A Study of Multidimensional Scaling Algorithms for Node Localization in Sensor Networks

Performance Assessment of Iterative Majorization and Improvements Thereof

Diploma Thesis

By

Manolis Koufidakis

Submitted to the Department of Electronic Engineering & Computer Engineering

Technical University of Crete

Advisor: Professor Sidiropoulos Nikolaos

Co-advisor: Associate Professor Liavas Athanasios

Co-advisor: Associate Professor Karystinos Georgios

CONTENTS

1. ABSTRACT

The problem of node localization from pairwise distance estimates has been of interest in many scientific fields, from psychometrics to signal processing and communications. Given a matrix of pairwise distances the localization problem asks to determine the relative node locations that generate these distances. Many different algorithms have been implemented in order to solve the localization problem without having to rely on Global Positioning System (GPS) which is expensive to implement in each node and requires Line of Sight (LOS) to three satellites. Sensor maps are useful for estimating the spatial distribution of measured phenomena, and for routing purposes. In this work we analyze, test and try to improve the distributed-weighted multidimensional scaling algorithm (DW-MDS) Costa, Patwari and Hero) [1], [2]. Furthermore, we compare it to the Fastmap-MDS algorithm (Latsoudas and Sidiropoulos) [3], pointing out the relative strengths and weaknesses.

The DW-MDS algorithm adaptively emphasizes the most accurate range measurements and naturally accounts for communication constraints within the sensor network. Each node adaptively chooses a neighborhood of sensors, updates its position by minimizing a local cost function and then passes this update to the neighboring sensors. On the other hand, Fastmap-MDS is a two-stage algorithm that combines algebraic initialization and gradient descent. An algebraic solution from the database literature is borrowed and it is adapted to the sensor network context, using a specific choice of anchor/pivot nodes. The resulting estimates are fed to a gradient descent iteration.

2. INTRODUCTION

Given a matrix of pairwise distance estimates between nodes, it is of interest to generate a map of node locations. This problem is widely applicable in wireless sensor networks. The initial knowledge of the distances can be obtained using Received Signal Strength (RSS), or Time Of Arrival (TOA) measurements and a path loss model. In its general form the above problem originates in the psychometrics field, and is known as Multi-Dimensional Scaling (MDS).

Researching the field of localization, one can find numerous different algorithms and solutions for the above problem. In this research we analyze, test and compare Distributed-Weighted Multidimensional Scaling algorithm (DW-MDS) Costa, Patwari and Hero [1] and the Fastmap-MDS algorithm (Latsoudas and Sidiropoulos) [3].

The DW-MDS is a method based on a weighted version of multidimensional scaling (MDS), which naturally incorporates local communication constraints within the sensor network. In DW-MDS the estimation of all sensor coordinates can be achieved while only a small fraction of sensors have a priori coordinate knowledge, and range measurements between many pairs of neighboring sensors are available.

While angle measurements have also been used for sensor localization, in DW-MDS the localization is based on range measurements. Its key features are:

1. a weighted cost function that amplifies or degrades each nodes contribution depending on the accuracy of the distance measurement;

2. an adaptive neighbor selection method in order to avoid biasing effects of selecting

neighbors when working on a noisy environment and;

3. a majorization method that assures that each iteration improves the value of the cost function.

On the other hand, the Fastmap-MDS algorithm incorporates both algebraic initialization and gradient descent. An algebraic method, borrowed from the database literature, is used to produce initial position estimates, which are fed to a gradient descent iteration. This sequence assures better performance and lower complexity than other full-connectivity algorithms. What is more, its performance is relatively close to the corresponding Cramer-Rao bound, especially for low values of range error variance. The complexity of the hybrid algorithm is $O(pmN^2), p << N$.

In their work, Costa, Patwari and Hero [1] propose ways of improving DW-MDS performance, by using suitable initialization techniques, as opposed to random initialization. In fact they propose the Savarese et al. [5] or the Capkun et al. [7] algorithm as two possible options. After implementing both of these methods, we reached in the conclusion that they cannot be used for initialization, as they often offer complex valued coordinate estimates when noise is present in the measurements [1].

As the previous methods do not work in our case, we use the Fastmap algorithm from the database literature in order to initialize DW-MDS and the results show better performance, both in terms of precision in position estimate and in terms of time consumption in order to produce the final constellation[2].

Furthermore, we propose a method of improving the performance of DW-MDS by changing the original algorithm and using a different technique for aligning the final constellation. More specifically, we now include both the unknown and anchor nodes in the estimation procedure despite the fact that for the anchor nodes we have almost perfect

[1] Sections 4.1 / 4.2

[2] Section 4.3

a priori knowledge. After the convergence of the DW-MDS and the retrieval of the final constellation, we use the alignment method proposed by Ji and Zha [9]. The Modified DW-MDS algorithm offers better performance in terms of estimation accuracy though it requires more time to finish the estimation process because of the extra computational load.[3].

2.1 Sensor Localization Challenges

For a large network (in the scale of thousands nodes) a centralized location estimation seems a non-applicable approach. Having one central sensor to which all the peripheral nodes send pair-wise distance estimates, and the sent back of the estimated coordinates by the same node would overwhelm the usually low bandwidth capacity of the entire network. In order to limit the communication and balance communication as well as computational load across the sensors in the network, the use of decentralized algorithms is necessary. Furthermore, when a sensor is in move, the ability to recalculate location locally rather than globally will result in substantial energy savings.

Several tradeoffs are encountered during the effort to improve estimation of the sensors positions. One of these is the tradeoff between energy cost and accuracy. For a given channel between two nodes, the accuracy of the calculated pairwise distance can be improved by increasing the SNR of the received signal through the increase of the transmit power. Another tradeoff is the one between the device cost and range accuracy. This is a field on which we will not refer to in the rest of the research, but it is worth to be mentioned. Using ultrawideband (UWB) (Fleming and Kushner 1995; Correal et al. 2003, [12]) or hybrid ultrasound/RF techniques (Girod et al. 2002, [13]) can achieve

[3] Section 5

accuracies on the order of centimeters, but at the expense of high device and energy costs. Alternatively, inexpensive wireless devices can measure RF RSS just by listening to the network packet traffic. However, range estimates from RSS may cause significant errors due to channel fading.

Another factor to be taken into consideration is that all range measurements tend to degrade in accuracy while distance between nodes is increasing.

3. FASTMAP-MDS ALGORITHM

In Latsoudas and Sidiropoulos [3] proposed Fastmap-MDS algorithm, the basic version of node localization problem is revisited by proposing a two-stage algorithm which combines algebraic initialization and gradient descent.

More specifically, the algebraic initialization is based on the Fastmap [4] algorithm, borrowed from the database literature. The produced estimates are forwarded to a gradient descent iteration. Fastmap is a linear-complexity tool. It is sensitive to range measurement errors due to the fact that crude distance measurements are used for the mapping, and because no weighting procedure is adopted in order to mitigate the possible errors due to noise or other environmental interference. In a 3-D scenario and with one node position already known, another node can define its position on the surface of a sphere having as semidiameter the measured distance from the first node. Distances as measured values are invariant to rotation, reflection and shift. Thus, there is a need to adopt at least $m + 1$ anchor/pivot nodes (where m denotes the dimension of the working space) whose position is accurately known in order to estimate the constellation correctly.

Since the estimated position of every node is based only on distances to the selected anchors, there is no averaging and this makes the algorithm sensitive to coordinate alignment. However, this problem can be bypassed by choosing three(since 2-D) pivot nodes to be on the outer edges of the network, forming an orthogonal triangle.

1. The FASTMAP algorithm

The basic element of Fastmap [4] is the projection of the objects on a properly selected line, defined by two anchor nodes. Define these nodes to be O_a,O_b. A pair of anchors is chosen for each of the m dimensions. In our research a 2-dimension model is used. The coordinates of the objects can be found by employing the cosine law. Thus, the first coordinate for object O_i is given by (3.1).

$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{bi}^2}{2d_{ab}} \tag{3.1}$$

where d_{ij} is the measured distance between nodes i and j and a,b are the pivot objects. After computing these coordinates for each node O_i, let us consider a hyperplane which is orthogonal to the pivot line and project the objects on this hyperplane. Repeat the previous process, this time using the new distance estimates as they are re-computed (3.2) after the first estimation of the x-coordinate of each unknown node.

$$d_{ij}^{2\,'} = d_{ij}^2 - (x_i - x_j)^2, i, j = 1, ..., N \tag{3.2}$$

As mentioned above, the estimated position of every node is based only on distances to the selected anchors and there is no averaging. As a result, the algorithm is sensitive to coordinate alignment. By selecting the pivots to be on the outer edges of the network, this problem can be overwhelmed. More specifically, the pivots are placed wittingly in a way they an orthogonal triangle(Figure 3.1). In other words, we use the pivots in order to simulate an orthogonal coordinates system. This placement can provide a high-quality initialization to the gradient descent because there are no alignment errors. Anchors #1 and #2 serve as pivots for determining the coordinates in the first dimension, while anchors #2 and #3 are used as pivots for the second dimension.

2. Two stage FASTMAP-MDS approach

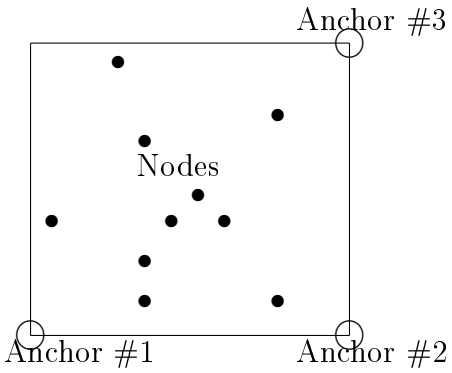

Fig. 3.1: Pivot node placement for using FASTMAP in sensor network localization

The resulting estimates of the previously described algorithm can be used as initialization for gradient descent. The steps that will be needed in this stage depend on the quality of the initialization and each step costs $O(N^2)$. The two-stage algorithm is shown in Table 3.1.

At this point, it is important to explain our notation.(x_i, y_i) denotes the estimated position of node i. For the studied case, the stress function that is used is a common one, described in (3.3).

$$stress^2 = \sum_{ij} (\widehat{d}_{ij} - d_{ij})^2 \tag{3.3}$$

with $\widehat{d}_{ij}$ be the Euclidean distance between nodes $X_i = (x_{i1}, x_{i2}, ..., x_{im})$ and $X_j = (x_{j1}, x_{j2}, ..., x_{jm})$

$$\widehat{d}_{i,j} = \sqrt{\sum_{k=1}^{m} x_{ik} - {x_{jk}}^2}. \tag{3.4}$$

From the above equations, the partial derivative of the stress function can be computed for every dimension of the studied space.

Input: **D**

1. Run Fastmap using as pivot the anchor nodes, which are placed on the three vertices of the square distribution area. Let X be the vector which contains all the estimated coordinates, which are returned by Fastmap.
2. Determine p, λ
3. For $i = 0$ to p

   begin

   - evaluate ∇ stress at the point X

   - X = X - $\lambda\nabla$ stress

   end
4. Output X

Tab. 3.1: The 2-D Hybrid Fastmap-MDS Algorithm

$$\frac{\partial stress}{\partial x_i} = \sum_{j \neq i} \frac{(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_{ij})(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{3.5}$$

3. Implementation - Results

During the implementation of the described algorithm we consider that the network has full connectivity (we have distance estimates for every pair of nodes). The distance estimates are assumed to contain an error which is proportional to the true distance between the nodes. Thus, the distance estimates are modeled to be

$$d_{ij} = p_{ij} + p_{ij} N(0, e_r), \tag{3.6}$$

where p_{ij} is the true distance between nodes i and j and e_r is the range error variance. Network nodes are considered to be uniformly distributed in a square with area equal to 1, i.e. the x and y coordinates of the sensor nodes are assumed uniformly distributed in $[0, 1]$. For aligning the estimated constellation, the technique

Algorithm	Complexity
Fastmap	$O(mN)$
Hybrid Fastmap-MDS	$O(pmN_2), p << N$

Tab. 3.2: Algorithm Complexities

described in section 3.1 is used. As an estimation performance metric we use the root mean squared error described in (3.7).

$$RMSE = \frac{\sum_{i=1}^{N} \sqrt{(x_{ri} - x_{ei})^2 + (y_{ri} - y_{ei})^2}}{N} \tag{3.7}$$

where x_{ei}, y_{ei} are the estimated coordinates, and x_{ri}, y_{ri} are the real ones. The complexities of Fastmap and hybrid Fastmap-MDS are shown in table 3.2.

In figures 3.2 and 3.3 we show the RMSE performance versus the range error variance for the Fastmap and hybrid Fastmap-MDS algorithms. For a constellation of 80 nodes and $\lambda = 0.01$, we observe that the hybrid algorithm outperforms FASTMAP in terms of accuracy. Moreover, this fact is also confirmed for 200 node groups and λ set now at 0.005. While comparing the two graphs we reach to the conclusion that the algorithms work good enough for both smaller and larger node groups, offering similar results.

On the other hand, when more noise is added to the distance measurements, the performance of the algorithms gets worse and inaccurate position estimates are provided (figure 3.4). Note that, for the noiseless case, both algorithms provide completely accurate position estimates. In all cases, the results were averaged over 100 Monte-Carlo simulations for each range error value.

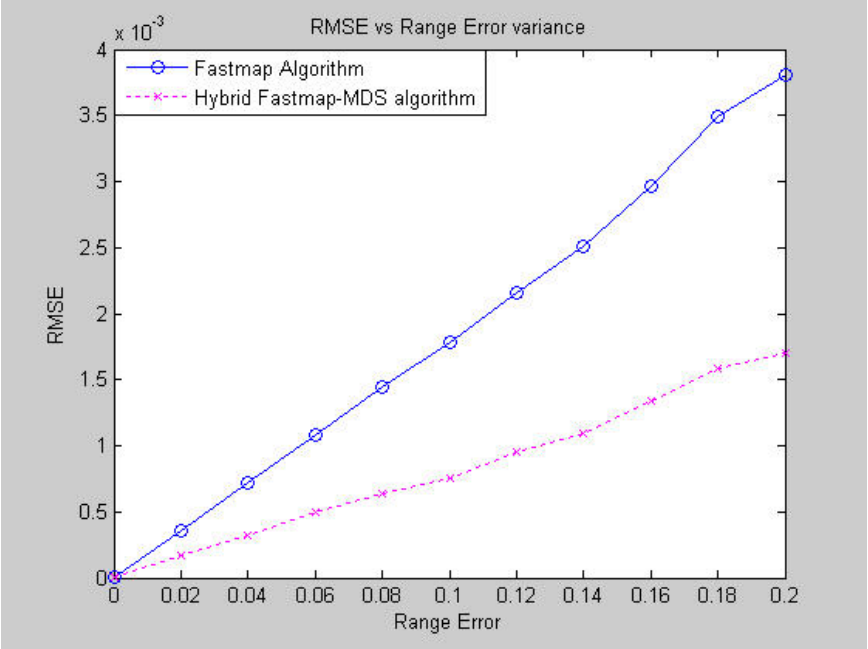

Fig. 3.2: Fastmap vs Fastmap-MDS Algorithm Performance for 80 Node Constellations

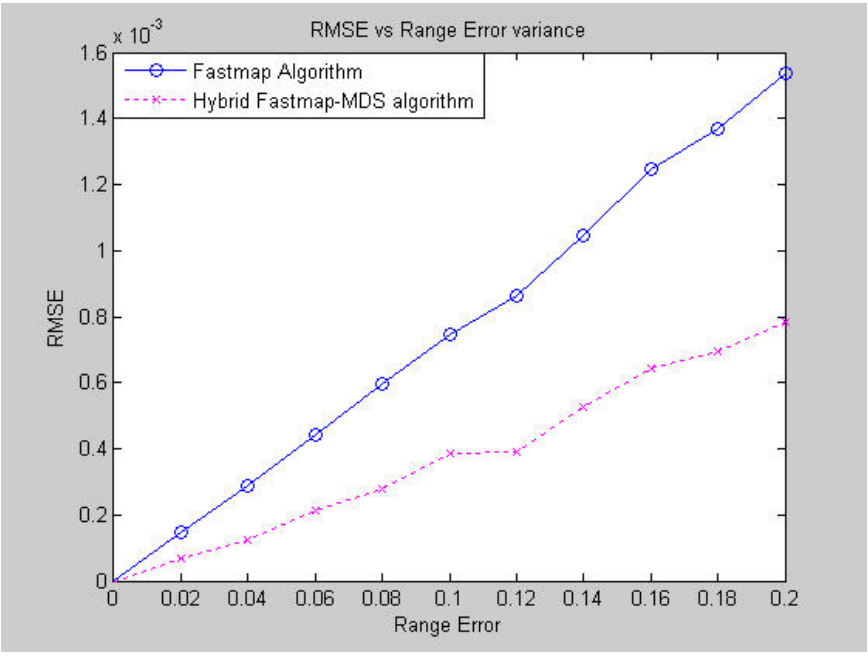

Fig. 3.3: Fastmap vs Fastmap-MDS Algorithm Performance for 200 Node Constellations

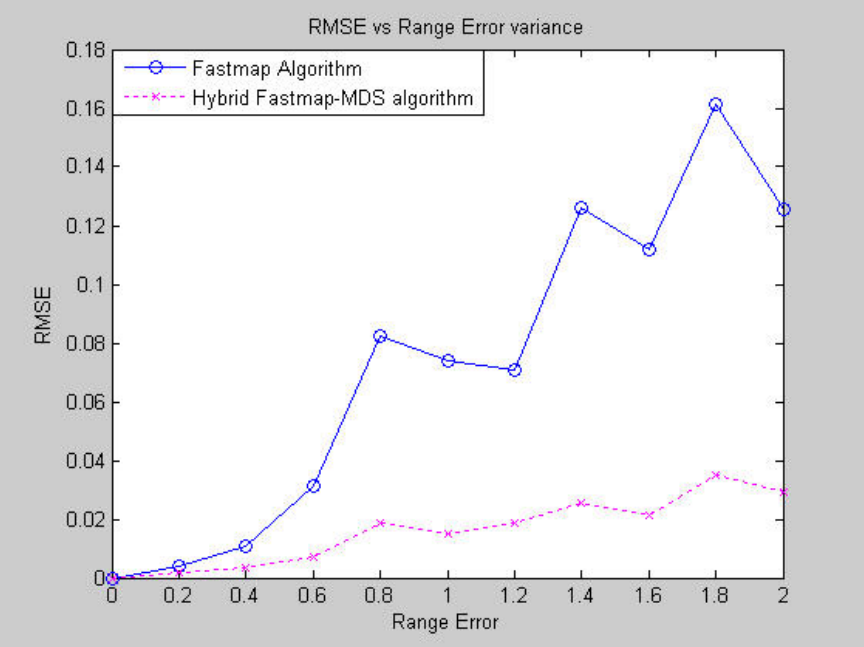

Fig. 3.4: Fastmap vs Fastmap-MDS Algorithm Performance for 80 Node Constellations - Different Range Error Scale

4. DISTRIBUTED WEIGHTED MULTIDIMENSIONAL SCALING

The goal of multidimensional scaling in the sensor localization problem is to find a map representation of a group of sensors, so that the distances between nodes fit the best possible way to a given set of measured pairwise dissimilarities[1]. These distances can be obtained by using RSS or TOA technics.

The proposed by Costa-Patwari-Hero algorithm refers to a distributed MDS implementation which falls in the successive refinement category of algorithms, and finds a minimum of a global cost function by minimizing local cost functions. The cost function that is used avoids the complicated step of merging local maps and a majorization algorithm ensures that each iteration decreases the global cost function.

The first step to sensor localization problem is to select neighborhoods for the range measurements. As it is most common in real life applications, the measurements are done in a noisy environment and this effects the neighbors-selection-process because some distances may be misestimated. In order to solve this problem, Costa-Patwari-Hero algorithm proposes a two-stage neighbor selection process that can be used to unbias location estimates even in high-noise environments. However, we are not going to cope with this matter profoundly in this work.

1. *Problem Statement*

A typical problem in sensor localization considers a network of $N = n + m$

[1] ie. distances between nodes

D	Dimensions of location estimates (D = 2 unless noted)
$N = n + m$	Total number of sensors
n	Sensors with imperfect or no a priori coordinate information
m	Sensors with perfect a priori coordinate knowledge (anchor nodes)
x_i	Actual coordinate vector of sensor i, $i = 1...n + m$
X	Actual coordinate matrix, $[x_1, ..., x_{n+m}]$
$d_{ij}, d_{ij}(X)$	Actual distance between sensors i and j in matrix X
$\delta_{ij}^{(t)}$	Range measured at time t between sensors i and j
$w_{ij}^{(t)}$	Weight given to the range measured at time t between sensors i and j
$\overline{\delta_{ij}^{(t)}}$	Weighted average measured range between sensors i and j
$\overline{w_{ij}^{(t)}}$	Weight given to the average measured range between sensors i and j
S	Global objective function to be minimized
S_i	Local objective function to be minimized at sensor $i = 1, ..., n$
$x_i^{(k)}$	Estimated coordinates of sensor i at iteration k
$X^{(k)}$	Estimated coordinate matrix at iteration k

Tab. 4.1: Symbols used in text and derivations

devices that live in a D-dimensional space (D < N). Let $\{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^D$ be the actual vector coordinates of sensors and let us define the matrix of coordinates $X = [\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x}_{n+1}, ..., \mathbf{x}_N]$. The first n sensors $(i = 1, ..., n)$ have either no, or some imperfect a priori coordinate knowledge and are called unknown-location nodes. Imperfect a priori knowledge about sensor $i \leq n$ is described by parameters r_i and $\overline{\mathbf{x}}_i$ where with accuracy r_i, $\mathbf{x}_i$ is believed to be around $\overline{\mathbf{x}}_i$. If no such knowledge is available then $r_i = 0$. The last m sensors $(i = n + 1, ..., N)$ have perfect a priori knowledge of their coordinates and are called anchor nodes.

The notation used in Costa-Patwari-Hero algorithm is summarized in Table 4.1.

Given the coordinates of the anchor nodes $\{\mathbf{x}_i\}_{i=n+1}^{N}$,imperfect a-priori knowledge $\{(r_i, \overline{\mathbf{x}}_i)\}_{i=1}^{n}$ and many pairwise range measurements, $\{\delta_{ij}^{(t)}\}$ taken over time $t = 1..K$, the localization problem describes the estimation of the coordinates $\{\mathbf{x}_i\}_{i=1}^{n}$.

2. *The DW-MDS Cost Function*

The estimation of sensor positions is done by minimizing the following global cost function (STRESS function [Cox and Cox 1994]):

$$S = 2 \sum_{1 \le i \le n} \sum_{i \le j \le n+m} \sum_{1 \le t \le K} w_{ij}^{(t)} (\delta_{ij}^{(t)} - d_{ij}(X))^2 + \sum_{1 \le i \le n} r_i \left\| \mathbf{x}_i - \overline{\mathbf{x}}_i \right\|^2 \tag{4.1}$$

Where the actual Euclidean distance is given $d_{ij}(X)$ by:

$$d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)} \tag{4.2}$$

It is assumed that for each pair of (i, j) up to K dissimilarity measurements are available. The weight $w_{ij}^{(t)} (t = 1, ..., K)$ quantifies the accuracy of measurement $\delta_{ij}^{(t)}$. If no measurement is available between i and j or its accuracy is zero then $w_{ij}^{(t)} = 0$. Generally $w_{ij}^{(t)} \ge 0$, $w_{ii}^{(t)} = 0$ and $w_{ij}^{(t)} = w_{ji}^{(t)}$.

As mentioned above the algorithm finds a minimum of the global cost function by minimizing local cost functions. Under this perspective S can be written as:

$$S = \sum_{i=1}^{n} S_i + c, \tag{4.3}$$

where one local cost function S_i is defined for each unknown node.

$$S_i = \sum_{j=1, j \ne i}^{n} \overline{w}_{ij} (\overline{\delta}_{ij} - d_{ij}(X))^2 + \sum_{j=n+1}^{n+m} 2\overline{w}_{ij} (\overline{\delta}_{ij} - d_{ij}(X))^2 + r_i \left\| \mathbf{x}_i - \overline{\mathbf{x}}_i \right\|^2, \tag{4.4}$$

and c is a constant independent of the nodes location. Since there are more than one and less than K range measurements between nodes i and j we summarize these measurements and the corresponding weights in a single quantity $\overline{w}_{ij} = \sum_{t=1}^{K} w_{ij}^{(t)}$

and $\overline{\delta}_{ij} = \sum_{t=1}^{K} w_{ij}^{(t)} \delta_{ij}^{(t)} / \overline{w}_{ij}$. As it is shown at (4.4), S_i depends only on the available measurements between i and the positions of the neighboring nodes, for which $\overline{w}_{ij} > 0$. Thus S_i can be effectively used as a local cost function.

3. *Minimizing the DW-MDS Cost Function*

Since no closed form expression exists for the minimum of the cost function S_i, the problem is solved by minimizing $S_i(\mathbf{x_i})$ iteratively, using quadratic majorizing functions (as in SMACOF - Scaling by MAjorizing a COmplicated Function) [14]. This process assures a pure decreasing sequence of STRESS values. Since it is not in the interest of this research, we do not present the whole minimization process, but only the final equations that define the nodes positions. The update for node's i position is given by (4.5),

$$\mathbf{x}_i^{(k+1)} = a_i(r_i \overline{\mathbf{x}}_i + \mathbf{X}^{(k)} \mathbf{b}_i^{(k)}), \tag{4.5}$$

where

$$a_i^{-1} = \sum_{j=1, j \neq i}^{n} \overline{w}_{ij} + \sum_{j=n+1}^{n+m} 2\overline{w}_{ij} + r_i, \tag{4.6}$$

and $\mathbf{b}_i^k = [b_1, ..., b_{n+m}]^T$ is a vector whose entries are given by the following formulas:

$$\begin{aligned} b_j &= \overline{w}_{ij}[1 - \overline{\delta}_{ij}/d_{ij}(X^{(k)})] && j \leq n, j \neq i \\ b_i &= \sum_{j=1, j \neq i}^{n} \overline{w}_{ij} \overline{\delta}_{ij}/d_{ij}(X^{(k)}) + \sum_{j=n+1}^{n+m} 2\overline{w}_{ij} \overline{\delta}_{ij}/d_{ij}(X^{(k)}) && \\ b_j &= 2\overline{w}_{ij}[1 - \overline{\delta}_{ij}/d_{ij}(X^{(k)})] && j > n \end{aligned} \tag{4.7}$$

For nodes j not in the neighborhood of node i, the weights $w_{ij}^{(t)}$ are zero. Thus, the corresponding entries of vector $\mathbf{b}$ will be zero, and the update for x_i will depend only on its neighborhood, not in the whole constellation.

Input: $\{\delta_{ij}^{(t)}\}$, $\{w_{ij}^{(t)}\}$, m, $\{r_i\}$, $\{\overline{x_i}\}$, ϵ, initial condition $X^{(0)}$

Initialize: k=0, $S^{(0)}$, compute a_i from equation 4.6

repeat

$k \leftarrow k+1$

for $i = 1$ to n

compute $b^{(k-1)}$ from equation 4.7

$x_i^{(k)} = a_i(r_i\overline{x_i} + X^{(k-1)}b_i^{(k-1)})$

compute $S_i^{(k)}$

$S^{(k)} \leftarrow S^{(k)} - S_i^{(k-1)} + S_i^{(k)}$

communicate $x_i^{(k)}$ to neighbors of node i (i.e, nodes for which $w_{ij} > 0$)

communicate $S^{(k)}$ to node $i + 1(\mathrm{mod}(n))$

end for

until $S^{(k-1)} - S^{(k)} < \epsilon$

Tab. 4.2: The DW-MDS Algorithm

4. *Algorithm*

The distributed weighed-multidimensional algorithm is shown at Table 4.2.

The computational complexity of the algorithm is $O(nL)$ where L is the number of iterations required until the algorithm converges. Although the majorization approach used by the authors guarantees a non-increasing sequence of STRESS values, the cost function may converge to a local minimum like any gradient descent method. As a result, the algorithm may not provide satisfactory results in some cases.

In order to improve the performance of the algorithm, a different one than the random initialization can be used. Costa-Patwari and Hero suggest the use of the algorithms proposed in [Savarese et al. 2001 [5]] or [Capkun et al. 2001 [7]].

However, our research showed that these algorithms cannot be used for this purpose since they may provide position estimates that are not real, but complex[Sections 4.1, 4.2].

5. *Implementation - Results*

The measurement that is used in order to authenticate the performance of the studied algorithms is the Root Mean Square Error (RMSE), which is defined to be $RMSE = \frac{\sum_{N}^{i=1}\sqrt{(x_{ri}-x_{ei})^2+(y_{ri}-y_{ei})^2}}{N}$. X_{ri} and Y_{ri} are the real coordinates of each node and in correspondence X_{ei} and Y_{ei} are the estimated ones. In figure 4.1 the algorithm's performance for 80 nodes constellations and for various noise environments is presented. The parameter that arranges the quantity of noise is the *range_error* variance and the parameter ϵ is set to 0.01. In all cases, the initialization of the algorithm was matrices with pairwise distances and a random intial estimation of the coordinates of the nodes. No former knowledge on nodes position is used except than the one of the anchor nodes. In addition, we consider full-connectivity networks where every pair of nodes can establish communication. The results were averaged over 100 Monte-Carlo simulations for each range error value.

At this point, it is of use to point out that the algorithm cannot work efficiently without the existence of at least $X+1$ anchor nodes where X defines the dimension in which we work. If no sufficient anchor nodes exist, then the estimated map may not be aligned[2] to the physical constellation even the relative positions between nodes are estimated right.

For noiseless environments, the algorithm offers $RMSE \approx 10^{-3}$ and as noise is added in the experiment, the performance becomes worse, as it should do. For range

[2] For more information in the alignment field check chapter 5

errors between 0 and 0.2, the algorithm's performance does not change dramatically and the RMSE balances around the value of $3.3 * 10^{-3}$. While more noise is added to the system and the range error variance increases, the algorithm's performance becomes worse, and this reflects in the rising swing of the RMSE. Thus, one can conclude to the fact that DW-MDS is rather insensitive to noise, especially for high SNR regions.

In order to give a more complete and actual view on what an RMSE value means in terms of nodes positions, it is useful to take a look at figures 4.2 and 4.3. Two 80-nodes complexes are presented in how they really are and how the algorithm estimates them in various noise environments.

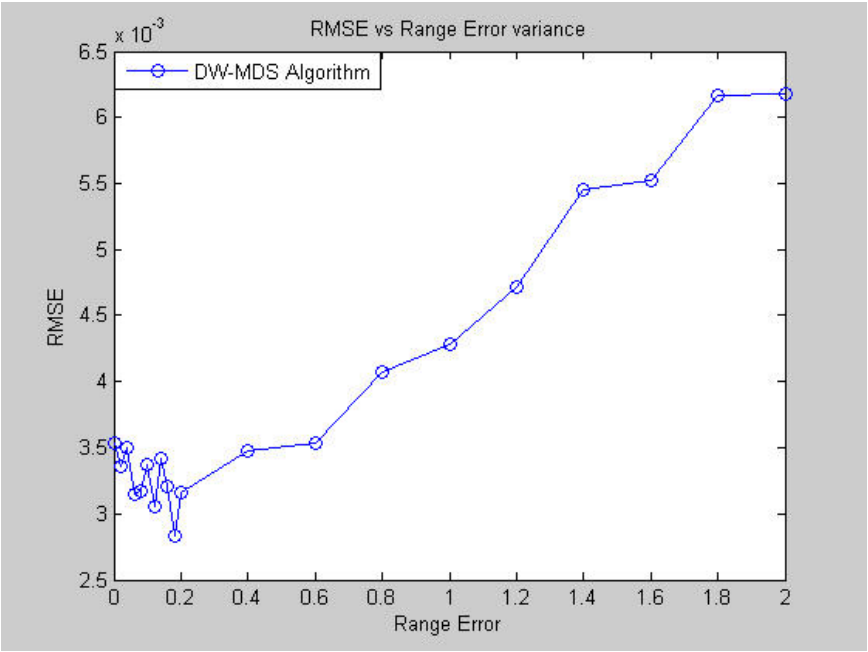

Fig. 4.1: DW-MDS Performance for 80 Node Constellations

4.1 Proposed DW-MDS initialization: Savarese's ABC algorithm

- Introduction

Through this research, Savareze, Rabaey and Beutel present the *Assumption Based*

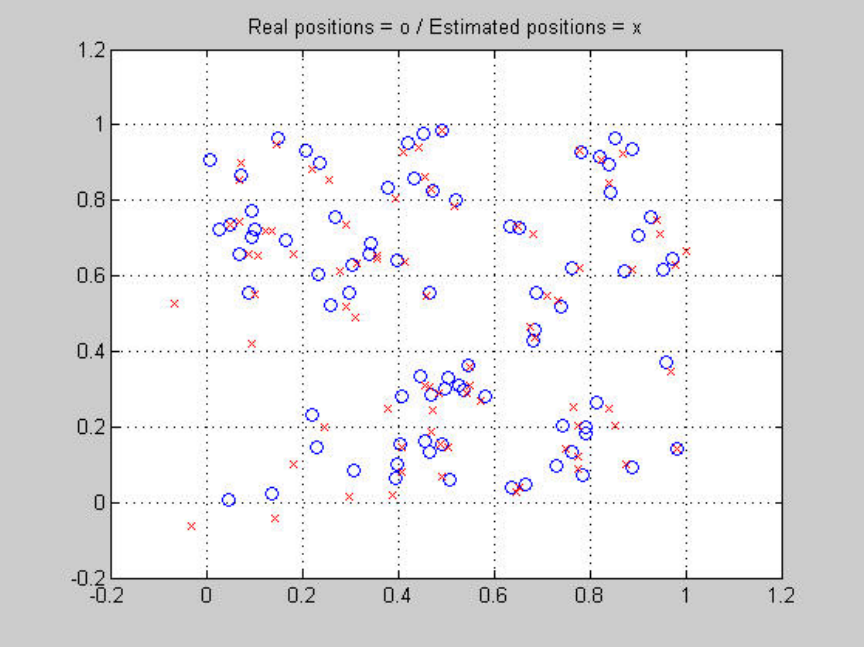

Fig. 4.2: DW-MDS Performance for Range Error=0 and RMSE=0.0004

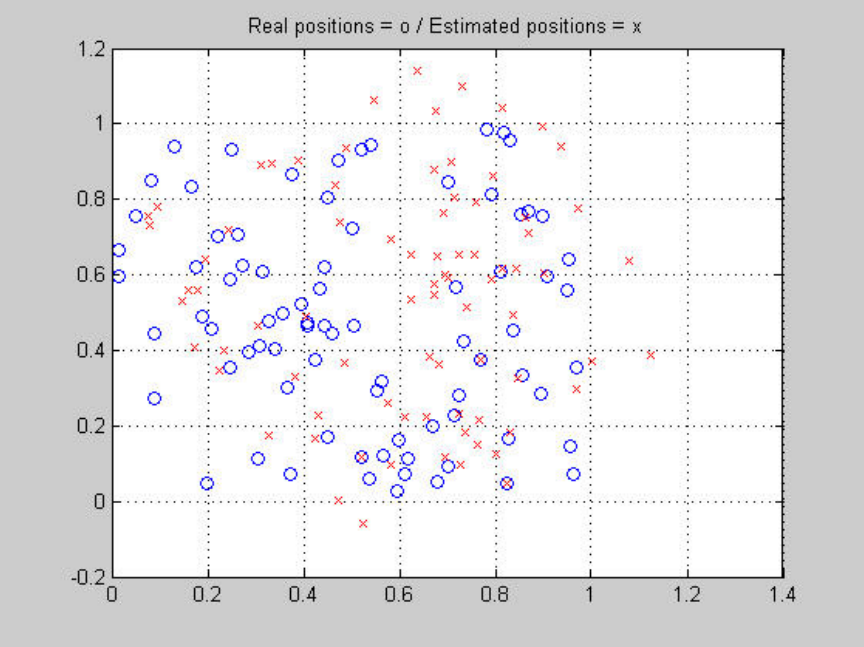

Fig. 4.3: DW-MDS Performance for Range Error=0.2 and RMSE=0.006

Coordinates (ABC) algorithm that tries to solve the following scenario: One node with a known position, receives range measurements from a large number of neighboring nodes with unknown positions and this information must be used in order to solve the localization problem. The ABC algorithm determines the locations of unknown nodes one at a time, in the order they establish communication, making assumptions where necessary. Under this general rule, it is useful to check on how it works for the first node n_0.

- ABC Algorithm

  The algorithm begins with the assumption that node n_0 is located at $(0;0;0)$. The first node to establish communication with n_0, n_1, is assumed to be located at $(r_{01};0;0)$, where r_{01} is the geometrical distance between n_0 and n_1. The location of the next node, n_2, can then be found, given two assumptions: the argument of the square root involved in finding y_2(4.8) is assumed to be a positive number, and z_2 is assumed to be 0. By x_i, y_i and z_i we refer to the corresponding x,y and z axis coordinate of node n_i. At this point, we must remark that in our research a 2-dimensional space is used, so the second assumption is not of concern. In addition, in noisy environment the first assumption cannot be assured. However, it must be mentioned in order to show the general concept of the algorithm. The coordinates of n_2 can easily be found by using the Pythagorean theorem as shown in (4.8).

$$x_2 = \frac{r_{01}^2 + r_{02}^2 - r_{12}^2}{2r_{01}} \qquad y_2 = \sqrt{r_{02}^2 - x_2^2}. \tag{4.8}$$

  The next node, n_3, is handled like n_2, except that only one assumption is made: the square root involved in finding z_3(4.9) is positive. Still in noisy environment this assumption cannot be assured. The new equations describing n_3 are shown in (4.9).

$$x_3 = \frac{r_{01}^2 + r_{03}^2 - r_{13}^2}{2r_{01}}$$

$$y_3 = \frac{r_{03}^2 - r_{23}^2 + x_2^2 + y_2^2 - 2x_2x_3}{2y_2} \tag{4.9}$$

$$z_3 = \sqrt{r_{03}^2 - x_3^2 - y_3^2}$$

From this point, the equations needed to define the coordinates of each new node cannot be easily determined since the node needs to interact and adjust with four others. Instead, the standard algorithm can be applied in order to build a local coordinates system.

- Implementation - Results

  After implementing in MATLAB script the above scenario, the results of the simulations showed that in noisy environments the ABC algorithm may give complex position estimates and, of course, they cannot be used as an initial estimation for the network topology. The problem originates in the assumptions made earlier, concerning the estimation of y_2 and z_3, where the unknown quantity under the square root is assumed to be positive. Thus, ABC algorithm cannot be used for the initialization of the DW-MDS algorithm.

4.2 Proposed DW-MDS initialization: Capkun's SPA algorithm

- Introduction

  The proposed from Capkun, Hamdi and Hubaux [7], algorithm is a GPS-free method that uses the distances between nodes in order to build a relative coordinate system. It is a distributed algorithm that enables the nodes to find their positions within the network area using only their local information, and is referred as the Self Positioning Algorithm (SPA). It uses range measurements between the nodes to build a Network Coordinate System. The Time of Arrival (TOA) method is used

to estimate the range between two mobile devices. Examples of application where this algorithm can be used include Location Aided Routing [15] and Geodesic Packet Forwarding [16] both in ad hoc and sensor networks.

In our research we tried to apply the mentioned algorithm in order to take an initial estimation of the topology of the examined network and use it in order to initialize the DW-MDS algorithm. However, the results were not so encouraging. It is of importance to explain first, how the SPA algorithm works in its initial steps that are enough to understand why it cannot be used for DW-MDS initialization.

- Building Local Coordinate System using SPA

  First of all, there are some assumptions which are made concerning the system model.

  1. The studied network is a network of mobile and wireless devices without any specific structure.
  2. All the devices (nodes) have the same technical characteristics.
  3. All the wireless links between the nodes are bidirectional.
  4. The nodes use omnidirectional antennas.
  5. The maximum speed of the movement of nodes is limited to 20m/s.

  In this section it is shown how every node builds its Local Coordinate System. In a few words, the node becomes the center of its own coordinate system with the position $(0, 0)$ and the positions of its neighbors are computed accordingly.

  If a node j can communicate directly (in one hop) with node i, j is called a one-hop neighbor of i. In a network of N nodes, define K_i to be the set of one-hop nodes

for $\forall i \in N$, and D_i the set of distances between i and each node $j \in K_i$. For every node i the following procedure is performed:

- detect all one-hop neighbors (K_i),
- measure the distances to one-hop neighbors (D_i),
- send the K_i and D_i to all one-hop neighbors.

Thus every node knows its one and two-hop neighbors, all the distances between its one-hop and some of the distances between its two-hop neighbors. A number of distances cannot be obtained due to power limitations or obstacles between the nodes.

As a plane is defined by three points not in the same line, in order to define node's i coordinate system, we need two more nodes. Let these nodes be $p \in K_i$ and $q \in K_i$, their distance be d_{pq} and nodes i, p and q don't lie in the same line. In addition, suppose that node p lies on the x-axis and node q has a positive q_y coordinate. Using this information, we can uniquely define i's Local Coordinate System, and the coordinates of nodes i, p, q are the ones shown in (4.10).

$$\begin{aligned} i_x = 0, i_y = 0 \\ p_x = d_{ip}, p_y = 0 \\ q_x = d_{iq}\cos\gamma, q_y = d_{iq}\sin\gamma \end{aligned} \tag{4.10}$$

where γ is the angle $\angle(p, i, q)$ and is obtained by using the cosines rule for triangles(4.11).

$$\gamma = \arccos\frac{d_{iq}^2 + d_{ip}^2 - d_{pq}^2}{2d_{ip}d_{iq}} \tag{4.11}$$

The positions of the nodes $j \in K_i, j \neq p, q$, for which the distances d_{ij}, d_{qj} and d_{pj} are known, are computed by triangulisation. Therefore, their coordinates can be found using (4.12).

$$j_x = d_{ij} \cos a_j$$

$$if \quad \beta_{ij} = |\alpha_j - \gamma| \Rightarrow j_y = d_{ij} \sin \alpha_j \tag{4.12}$$

$$\text{else} \Rightarrow j_y = -d_{ij} \sin \alpha_j$$

where α_j is the angle $\angle(p, i, j)$ and β_j is the angle $\angle(j, i, q)$.

By using the cosine rule the values of α_j and β_j are obtained from (4.13) and (4.14).

$$\alpha_j = \arccos \frac{d_{ij}^2 + d_{ip}^2 - d_{pj}^2}{2 d_{ij} d_{ip}} \tag{4.13}$$

$$\beta_j = \arccos \frac{d_{iq}^2 + d_{ij}^2 - d_{qj}^2}{2 d_{ij} d_{iq}} \tag{4.14}$$

Figure 4.4 shows the example of the above computation for node j. For every other node $m \in K_i$, its position can be computed in the same way, only by knowing the positions of node i, two other nodes and their pairwise distances.

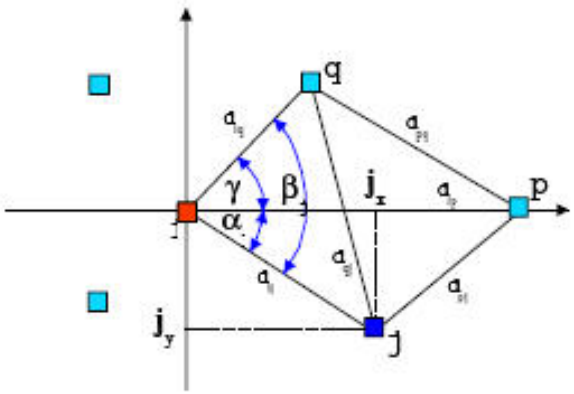

Fig. 4.4: Example illustrating how to obtain the position of node j in the coordinate system of node i

- Implementation - Results

  After implementing the algorithm for the local coordinate system in Matlab script,

we noticed the existence of complex values in the estimation of the nodes' coordinates. This origins from the fact that we compute the inverse cosine of an unknown quantity -suppose x-, for which we don't have any restrictions or haven't made any assumptions. We know from trigonometry rules that if $|x| > 1$ then $\arccos x$ gives a complex solution. This value is furtherly used in the computation of $\cos(\arccos x)$ and $\sin(\arccos x)$ which also gives a complex number since the argument is not real. As a result, the coordinate estimates which are computed cannot be used as initialization for DW-MDS, since we need real data.

4.3 Proposed DW-MDS initialization: FASTMAP algorithm

- Introduction

  As described in chapter 3, the Fastmap algorithm is an algebraic method that can be used in order to produce an initial estimate of the nodes location. We will not currently explain the function of the algorithm but present the results of the combination of DW-MDS algorithm with Fastmap initialization.

- Implementation - Results

  The input of FASTMAP is a matrix of pairwise distances. The distances are affected by noise proportional to the actual distance of the nodes. We assume that the nodes are spread in a square and that three anchors are placed at three edges of this square. The output is used as input for the DW-MDS.

  The simulations have shown faster convergence for the hybrid algorithm than the traditional DW-MDS, and better performance in terms of accuracy. The faster convergence is justified by the fact that the new initialization of Costa's algorithm is not random anymore, but an algebraic estimation of the final map. So the algorithm needs fewer iterations in order to converge. The same justification also

counts for the accuracy part. In figures 4.5, 4.6 and 4.7 the original DW-MDS algorithm's performance is compared to the DW-MDS with Fastmap initialization. The second offers better performance especially for low range error values. Due to the fact that we actually use two algorithms for the localization process, it is natural that the results are better and more close to reality. On the other hand, in high-noise environments, the Fastmap algorithm offers rather poor localization estimates which affect the Fastmap-DW-MDS overall performance.

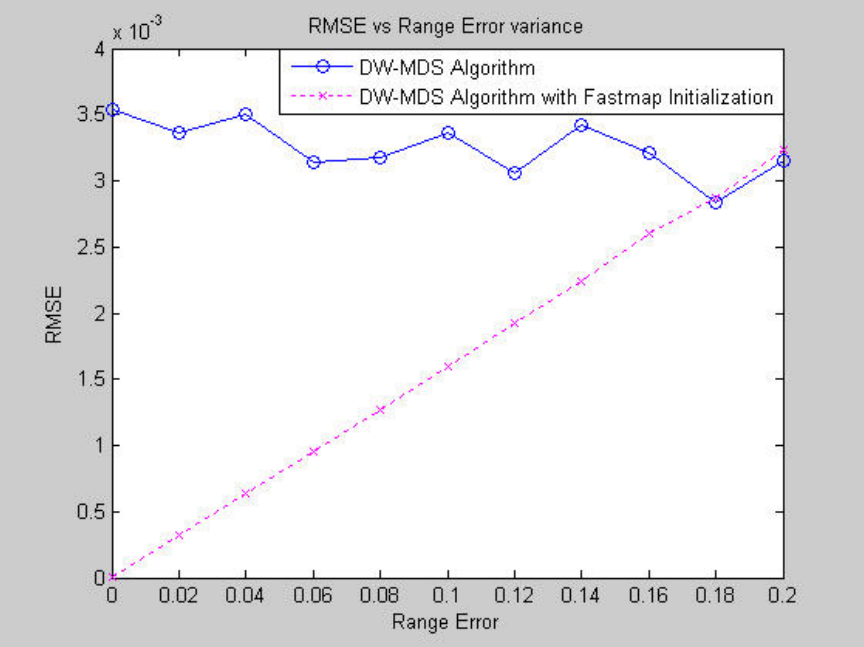

Fig. 4.5: DW-MDS vs DW-MDS with Fastmap initialization Performance for 80 Nodes Constellations

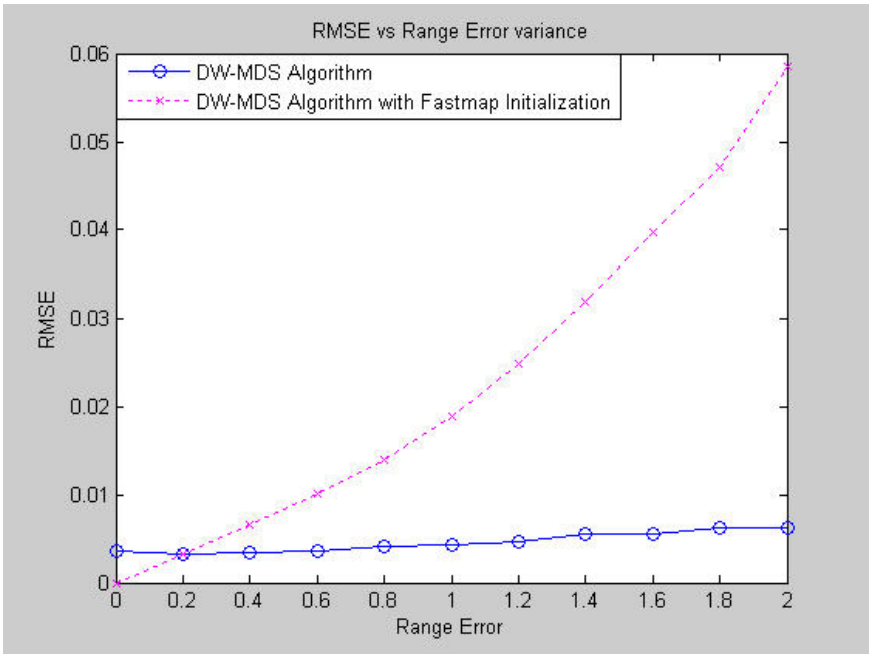

Fig. 4.6: DW-MDS vs DW-MDS with Fastmap initialization Performance for 80 Nodes Constellations - Different Rance Error Scale

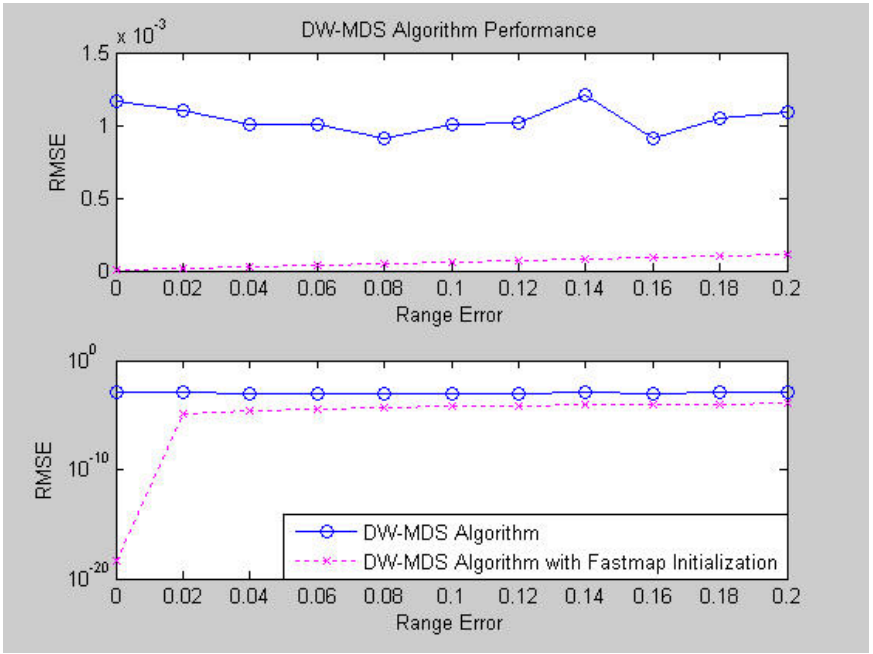

Fig. 4.7: DW-MDS vs DW-MDS with Fastmap initialization Performance for 200 Nodes Constellations

5. PROPOSED MODIFIED DW-MDS ALGORITHM

1. Introduction

Many of the available algorithms that solve the node localization problem require the existence of some anchor nodes (in most of the cases at least three) for whom we have almost perfect a-priori knowledge. They are essential in order to estimate the constellation.

As we saw in chapter 3, the Fastmap algorithm needs at least one pair of pivots for every dimension of the working space, in order to project the rest of the nodes on the formed line. Unfortunately, Fastmap is very sensitive to coordinate alignment because the estimated position of every node is only based on distances to the chosen pivots and thus there is no averaging. In order to minimize this problem the pivots are chosen to be placed on the outer edges of the network, forming an orthogonal triangle.

On the other hand, DW-MDS needs the anchor nodes in order to align the estimated nodes coordinates to their physical positions, as the algorithm does not use any special alignment procedure. The positions of the anchors are not re-estimated -since they are already known- and the whole map is built around them.

2. Challenges in the precise estimate of the location of anchor nodes

We should now take into consideration the words of the ancient Greek philosopher Socrates who said that : "'Εν οἶδα, ὅτι ουδέν οἶδα"'. In free translation this means

that: "'One thing I know is that I know nothing"'.

Thus taking for granted that we have perfect knowledge on something is not always true. In our case, it is quite difficult and sometimes not possible to have perfect a-priori knowledge for a node location. In most of the proposed algorithms, the location of anchor nodes is found through GPS, something which is not always feasible [17].

- GPS cannot be used in indoors networks because its precision is very low, as its jammed from many obstacles.
- GPS offers an accuracy of the order of 25m in the horizontal plane and 43m in the vertical plane for custom civilian applications. Thus, GPS measurements cannot be used in order to achieve reliable node position estimates in networks where the distances between nodes are of the same or lower order of the offered accuracy.
- GPS performance is affected by many factors, such as the changing in atmospheric conditions as the GPS signals pass through the Earth's atmosphere and ionosphere. In addition, GPS signals can be affected by multipath issues, where the radio signals reflect on the surrounding terrain, buildings, hard ground, etc. These delayed signals can cause inaccuracy. Furthermore, jamming of any radio navigation system, including satellite based navigation is possible.

Under these conditions, the estimation of the position of a sensor network which is located in a canyon, or in a battlefield could be very tricky if the use of GPS is demanded.

On the contrary, in cases that we can use the GPS effectively, or some other method of acquiring the anchor nodes positions with an accepted accuracy, there

are still problems that may appear.

- The possible error in the calculation of the anchor node's position, is transfered in the estimation of the rest of the constellation and if no separate alignment procedure is used, then the output map may be well estimated in terms of pairwise distances, but actually improperly aligned.
- In order to achieve greater accuracy, the use of expensive or complicated machines is possible. However, for the sake of cost and complexity reduction of every node, we chose not to use them in the system.

From the hindrances described above, we reached to the conclusion that it is better to use the knowledge for the anchors as much as we can, but it would be wiser to use them carefully in order to align the final map to the physical positions. Aligning procedure is very important for acquiring the best performance of the localization algorithm used. Thus, constellation estimation and map alignment should be treated separately if possible. The blind use of the anchor nodes positions knowledge for aligning the map, could be treacherous.

3. Modified DW-MDS algorithm

In the Modified DW-MDS algorithm, node position estimation is treated separately from the final coordinates alignment. The algorithm still uses its knowledge of the anchor nodes position, and calculates the position of the rest of the nodes according to them. However, the algorithm now re-estimates the position of the anchor nodes as if they were unknown. The result of the estimation of the anchor nodes can provide a hint about how well the algorithm worked. Now that DW-MDS "loses" its anchors, the output map needs to be aligned to the physical positions. We adopt the aligning algorithm, proposed by Ji and Zha [9] which is presented

Input: $\{\delta_{ij}^{(t)}\}$, $\{w_{ij}^{(t)}\}$, m, $\{r_i\}$, $\{\overline{x_i}\}$, ϵ, initial condition $X^{(0)}$

Initialize: k=0, $S^{(0)}$, compute a_i from equation 4.6

repeat

$k \leftarrow k+1$

for $i = 1$ to **N**

compute $b^{(k-1)}$ from equation 4.7

$x_i^{(k)} = a_i(r_i\overline{x_i} + X^{(k-1)}b_i^{(k-1)})$

compute $S_i^{(k)}$

$S^{(k)} \leftarrow S^{(k)} - S_i^{(k-1)} + S_i^{(k)}$

communicate $x_i^{(k)}$ to neighbors of node i (i.e, nodes for which $w_{ij} > 0$)

communicate $S^{(k)}$ to node $i + 1(\mathrm{mod}(n))$

end for

until $S^{(k-1)} - S^{(k)} < \epsilon$

Run Alignment Algorithm described in section 5.4

Tab. 5.1: The modified DW-MDS Algorithm

next. This method, needs the anchor nodes position to be re-estimated in order to align them first, and then align the whole map according to the information taken from the anchors alignment.

The new algorithm is very similar to DW-MDS and its differences are pointed out in table 5.1.

4. Alignment algorithm

For a group of nodes, at least three anchors are needed in order to align the estimated positions to the physical ones, in a 2-D case. The alignment procedure includes three processes. Shift, rotation, and reflection of coordinates.

Assume that, $R = [r_{ij}]_{2*n} = (R_1, R_2, ..., R_n)$ denotes the relative positions of a set of n nodes in a 2-dimensional space and that $T = [t_{ij}]_{2*n} = (T_1, T_2, ..., T_n)$ denotes the true positions of the same nodes. Also, nodes $1, 2, 3$ are the anchors.

A vector R_i may be shifted to $R_i^{(1)}$ by $R_i^{(1)} = R_i + X$. It may be rotated counter-clockwise through an angle α to $R_i^{(2)} = Q_1 R_i$ where Q_1 is,

$$Q_1 = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{5.1}$$

It may also be reflected across a line

$$S = \begin{bmatrix} \cos\beta/2 \\ \sin\beta/2 \end{bmatrix} \tag{5.2}$$

to $R_i^{(3)} = Q_2 R_i$, where

$$Q_2 = \begin{bmatrix} \cos\beta & \sin\beta \\ \sin\beta & -\cos\beta \end{bmatrix} \tag{5.3}$$

Before alignment, we only know R and at least three anchor nodes physical positions T_1, T_2, T_3. Based on them, it is possible to compute $T_4, T_5, ..., T_n$, by using the above described rules. First we have,

$$(T_1 - T_1, T_2 - T_1, T_3 - T_1) = Q_1 Q_2 (R_1 - R_1, R_2 - R_1, R_3 - R_1) \tag{5.4}$$

and let's define Q to be the alignment factor equal to $Q = Q_1 Q_2$. Then $(T_4, T_5, ..., T_n)$ can be calculated with

$$(T_4 - T_1, T_5 - T_1, ..., T_n - T_1) = Q(R_4 - R_1, R_5 - R_1, ..., R_n - R_1) \tag{5.5}$$

and finally

$$(T_4, T_5, ..., T_n) = Q(R_4 - R_1, R_5 - R_1, ..., R_n - R_1) + (T_1, T_1, ..., T_1) \tag{5.6}$$

By the process described above, we achieve the alignment of the estimated map to the physical positions.

5. Implementation - Results

Through the simulation of the Modified DW-MDS algorithm, one observes that the results that it offers are quite inviting in terms of accuracy in estimating the nodes locations. Figures 5.1, 5.2 and 5.3 present the performance of Modified DW-MDS for 80 and 200 nodes constellations and for various noise environments, which are described by the range error variance. In all cases, the results were averaged over 100 Monte-Carlo simulations for each range error value.

More specifically, for the 80 node map, the algorithm offers almost perfect node position estimation for a noiseless case ($RMSE \approx 10^{-5}$), and then the RMSE increases non-linearly until the level of 10^{-2}. The peaks that may appear in some RMSE measurements are justified by the fact that the algorithm is based on the original DW-MDS, whose majorization approach may not converge to the global minimum of the cost function but in a local one instead. Thus, this non-convergence can give really bad results in the map estimation which effect on the overall experiment. Furthermore, another important notice comes from the study of figures 5.1 and 5.2. In the 80 nodes case the mean square error varies in the range between $[10^{-5}10^{-3}]$, though in the 200 nodes case in the $[10^{-6}10^{-4}]$ range. In other words, the algorithm offers better accuracy in larger scale networks where more nodes/distance measurements are included, and one node's position can be better averaged.

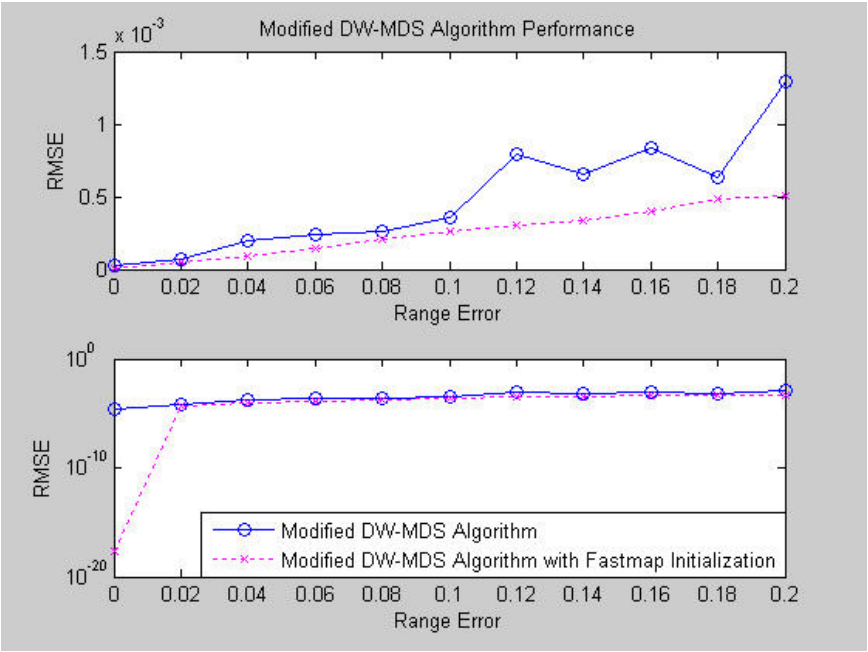

Fig. 5.1: Modified DW-MDS Algorithm Performance for 80 node constellations

In order to have a more accurate view on the algorithms performance it is worthwhile seeing graphs 5.4, 5.5 and 5.6 which show the real and estimated nodes positions on the same surface. RMSE's in the range of 10^{-5} imply great position estimation though RMSE's around 10^{-3} give rather inaccurate results. For RMSE's in the area of 10^{-4} the result map is quite acceptable and close to reality.

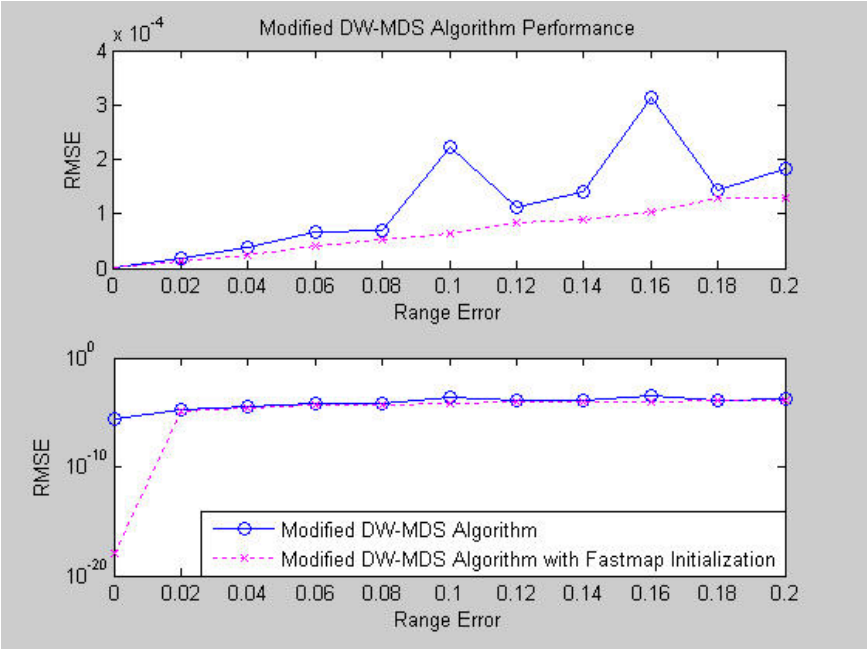

Fig. 5.2: Modified DW-MDS Algorithm Performance for 200 node constellations

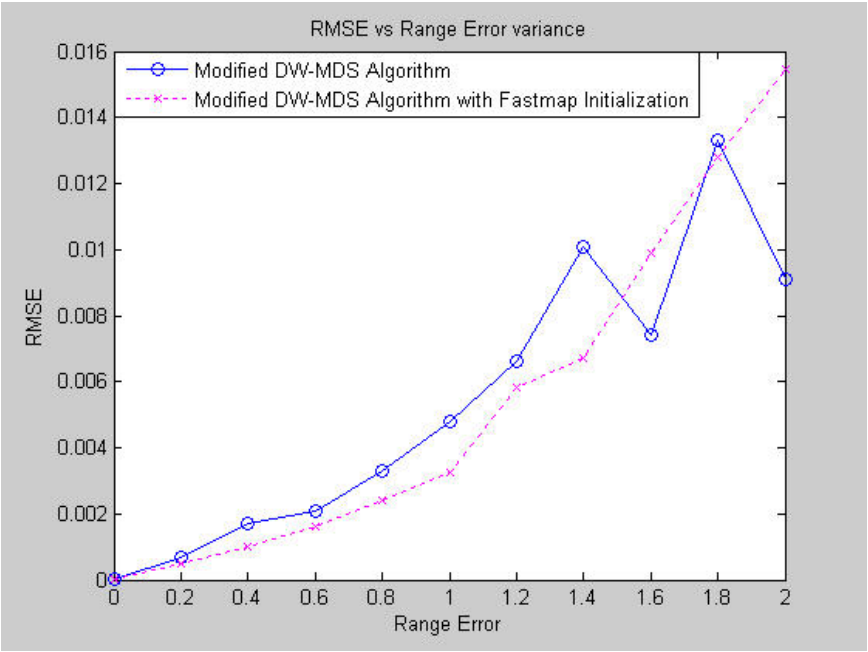

Fig. 5.3: Modified DW-MDS Algorithm Performance for 80 node constellations - different range error scale

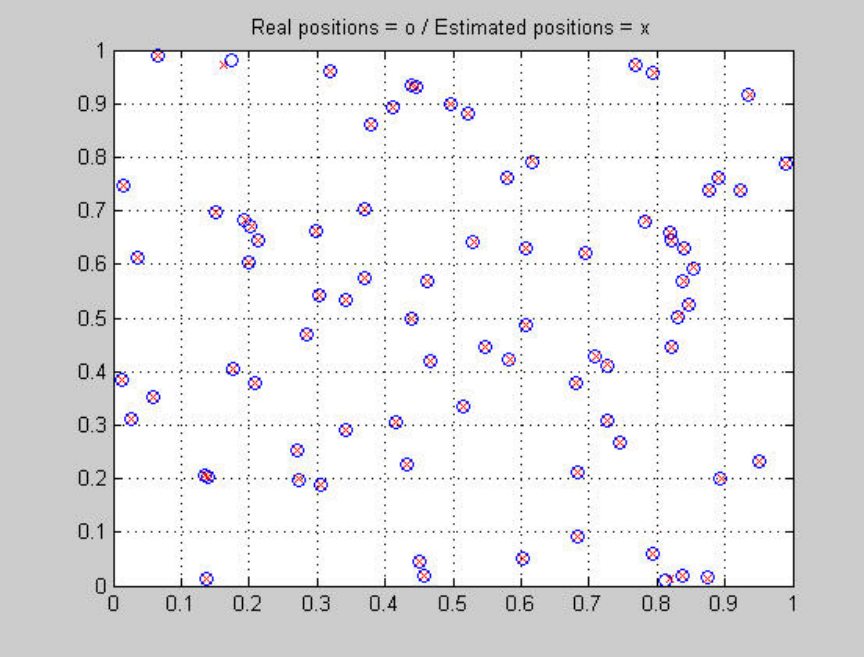

Fig. 5.4: Real vs Estimated positions for Range Error=0 and RMSE=0.0000085

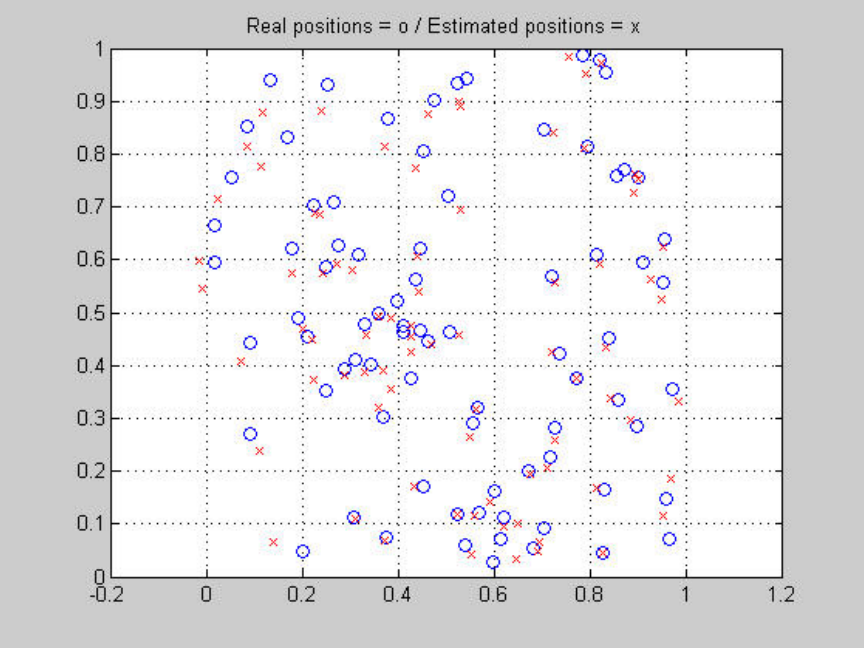

Fig. 5.5: Real vs Estimated positions for Range Error=0.2 and RMSE=0.00035

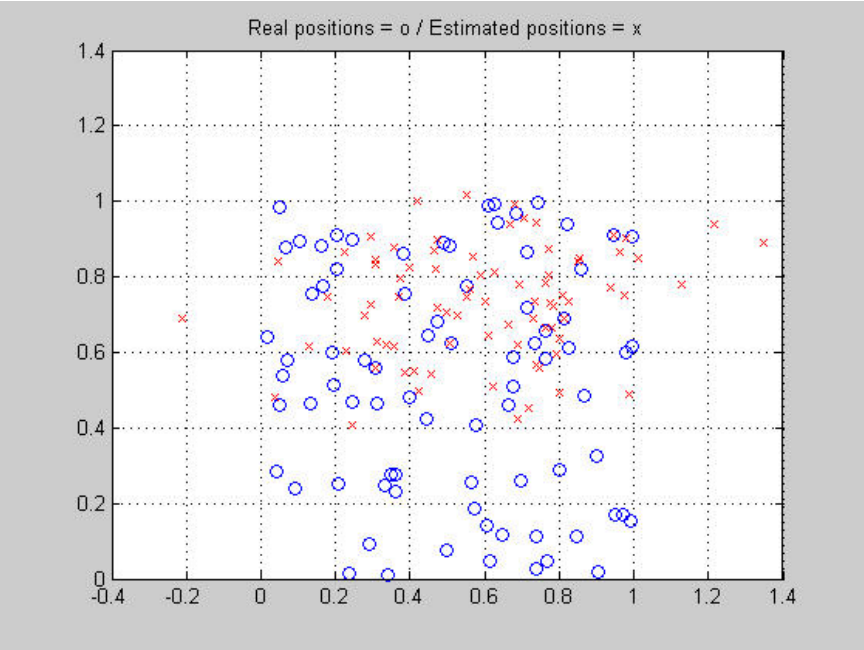

Fig. 5.6: Real vs Estimated positions for Range Error=2 and RMSE=0.0086

6. CONCLUSIONS

In the previous research, we explored various solutions in order to solve the problem of localization in wireless networks. The use of the basic Multidimensional Scaling algorithm as well as its variants is dominant in the sector and many proposed solutions depend on them.

Distributed Weighted Multidimensional Scaling algorithm is one of the classical MDS variations. We analyzed its functionality and checked the results that it offers in terms of accuracy and complexity. At a second stage, we tested methods proposed by the authors of the algorithm in order to change its initialization and achieve a better performance. Both Savarese's ABC algorithm [Savarese et al. 2001 [5]] and Capkun's SPA algorithm [Capkun et al. 2001 [7]], fail to initialize DW-MDS, as they may offer complex position estimates, while DW-MDS needs real ones. On the other hand, Fastmap algorithm can be successfully used in their place in order to acquire a first estimate of the constellation, and give it as input to the DW-MDS. The results of the DW-MDS with Fastmap initialization, are better as far as the accuracy and time needed to converge is concerned, compared to the original DW-MDS.

Furthermore, we studied and tested the proposed by Latsoudas, Sidiropoulos [3] FASTMAP-MDS algorithm. Since Fastmap is an algebraic method, it is quite sensitive to noise and in low SNR environments it does not work as efficiently as in high SNR cases.

Finally, we proposed a new version of the DW-MDS algorithm. In this new case, the position of the anchor nodes is re-estimated. However, now, in order to align the final

position estimates, to the physical positions of the nodes, a different method for aligning is used. We applied the method described by Ji and Zha [9].

The setup of the experiments included 80 or 200 nodes constellations, various noise environments and full-connectivity networks. The initialization of the algorithms was either matrices with pairwise distances and a random intial estimation of the coordinates of the nodes, or an "early version" of the constellation, estimated using Fastmap. For DW-MDS and Modified DW-MDS, the ϵ threshold was set at 0.01. The results were averaged over 100 Monte-Carlo simulations for each range error value. By taking a look in figures 6.1, 6.2 and 6.3 we remark the significantly better performance of Modified DW-MDS algorithm, in comparison to the original DW-MDS and Fastmap-MDS algorithms, from the viewpoint of RMSE. On the other hand, for 200 node constellations Fastmap-MDS offers better results than DW-MDS, but worse than Modified DW-MDS. However, when Fastmap initialization is used for DW-MDS algorithm, its performance is crucially improved, and outmatches Fastmap-MDS.

If one compares the present results to the ones extracted from Latsoudas-Sidiropoulos [3] research, may notice a difference in the performance of DW-MDS algorithm. This is due to the use of a lower threshold which improves the offered RMSE, in expense of the time needed for the algorithm to converge.

As future work, one could try to apply better technics for initializing DW-MDS in order to achieve better performance, both on RMSE and time of convergence aspects. In addition, more sophisticated methods for aligning could be developed.

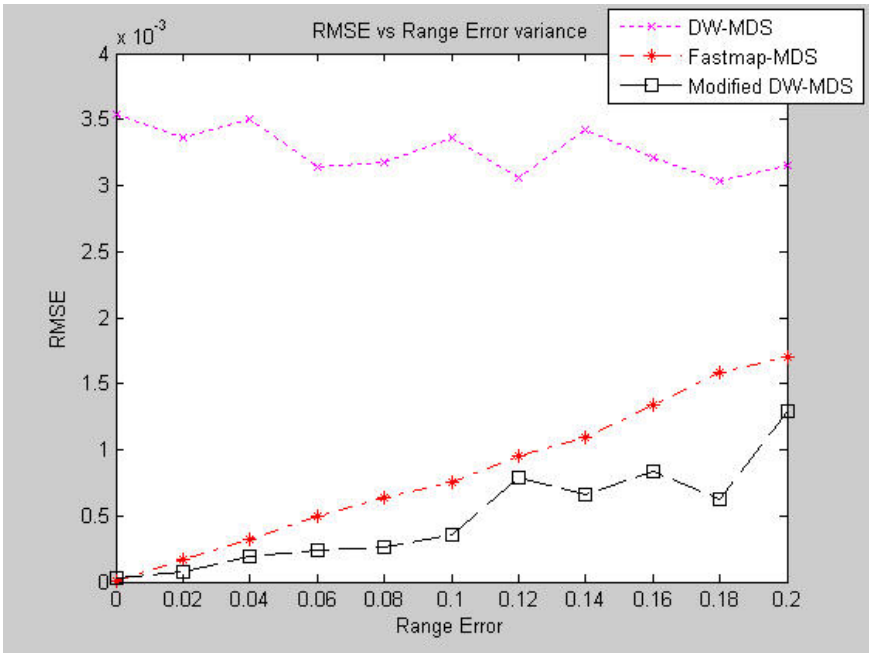

Fig. 6.1: Performance Measurement of DW-MDS, Fastmap-MDS and Modified DW-MDS, N=80

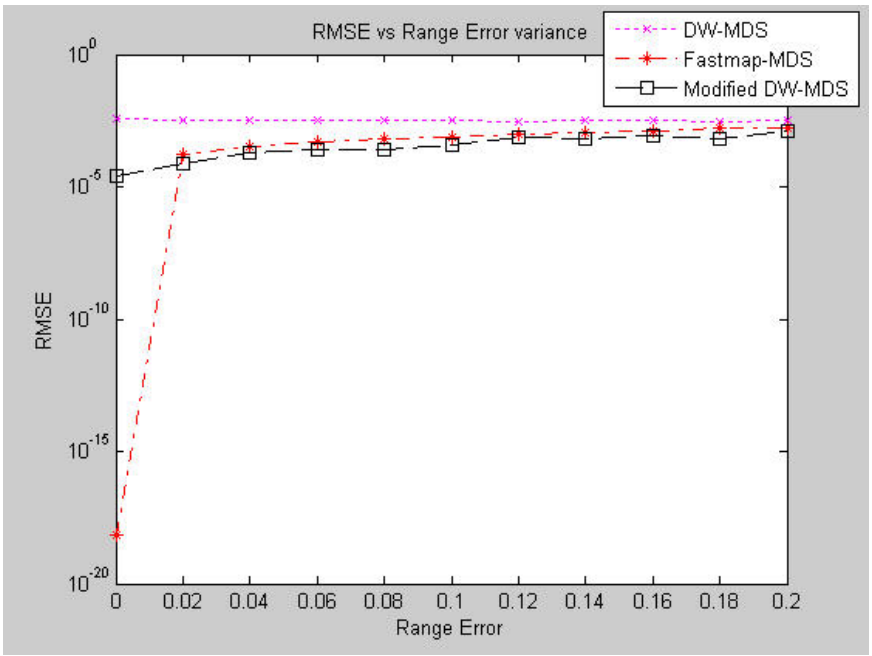

Fig. 6.2: Performance Measurement of DW-MDS, Fastmap-MDS and Modified DW-MDS - Logarithmic y-scale, N=80

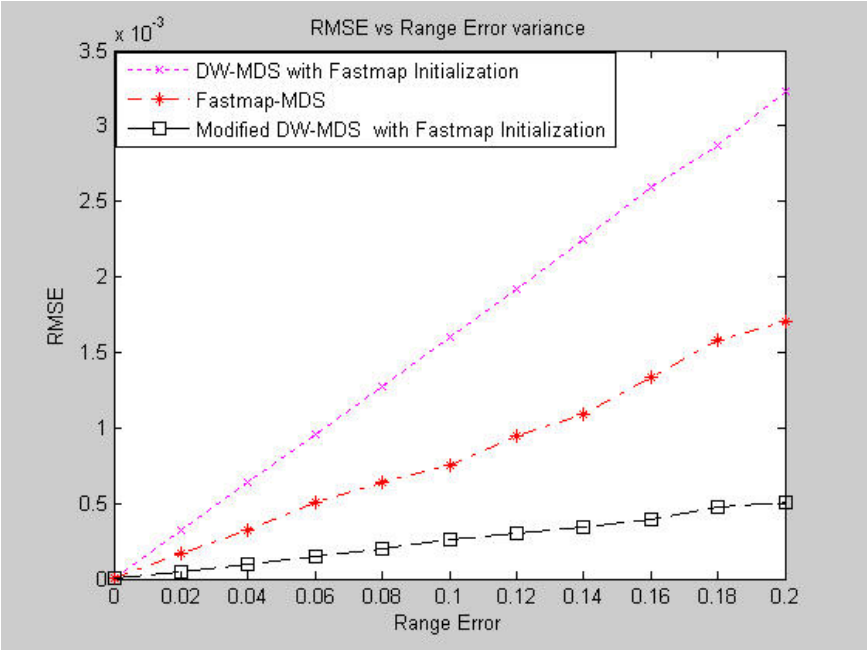

Fig. 6.3: Performance Measurement of DW-MDS with Fastmap Initialization, Fastmap-MDS and Modified DW-MDS with Fastmap Initialization, N=80

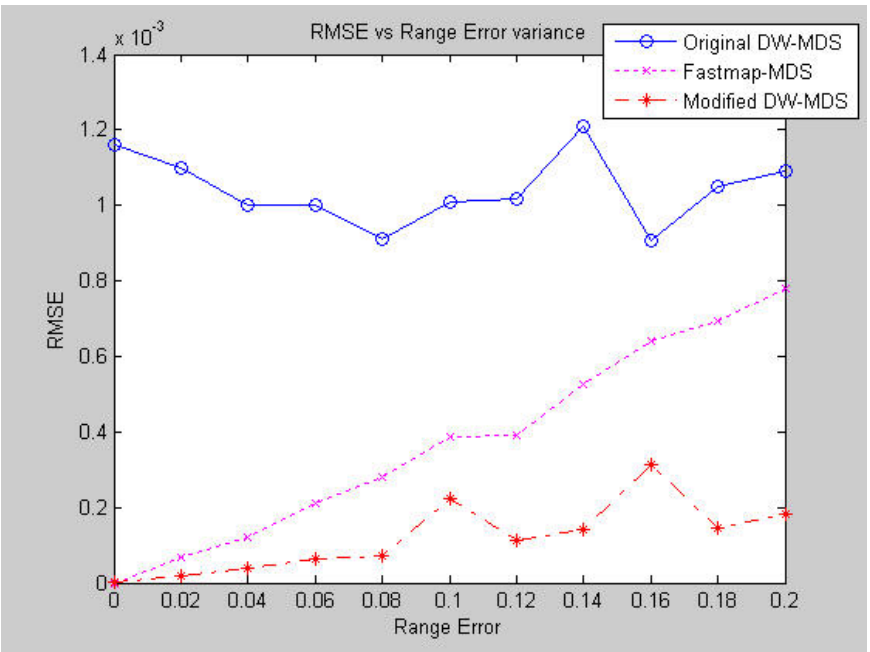

Fig. 6.4: Performance Measurement of DW-MDS, Fastmap-MDS and Modified DW-MDS, N=200

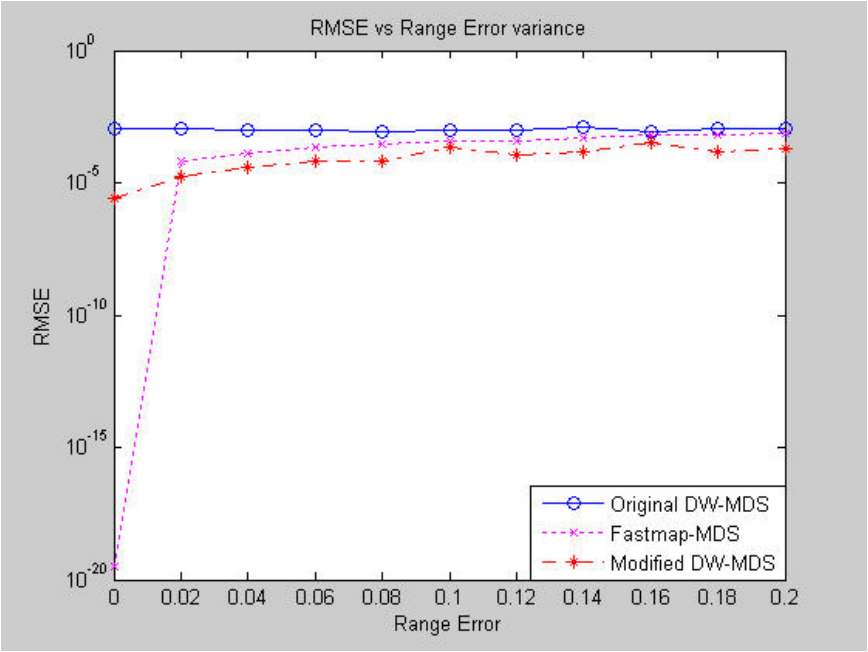

Fig. 6.5: Performance Measurement of DW-MDS, Fastmap-MDS and Modified DW-MDS - Logarithmic y-scale, N=200

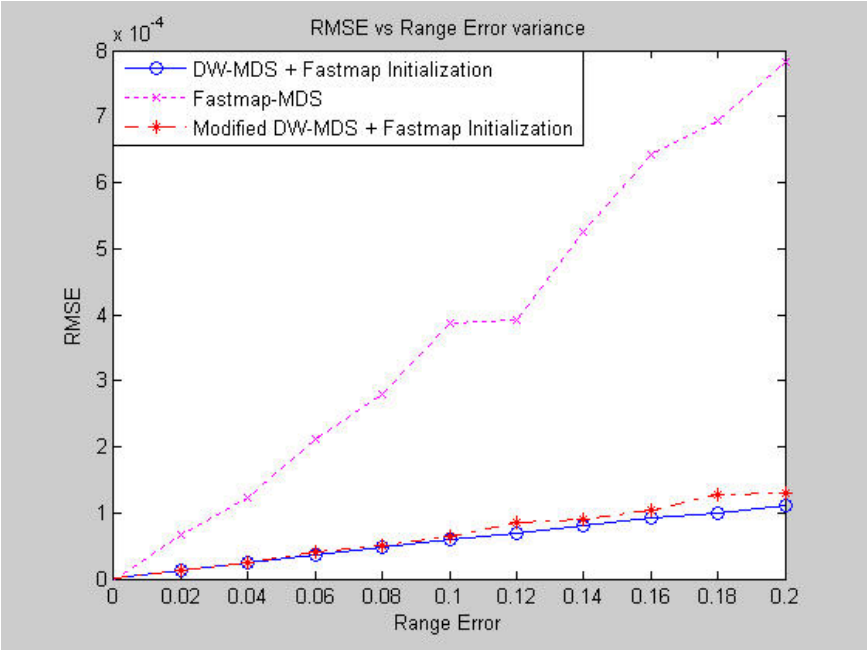

Fig. 6.6: Performance Measurement of DW-MDS with Fastmap Initialization, Fastmap-MDS and Modified DW-MDS with Fastmap Initialization, N=200

BIBLIOGRAPHY

[1] JOSE A. COSTA, NEAL PATWARI, and ALFRED O. HERO III, *Distributed Weighted-Multidimensional Scaling for Node Localization in Sensor Networks 2006*, University of Michigan.

[2] Jose A. Costa, Neal Patwari and Alfred O. Hero III, *Adaptive Distributed Multidimensional Scaling for Localization in Sensor Networks 2006.*

[3] Georgios Latsoudas, Nicholas D. Sidiropoulos, *A Two-Stage FASTMAP-MDS Approach for Node Localization in Sensor Networks*, Technical University of Crete.

[4] Christos Faloutsos, King-Ip Lin , *FastMap: A Fast Algorithm For Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets.*

[5] Chris Savarese, Jan M. Rabaey, Jan Beutel , *Locationing in Distributed Ad-Hoc Wireless Sensor Networks.*

[6] Leon Evers, Wouter Bach, Dennis Dam, Mischa Jonker, Hans Scholten, Paul Havinga, *An Iterative Quality-based Localization Algorithm for Ad Hoc Networks*, University of Twente, Department of Computer Science, Enschede, the Netherlands.

[7] Srdjan Capkun, Maher Hamdi, Jean Pierre Hubaux, *GPS-free positioning in mobile ad hoc networks.*

[8] Yi Shang, Wheeler Ruml, *Improved MDS-Based Localization*, University of Missouri-Columbia.

[9] Xiang Ji, Hongyuan Zha, *Sensor Positioning in Wireless Ad-hoc Sensor Networks Using Multidimensional Scaling*, The Pennsylvania State University.

[10] Ying Zhang, Qingfeng Huang and Juan Liu, *Sequential Localization Algorithm for Active Sensor Network Deployment.*

[11] Neal Patwari, Alfred O. Hero, Matt Perkins, Neiyer S. Correal, Robert J. O'Dea, *Relative Location Estimation in Wireless Sensor Networks.*

[12] FLEMING, R. AND KUSHNER, *Low-power, miniature, distributed position location and communication devices using ultra-wideband, nonsinusoidal communication technology*, Tech. rep.,Aetherwire Inc, Semi-Annual Technical Report, ARPA Contract J-FBI-94-058(July 1995).

[13] GIROD, L., BYCHKOVSKIY, V., ELSON, J., AND ESTRIN, *Locating tiny sensors in time and space: a case study*, In IEEE International Conference on Computer Design. 214-219, 2002.

[14] GROENEN, *The majorization approach to multidimensional scaling: Some problems and extensions*, DSWO Press, 1993.

[15] Y.B. Ko and N.H Vaidya, *Location aided routing in ad-hoc networks*, MOBICOM, 1998.

[16] Lj. Blazevic, S. Giordano and J. Y. Le Boudec, *Self Organizing Wide-Area routing*, SCI 2000 / ISAS 2000, Orlando, July 2000.

[17] Global Positioning System, *http://en.wikipedia.org/wiki/GPS*