracking”. IEEE Trans. on Computers, 57(2):246-260, 2008.

[19] R. Huang, G.V. Zaruba. “Static Path Planning for Mobile Beacons to Localize Sensor Networks”. In Proc. of IEEE PerCom workshops, 323-330, 2007.

[20] D. Koutsonikolas, S.M. Das and Y.C. Hu. “Path Planning of Mobile Landmarks for Localization in Wireless Sensor Networks”. Computer Communications, 30(13):2577-2592, 2007.