Range-Free Ranking in Sensors Networks and Its Applications to Localization

Zvi Lotker*, Marc Martinez de Albeniz*, and Stéphane Pérénnes*

CNRS-INRIA-Sophia Antipolis-I3S, 06902, France
zvilo@eng.tau.ac.il, marcalbeniz@hotmail.com, stephane.perennes@inria.fr

Abstract. We address the question of finding sensors' coordinates, or at least an approximation of them, when the sensors' abilities are very weak. In a d dimensional space, we define an extremely relaxed notion of coordinates along dimension i. The $rank_i$ of a sensor s is the number of sensors with ith-coordinate less than the i-coordinate of s. In this paper we provide a theoretical foundation for sensor ranking, when one assumes that a few anchor sensors know their locations and that the others determine their rank only by exchanging information. We show that the rank problem can be solved in linear time in $\mathbb{R}$ and that it is NP-Hard in $\mathbb{R}^2$. We also study the usual localization problem and show that in general one cannot solve it; unless one knows a priori information on the sensors distribution.

1 Introduction

One of the fundamental problems in sensor networks is the *localization problem*, i.e., each sensor wishes to know its exact location. In the last two years several variations of this problem had been studied [6,13,11,7,16]. For an application to localization we refer the reader to [17]

Since sensors are supposed to be cheap the idea of using a GPS devices in every sensor is not practical. One possibility to reduce the cost is to divide the sensors into two sets. One set will serve as base stations and will be equipped with a GPS. The sensors in the second set will use the base stations in order to compute their location. Many localization algorithms for sensor networks have been proposed. There are mainly two groups of algorithms: *range based* and *range free*. In the range based case it is assumed that the sensors have an estimation of the point-to-point distance or angle. In the range free case no such assumption is made. Because of the hardware limitations of sensors networks devices, solutions in range-free localization are being pursued as a cost-effective alternative to more expensive range-based approaches.

In this paper we study the range free localization and ranking problems. The main questions that arise are: *Uniqueness* do we have enough information so that all the solutions that are consistent with the sensors observations are "the same"? *Practical feasibility*: Can we find at least one solution in reasonable time? *Distributed efficiency*: can we provide a fast distributed algorithm that solve the problem ? *Best solution*: Can we find the solution which is the most consistent with the observations?

* Project Mascotte CNRS-INRIA-Sophia Antipolis-I3S, 06902, France, this work was supported by the European project Aracne.

I. Nikolaidis et al. (Eds.): ADHOC-NOW 2004, LNCS 3158, pp. 158–171, 2004.

Our Results. We define the ranking problem for the d dimensional space ($\mathbb{R}^d$). For the one dimension case, we show that with uniform power (resp. arbitrary power), one can solve the range free ranking problem in linear time (resp. polynomial time). We also show an instances on which any algorithm that solves the range free localization can make large errors due to the lack of information. We also show that the best possible approximation on the coordinate value is a 2-approximation. Last, we show that when the sensors are identically and independently distributed (*i.i.d*), one can solve range free localization with high probability and good precision.

In the 2 dimensional case we show that range free ranking is not only $\mathcal{NP}$ hard but also APX hard. Our proof also implies that the range free localization is APX hard. As a corollary we prove that finding an approximated 2 dimensional realization of a unit disk graph is $\mathcal{NP}$ hard, thus answering an opened question in [13]

Related Work. Localization in Ad-hoc wireless networks and sensor networks have been the subject of extensive research in recent years. A detailed survey of the applications which describes a spectrum of current products and explores the latest research in that field is provided in [8]. The average estimation was proposed in [4] and was tested by simulation. The main differences between the average algorithm and our algorithm is that we compute the average location on all the sensors while the algorithm in [4] uses only the base stations (anchors). Another difference is that we use the average in order to compute the ranking, and then we use the ranking to get a more accurate estimation. In a later publication [5], improves his work by suggesting a novel density adaptive algorithm (HEAP) for placing additional anchors to reduce estimation error. Another approach for solving the localization proposed in [10] consists in sensing nodes and base stations. Instead of single hop broadcasts, anchors flood their location throughout the network maintaining a running hop-count at each node along the way. Nodes calculate their position based on the received anchor locations, the hop-count from the corresponding anchor, and the average-distance per hop; a value obtained through anchor communication again the algorithm was tasted by simulation. In the paper [7] another alternate solution performs best when an irregular radio pattern and random node placement are considered, and low communication overhead is desired this claim was demonstrated by simulation.

In [6] it is shown that for dimension 2 the range based localization problem is $\mathcal{NP}$ hard (mainly because finding a 2 dimensional realization of a network under distance constraints is $\mathcal{NP}$ hard). The authors also show how to solve the localization if the neighborhood graph is rigid.

Paper Organization. In Section 2 we give the basic definitions and notations. In Section 3 we prove our results for ranking in one dimension, we use the ranking problem to get estimation for the location when the sensors are uniformly distributed. In section 4 we study ranking and localization in dimension 2 and higher.

2 Model and Notation

Let the sensor set be S, sensors will be denoted s. In $\mathbb{R}^d$ we denote the real location of a sensor s by $(s(1), \ldots, s(d))$.

We assume that each sensor has a unique label $name(s)$. In the one dimensional case the rank of sensor s is the number of sensors that have a smaller coordinate than $s : rank_1(s) = |\{s' \in S : s'(1) < s(1)\}|$, and in the d dimensional case we have one ranking function per dimension : $rank_i(s) = |\{s' : s'(i) < s(i)\}|$.

The digraph associated to the sensor network has as vertices the sensors and contains an arc (s, s') whenever s can transmit to s', this digraph will be denoted G and we will assume that it is connected. The out-neighbors set of a sensor s will be denoted $N(s)$. The number of hops $h(s, s')$ from s to s' is the distance from s to s' in the graph G. We assume, unless specifically stated, that all sensors can transmit in a ball of radius r, according to some norm (usually the l_2 one), i.e. a sensor u can transmit to all the sensors v such that $|u - v| \leq r$. According to our hypothesis, the associated digraph G is a Unit Disk Graph [1] (UDG) of $\mathbb{R}^d$ for the chosen norm (usually l_2). This implies that G is a symmetric digraph, and we will identify it with its underlying graph.

We say that two sensors s, s' are equivalent if $N(s) = N(s')$. Since in the range free model two equivalent sensors cannot be distinguished, we assume that we do not have equivalent sensors (this is not restrictive since equivalence can be detected and one can elect one sensor per equivalence class). We also assume that communication is carried out in a synchronous manner, i.e., all the vertices are driven by a global clock. Messages are sent at the beginning of each round, and are received at the end of the round. (Clearly, our lower bounds hold for asynchronous networks as well). Since we work above the MAC layer we allow any set of communication to be performed during a round. The algorithms can be modified for asynchronous using standard techniques.

To conclude note that according to our model the only information that an algorithm can extract from the sensors observation is the graph G and the location of the base stations.

3 One Dimensional Case

In many practical cases i.e., in industrial line, road net, narrow street, along a fence or border we can assume that the sensors are placed in a one dimension environment. As we will see, the one dimensional case is much easier than the two dimensional one.

Without loss of generality, we assume in this section that the sensors are placed in the unit interval and that there are only two base stations placed at 0 and 1; we denote them respectively B_0, B_1.

Consider first the more general case, in which the sensor range may vary. Each sensor s knows its transmission range $r(s)$. The graph G associated to the sensor network is then almost like an *interval graph* (see [9] for a definition). Theoretically to solve ranking or localization one would gather all the sensor observations: that is the adjacency matrix of the network G and the sensor ranges and then try to find 1 dimensional representation of G in which vertex B_0 is mapped on 0 and vertex B_1 on 1 and where the interval of transmission associated to sensor s has length $r(s)$. Using the same ideas that allows

[1] Unit disk graphs are intersection graphs: each vertex is represented by a disk and two vertices are adjacent if the disks overlap. Note that in our case 2 nodes are connected if the center of one disk is included in the other disk, and vice versa. However, there is a simple reduction between the 2 models: if we divide the radius by 2, we get the unit disk graph model.

to recognize interval graphs (see [9]), one can easily find a solution to the localization problem (note that in section 3.4 we will see that finding a solution reduces to solving a simple linear program). This implies that ranking is unique and can be computed in polynomial time, still the actual localization of the sensors remains very uncertain.

In the next subsections we use the following definitions. Let $\mathcal{A}_i$ be an algorithm that computes the estimated location of a sensor s_i. For each sensor i we denote by $\mathcal{A}_i^t(x)$ the result of the algorithm $\mathcal{A}_i$ in the t iteration. In two dimensions we use the $\mathcal{A}_i^t(x)$ and $\mathcal{A}_i^t(y)$ for the x, y coordinates that are computed by the algorithm $\mathcal{A}_i$ in the t iteration.

3.1 Ranking

The fact that one can solve the ranking problem is mainly due to the fact that there is a natural ordering in $\mathbb{R}$ (and not in $\mathbb{R}^2$). Indeed the 1 dimensional representations of the graph G can be partitioned into 2 classes, in the first one a sensor s gets rank $r(s)$ and in the other $n - r(s)$. We break this left-right symmetry since we know that station B_0 has rank 0.

We now describe a fast distributed algorithm that solves the ranking problem when the sensors range is uniform in $O(h(s_1, B_1))$ time steps.

The general idea of the algorithm is to use the fact that different sensors see different environments. Furthermore, if all the sensors start with being in the neighborhood of the base station in B_0 eventually sensors that are near B_1 will have a bigger average. Our algorithm works in $\lceil \frac{1}{r} \rceil$ phases.

Lemma 3.1. *If* $0 \leq a_1 \leq a_2 \leq a_3 \leq ... \leq a_i \leq ... \leq a_{i+j} \leq ... \leq a_{i+j+k}$ *then* $\frac{\sum_{n=1}^{i+j} a_n}{i+j} \leq \frac{\sum_{m=i}^{i+j+k} a_m}{j+k}$, *and equality only if* $a_1 = a_{i+j+k}$.

Proof. By induction on $i + j + k$, if $i + j + k = 1$ then the lemma holds. Assume that $i + j + k = n + 1$. If $a_1 = 0$ then $\frac{\sum_{n=1}^{i+j} a_n}{i+j} \leq \frac{\sum_{n=2}^{i+j} a_n}{i+j-1}$ and we can use the induction. If $a_1 > 0$ we can define a new variable $a'_k = a_k - a_1$ and we get that $a'_1 = 0$ so we can use the previous argument.

In order to prove the correctness of the algorithm we need the following definition. Let the set of sensors $\Gamma_i = \{s : h(s, B_1) \leq i\}$ be the sensors that are in hop-distance $\leq i$ from B_1.

Lemma 3.2. $\mathcal{A}_j^t(\mathcal{X}) > 0$ *if and only if* $s \in \Gamma_t$.

Proof. The proof is by induction on the number t. For $t = 1$, the only sensors that don't change their estimation of their location are the ones that receive the transmission from B_1. This is exactly Γ_1. The rest of the sensors still estimate their location to be 0.

Now assume the lemma holds for $t = k$. From lemma 3.1 we get that for all $s_j \in \Gamma_k$, $\mathcal{A}_j^{k+1}(\mathcal{X}) > 0$. Now let $s \in S \setminus \Gamma_{k+1}$ since $N(s) \cap \Gamma_k = \emptyset$ and using the induction hypothesis we get that $\mathcal{A}_j^t(\mathcal{X}) = 0$. It remains to prove that for all $s_j \in \Gamma_{k+1} \setminus \Gamma_k$, $\mathcal{A}_j^{k+1}(\mathcal{X}) > 0$. In this case the lemma follows from the fact that $N(s) \cap \Gamma_k \neq \emptyset$ and lemma 3.1.

Using the previous lemma we get a bound on the time $stop_0$ signal is sent.

Initialize s_i,
$\mathcal{A}_i^0(\mathcal{X}) \leftarrow 0$
$rank(s_i) = \emptyset$
$stop \leftarrow 0$
While ($stop = 0$)
$\mathcal{A}_i^{t+1}(\mathcal{X}) \leftarrow \frac{\sum_{x_j \in N(x_i)} \mathcal{A}_j^t(\mathcal{X})}{|N(x_i)|}$
if ($\exists s_j \in N(s_i) \wedge rank(s_j) \neq \emptyset \wedge rank(s_i) = \emptyset$)
then $rank(s_i) \leftarrow rank(s_j) + |\{k \in N(s_i) : \mathcal{A}_j^t(\mathcal{X}) < \mathcal{A}_k^t(\mathcal{X}) < \mathcal{A}_i^t(\mathcal{X})\}|$
if ($B_0 \in N(s_i) \wedge \forall j \in N(s_i), \mathcal{A}_i^t(\mathcal{X}) < \mathcal{A}_j^t(\mathcal{X})$)
then $rank(s_i) \leftarrow 0$
transmit $\mathcal{A}_i^{t+1}(\mathcal{X})$
transmit $rank(s_i)$
If(receive $stop_0 \wedge$ receive $stop_1$)
then $stop \leftarrow 1$
$t \leftarrow t + 1$
if(s_i receive $stop_k$ signal)
then transmit $stop_k$ signal

Fig. 1. Pseudo code for algorithm for sensor s_i

Transmit the location i
If($\forall j \in N(B_i), \mathcal{A}_j^t(\mathcal{X}) > 0$)
then transmit $stop_i$ signal.

Fig. 2. Pseudo code for algorithm for base station B_i

Corollary 3.3. *B_0 transmit the $stop_0$ at time $h(B_0, B_1) + 1$*

The next lemma shows that after n iterations Γ_n is sorted.

Lemma 3.4. *After $t = h(s_1, B_1)$ iterations, if $x_i < x_j$ then $0 < \mathcal{A}^t(x_i) < \mathcal{A}^t(x_j)$.*

Proof. The proof is by induction on the number of hops from the base station. We claim that after n iterations Γ_n is sorted. First we see that Γ_1 is sorted after the first iteration. Since all sensors in Γ_1 see B_1 and all the other values are 0, and so from lemma 3.1 the sensors in Γ_1 are sorted after applying the first iteration. Now assume that Γ_n is sorted at time $t = n$. For $s_i \in \Gamma_{n+1} \setminus \Gamma_n$, since $\mathcal{A}_i^n(\mathcal{X}) = 0$. From the induction hypothesis we get that for all $j \in \Gamma_n$, $\mathcal{A}_i^n(\mathcal{X}) > 0$. From the fact that the graph G_N is connected we get that $N(s_i) \bigcap \Gamma_n \neq \emptyset$. Now using lemma 3.1, we get that all sensors in $\Gamma_{n+1} - \Gamma_n$ will be sorted at time $n + 1$. Sensors in Γ_n will still be sorted as long as they compute their average at the same time step.

The next lemma shows that the algorithm stops after $2h(B_0, B_1) + 1$ iterations.

Lemma 3.5. *After $t = 2h(B_0, B_1) + 1$ iterations, the algorithm stops and if $s_1 < s_1'$ then $rank(s) < rank(s')$*

Proof. By corollary 3.3 B_0 sends the $stop_0$ signal at time $h(s_1, B_1) + 1$ and B_1 sends the $stop_1$ signal at time 2. In order for the sensors to stop computing their new location they have to receive both these signals. It follows that none of the sensors sets its $stop$ to 1 before time $h(s_1, B_1)+1$, and the last of the $stop$ signals is set at time $2h(B_0, B_1)+1$, since this is the time that it takes for the signal $stop_0$ to reach the last sensors. Note that all the sensors set their ranks before time $h(B_0, B_1)+1$(before signal $stop_0$ was sent). From lemma 3.4 we get that the ranks are consistent i.e., if $s_1 < s_1'$ then $rank(s) < rank(s)$. Now the $stop_1$ signal and the ranks spread at the same speed and arrive at the last sensors at time $h(B_0, B_1)$. So the signal that makes the sensors set their $stop$ signals is actually $stop_0$.

Theorem 3.5.1. In one dimension the ranking problem can be solved in linear time.

3.2 Uniqueness of Localization

We remark that knowing only the graph G we can get a 2 approximation of the localization, indeed if $h(s, B_0) = k$ then $s(1)$ is at least $\frac{k-1}{2} \cdot r$ and at most $k \cdot r$. We show now that even if it is easy to find a localization of the sensors that is consistent with the observations, there exist instances for which one can find two consistent localizations in which the coordinates may differ by more than $1/6$ (and since the sensors are in $[0, 1]$ this is about 16% of the network width).

Note that sensor localization is not "hard"algorithmically, it's simply impossible since we miss information.

Theorem 3.6. *For any algorithm $\mathcal{A}$ that solves the localization problem in one dimension there exists a location of sensors $x_1, x_2, ..., x_n$ such that the error of the estimation $|x_i - \mathcal{A}(x_i)|$ for some i is bigger than $1/6$.*

Proof. Let ρ be the radius of the sensors. We put $n = \frac{1}{\rho + \frac{\rho+\epsilon}{2}}$ points in the interval $[0, 1]$. This means that $n\rho + \frac{n}{2}(\rho + \epsilon) = 1$. We consider two different scenarios, G and G'. The location in G is:

$$x_i = \begin{cases} i \cdot \rho & i \leq \frac{n}{2} \\ \frac{n\rho}{2} + k(\rho + \epsilon) + \rho & i = \frac{n}{2} + (2k + 1) \\ \frac{n\rho}{2} + k(\rho + \epsilon) & i = \frac{n}{2} + 2k. \end{cases}$$

Similarly, the location of G' is:

$$x_i = \begin{cases} k(\rho + \epsilon) + \rho & i = (2k + 1) \leq \frac{n}{2} \\ k(\rho + \epsilon) & i = 2k \leq \frac{n}{2}. \\ \frac{n}{4}(\rho + \epsilon) + (i - \frac{n}{2}) \cdot \rho & i \geq \frac{n}{2} \end{cases}$$

If we let ϵ small enough, both locations give the same neighborhood graph and we can't differentiate them. We see that forcing $\epsilon \to 0$, for $i = n/2$, in G, x_i is placed on $2/3$ and in G', x_i is placed in $1/3$. Since the distance between them is $1/3$, the best estimation we can do will give an error of $1/6$. This proves the theorem.

3.3 Approximated Localization Under Distribution Assumption

We now use the rank in order to get a location estimator under the assumption that the sensors distribution is known. We illustrate the idea with an example that uses the uniform distribution.

Assume that the sensors are $i.i.d$ distributed, we can use the ranking algorithm combined with the distribution of the order statistics to solve the localization problem. Naturally the localization of the sensor with rank i will be $E[x^{(i)}]$ which is the expectation of the i-order statistics. We remark that the variance of the i-order statistics has been well studied (for a background on this subject we refer the reader to [1]. For example assume the sensors are placed by a uniform distribution in the $[0, 1]$. In this case it is well known that the ith order statistics has a $Beta(i, n - i + 1)$ distribution. Therefore the algorithm will compute the location of the sensor with rank i as being $\frac{i}{n+1}$, and the variance of this location is $\frac{i(n+1-i)}{(n+1)^2(n+2)}$. Simple calculation shows that the middle sensor with rank $\frac{n+1}{2}$ has the maximum variance, which is equal to $\frac{1}{4(n+2)}$, this variance is extremely small compared to the $1/6$ error that one can made in the deterministic case.

For a general distribution, one needs simply to compute an estimation of $E[x^{(i)}]$ and to output it as coordinate. The quality of this rank based estimator will depend on the variance of the i-order statistic.

3.4 Best Localization

According to section 3.2 we have many solutions that are consistent with the observation. Therefore we can not hope to find the sensor coordinates, but we may find a "best solution". Many different quality measures may be chosen. The most natural measure would be to choose the mean of all the solutions. We start with some definitions. A solution of the localization problem can be identified with a vector of $\mathbb{R}^n$. Let s_i denote the sensor with rank i, we associate to a solution A the vector $v(A)$ of $\mathbb{R}^n$ defined by $v(A)_i = s_i(1)$ ($v(S)_i$ is the coordinate of s_i in the solution). The solution set happens to be a convex polytope P, indeed a vector v represents a solution if and only if :

- $\forall i, j, |v_i - v_j| \geq r$ if s_i cannot transmit to s_j;
- $\forall i, j, |v_i - v_j| \leq r$ if s_i can transmit to s_j
- $v_1 = 0, v_n = 1, \forall i, v_i \geq 0, v_i \leq 1$.

This system is non linear but if one assumes that the sensors are ranked the sign of $v_i - v_j$ is known and the system becomes linear. The mean $\overline{v}$ of all solutions minimizes the expectancy of $\sum_{v \in P} |v - \overline{v}|^2$, assuming that all the solutions $v \in P$ are equally likely.

Finding the average solution reduces to computing the center of mass of polytope P. This problem is believed to be $\mathcal{NP}$ hard, but one can get a good approximation of the center of mass by sampling the points of the polytope and taking the sample average. In order to get in polynomial time a good sample of the polytope one can simply start from a point inside the polytope and walk randomly using a *hit and run walk* [12].

In our case we can get a better result, since the solution polytope is very particular. Indeed one can show that its vertices are integral and use Fourrier Motzkin [15] method to compute the integral $\int_{x \in P} x$ exactly in polynomial time.

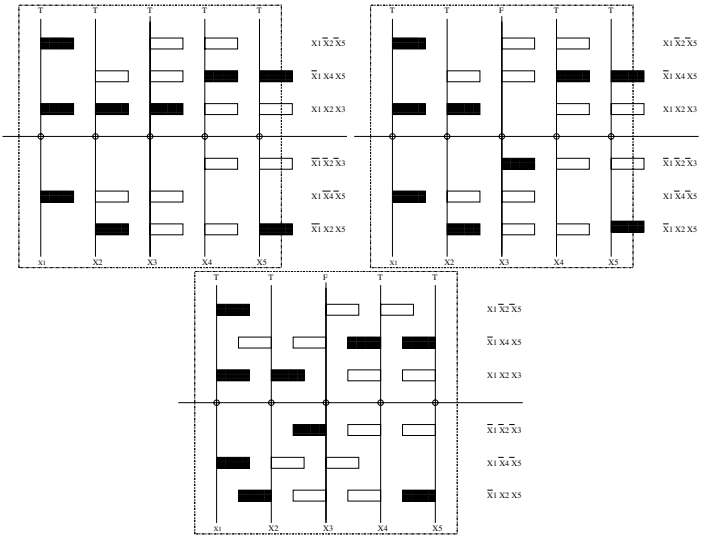

Fig. 3. Exempla of a Flip Game for the formula $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_4 \vee x_5) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_5})$

Theorem 3.6.1. In the one dimensional case one can compute the average solution to the localization problem in time $O(n^4)$.

4 Two Dimensional Case

In this section we sketch the proof that it is $\mathcal{NP}$-hard to solve the ranking and the localization problem in dimension 2. We also show that there exists no polynomial algorithm computing a localization that is approximately consistent with the observations unless $\mathcal{P} = \mathcal{NP}$, this implies that finding an approximated drawing of unit disk graphs is $\mathcal{NP}$ hard. This problem was mentioned as open in [14]; indeed the exact drawing problem was known to be $\mathcal{NP}$ hard [3], but the proof was much more complex than the one we give and could not be extended to the approximated case.

Note that we just give the main ideas of our proofs, a detailed version will be given in the final version.

We first give a sketch of the hardness proof. As in [2], we use a reduction to a specific form, a $3Sat$, the Jigsaw that we use is also directly inspired from this work.

The *Symmetric* $3Sat$ *problem* in which one wishes to satisfy not only any clause C of a formula F but also all the symmetric clauses $\overline{C}$ (eg. at least one literal is false in any clause) is $\mathcal{NP}$ Hard.

A Jigsaw Game to Emulate Symmetric $3Sat$

The next Jigsaw that we will call *Flip game* is a game which emulates a symmetric $3Sat$ instance with a formula F. For an example of the game see figure 3.

- The Jigsaw is made with one horizontal axis on the $y = 0$ axis; on this axis n vertical rigid *bars* $B_1, B_2, \ldots, B_n$ can rotate (more exactly for each bar we can use the $y = 0$ symmetry).
- The bar B_i represents the variable x_i, and can be set up to represent the setting $x_i = TRUE$ or down $x_i = FALSE$.
- The half plane $y > 0$ is divided in to m strips $S_1, S_2, ..., S_m$, each strip S_i representing the clause C_i.
- symmetrically, the half plane $y < 0$ is divided in to m strips $\overline{S_1}, \overline{S_2}, \ldots, \overline{S_m}$, the strip $\overline{S_i}$ representing the clause $\overline{C_i}$.
- On each bar are attached *flippers* , a flipper is a rigid structure that can be flapped by symmetry around the bar axis.
- Flippers are attached to the bar with the following rules :

  If x_i is non negated in clause j we put a flipper on bar B_i in the up-strip j
  If x_i is negated in clause j we put a flipper on bar B_i in the down-strip j
  If x_i does not appear in clause j we put one flipper in the down-strip and the up-strip j.
  The space between two bars can contain at most one flipper.

The Flip game consists in rotating the bars and flipping the flippers so that flippers do not overlap.

Lemma 4.1. *The Flip game is $\mathcal{NP}$-Hard.*

Proof. Consider a winning position, we associate to it a Truth setting (x_i is true is the bar B_i is up). Since any strip can hold at most $n-1$ non overlapping flippers, we conclude the bars are set so that in each strip we find at most $n-1$ flippers. The number of flippers in any up (resp. down) strip i is respectively n minus the number of true literals (minus the number of false literals) in clause C_i. Hence flippers are non overlapping only if the associated truth setting satisfies F and $\overline{F}$.

Conversely, given any truth setting that satisfies the formula F we can set the bars according to it, them we have enough room to set the flippers in non overlapping position by flipping then left or right.

Emulating the Flip Game
Overall Argument

Consider a flip game, and any winning position drawn in the plane. We will associate to this drawing a sensor network with uniform transmission range 1, the key point is that the associated graph G does not depends on the winning position of the flip game.

Now, assume that we are given the graph G, due to rigidity constraints we can easily recover the flip game structure and moreover if we want to draw G and respect the sensor observations we have to draw it as a flip game in a winning position. Indeed the only freedom that we have when we try to draw G corresponds exactly to the possible moves of the flip game, and due to the sensors observations we cannot drawn overlapping flippers. Hence if we manage to draw G we have been able to find a winning position of the flip game.

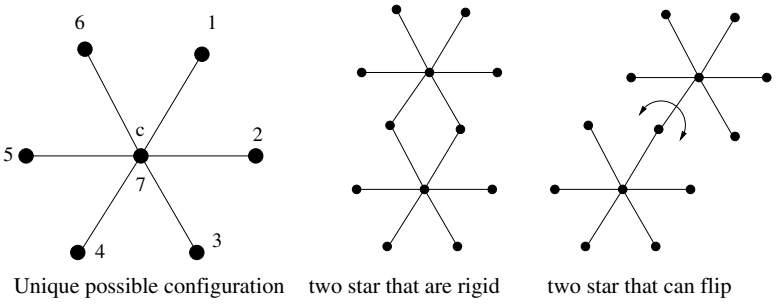

Fig. 4. Example of the three simple gadgets, the first is the rigid 7-star; this star has 7 leaves, note that in our drawing 2 points are placed at the center (the center of the star and one leave). The second gadget show how to connect two rigid 6-star and get a rigid shape. The last gadgets is the flipper gadget.

To emulate the flip game with sensors we will first force the sensors to be on the hexagon lattice using rigid graph structures; second we will maintains the jigsaw game degree of freedom. For this we use two gadgets; we call the first the rigid gadget, since it create a rigid bounding box which forces all the sensor to be set on the hexagon lattice. We call the second gadget flipper.

In what follows, we consider a drawing of a graph G modulo $\mathbb{R}^2$ isometries (for l_2 that is rotations translation), eg isometric drawings are considered as identical. When $d(u, v) = 1$ we may choose to have $[u, v]$ in $E(G)$ or not. We call *rigid* any graph which admits up to isometries one unique drawing.

Standard ball packing argument shows that the 7-star is rigid see figure 4, so if the graph G contains a 7 star it must drawn like on figure 4 up to translation or rotation.

Using those as basic blocks we can create rigid graphs of any shape on the hexagon lattice. We Remark that whenever we fix two vertices a and b of a rigid subgraph there are at most two ways to draw it, that are symmetric along the line (a, b).

Our graph is made from rigid components, components that are attached to the remaining of the structure by either 2 points (u, v) (so they can rotate along the (a, b) axis) or one point u (the can rotate around u) see figure 5.

We describe here how to simulate a Flip game (see figure 5, and figure 6 for an examples).

- We start defining our graph by making a large rigid bounding box. Inside we add a line of hexagons. At two opposite points of the box, this line is flexible but since it contains exactly enough hexagons to connect the two points, it must be drawn straight. We call this line the horizontal bar (see fig...). Note that each hexagon of the line still has the freedom to rotate around the y axis. Bars will be attached to those hexagons allowing them to rotate.
- Each Bar is built from a vertical line of hexagons, this line is associated to a rigid structure that force it to be drawn straight. On each bar each hexagon can be rotated around the bar axis.

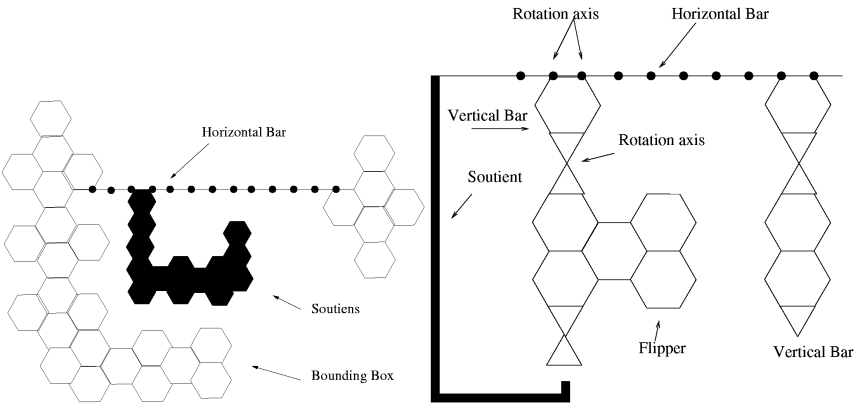

Fig. 5. Example of a Flip Game emulation

- Flippers are simply rigid structures that are attached to the bars, note that they can be flipped.

4.1 Range-Free Localization in Two Dimension Is APX

From our proof, it follows that we cannot approximate the ranking (that is find approximated ranks that correspond to localizations that are consistent with the data). Indeed, in our reduction ranking and localization are the same problem since the sensors are forced to be on the hexagonal lattice.

We now extend our result and sketch the proof that finding a localization that is *almost* consistent with the observations is $\mathcal{NP}$ hard.

Definition 1. *A localization is an ϵ approximated solution if s can transmit to s' when $d(s, s') \leq r$, s cannot transmit to s' if $d(s, s') \geq (1 - \epsilon)r$.*

The above definition simply means that the location of the sensors is such that the transmission range r is more or less respected.

Theorem 4.1.1.

- *In dimension 2, there exists an $\epsilon > 0$ such that finding an ϵ approximated localization is $\mathcal{NP}$ hard.*
- *finding an ϵ approximated realization of a unit disk graph is $\mathcal{NP}$ hard.*

Assume that we allow non edges to have length $1 - \varepsilon$ then the structures that we use are not perfectly rigid. But one can show that if ε is fixed and small enough the drawing of the rigid structures remains unique (up to isomorphism) and to some move of ε' that are sufficiently small so that disjoint structures cannot overlap. To avoid the small moves to add up and to bend the lines we use the next gadgets:

- We want to force the flippers of a given strip to be drawn in the same horizontal area, in order to bound the non rigidity drift we add on each bar and each strip a device that force the chunk i of two adjacent bars to be drawn in the same area (cf figure 7).

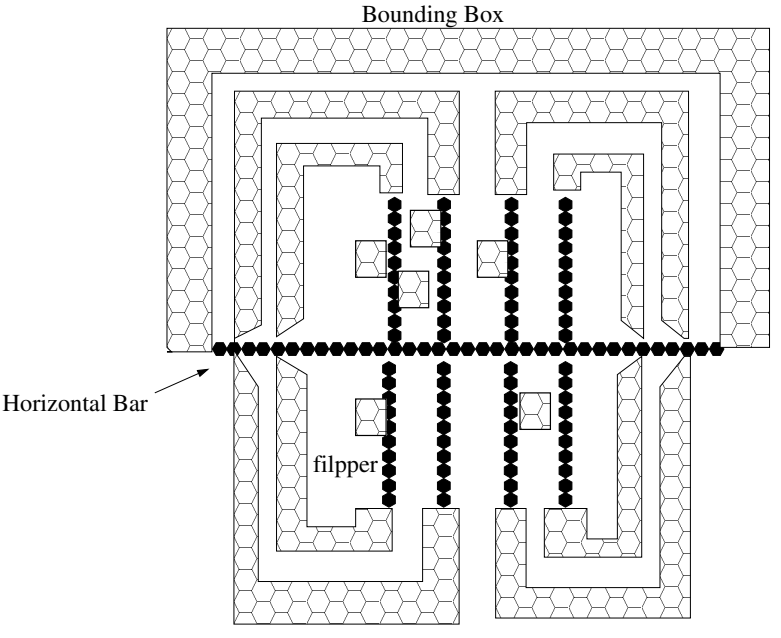

Fig. 6. The big picture

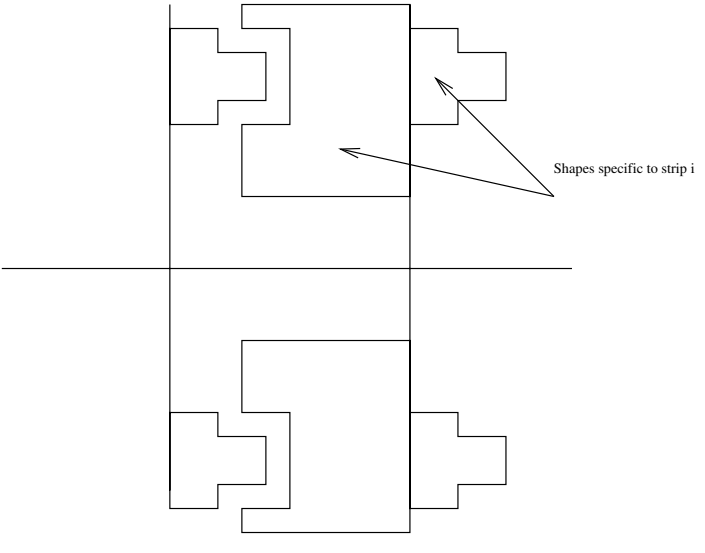

Fig. 7. Synchronizing bars with specific shapes

– We choose the flippers of adjacent bars large enough on the horizontal side so that they would overlap if they are flapped on the same size.

Assume that L is the distance of bars when they are rigid, L is also the horizontal size of a flipper. Since we have m strips, when the bars are not rigid the distance between them in some strip is at most $L(1+\varepsilon')+\varepsilon' m$. We need it to be less than $2L$ in order to prevent two flippers to be drawn on the same side. If we choose $L = m$ both conditions are satisfied for $\varepsilon' \leq 1/2$.

5 Conclusion

In this paper we introduced the range free ranking problem. We showed that from a complexity point of view there is a big difference between ranking in one dimension and ranking in two dimensions. We also shown that in one dimension there is an important difference between worst case scenario and a random scenario, for example in the random uniform case the error (variance $< \frac{1}{4(n+2)}$) is a function of the number of sensors while the error in the worst case can be of 16% of the total network width. This result shows that in the random case it is better to include the sensors in the averaging process, and not to depend only on the anchors (base stations). We also proved some hardness results, and one implies that drawing approximately unit disk graphs is $\mathcal{NP}$ hard.

References

[1] Barry C. Arnold, N. Balakrishnan, H. N. Nagaraja July, A First Course in Order Statistics, Wiley Series in Probability and Mathematical Statistics: Probability and mathematical Statistics, 1992.

[2] N. Bhatt , Stavros S. Cosmadakis, The complexity of minimizing wire lengths in VLSI layouts, Information Processing Letters, v.25 n.4, pp. 263-267, June 17, 1987

[3] Heinz Breu and David G. Kirkpatrick. Unit Disk Graph Recognition is NP-Hard. Computational Geometry, 9:3-24, January 1998.

[4] N. Bulusu, J. Heidemann and D. Estrin, GPS-less Low CostOutdoor Localization for Very Small Devices, IEEE Personal Communications Magazine, 7(5):28-34, October 2000.

[5] N. Bulusu, J. Heidemann and D. Estrin, Density Adaptive Algorithms for Beacon Placement in Wireless Sensor Networks, In IEEE ICDCS '01, Phoenix, AZ, April 2001.

[6] T. Eren, D. Goldenberg, W. Whiteley, Y.R. Yang, A.S. Morse, B.D.O. Anderson, P. N. Belhumeur.Rigidity, Computation and Randomization in Network Localization, IEEE INFOCOM 2004.

[7] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, Tarek Abdelzaher, Range-free localization schemes for large scale sensor networks, MobiCom 2003, pp 81-95.

[8] J. Hightower and G. Boriello. Location systems for ubiquitous computing. IEEE Computer, 34(8) Aug. 2001, pp 57-66.

[9] N. Korte and R. H. Mohring. An incremental linear time algorithm for recognizing interval graphs. SIAM J. Comput., 18:68–81, 1989

[10] D. Niculescu and B. Nath, DV Based Positioning in Ad hoc Networks, In Journal of Telecommunication Systems, 2003.

[11] L. oherty, K. S. J. Pister, and L. E. Ghaoui. Convex position estimation in wireless sensor networks. In Proceedings of the IEEE Infocom, pages 1655-1663, Alaska, April 2001.

[12] L. Lovasz, Hit-and-run mixes fast, Mathematical Programming, Issue: Volume 86, Number 3, December 1999 pages 443-461.

[13] Teemu Roos, Petri Myllymaki, Henry Tirri, A Statistical Modeling Approach to Location Estimation, IEEE Transactions on Mobile Computing – January 01, 2002 Issue 1, pages 59-69.

[14] Thomas Rusterholz, Report On the Approximation of Unit Disk Graph Coordinates, http://dcg.ethz.ch/theses/ss03/virtualCoordinates_report.pdf.

[15] M. Schechter. Integration over Polyhedra, an Application of the Fourier-Motzkin Method", Am. Math. Monthly, 105(1997), pages 246-251.

[16] Yi Shang, Wheeler Ruml, Ying Zhang, Markus P. J. Fromherz, Localization from mere connectivity, MobiHoc 2003. pages 201-212.

[17] (Editor) Ivan Stojmenovi, Handbook of Wireless Networks and Mobile Computing. John Wiley and Sons. February 2002.