

## Localization of Sensor Networks

Lecturer: Jie Gao

Scribe: Girishkumar Sabhnani

### 1 Lecture Summary

We present a distributed, linear time, anchor-free algorithm to localize the sensor nodes, followed by the NP-hardness proof of Unit Disk Graph embedding.

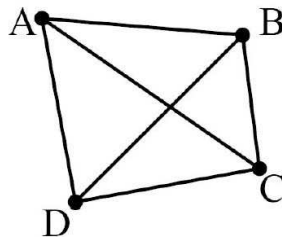
### 2 Distributed Network Localization

[2] describes a robust distributed algorithm for sensor network localization with noisy range measurements. The notion of *robust quadrilaterals* is used to localize the nodes in the absence of anchor nodes. Briefly the algorithm can be explained by the following steps :

- Identify robust quadrilaterals, merge them to form bigger clusters and localize the cluster with a local coordinate system.
- Optimize the node locations inside a cluster. This is an optional phase.
- Stick the local clusters and transform them to a global coordinate system with suitable rotation, translation and reflection as needed.

#### 2.1 Robust Quadrilaterals

As shown in figure 1, robust quadrilateral consists of four nodes with fixed pairwise distances. Note that it is the smallest 3-connected redundantly rigid graph, thus is a globally rigid graph. Also there is a unique realization of such a robust quadrilateral if the noise in distance measurement is bounded.



**Figure 1:** Robust quadrilateral in which all the 6 pairwise distances between the four nodes are known.

##### 2.1.1 Bound on the distance measurement noise

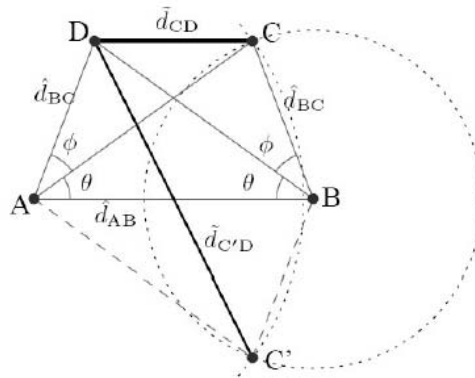
The distance measurement noise needs to be bounded in order to avoid the flips of the vertices giving more than one realization. We define *k-flip* to be one in which *k* vertices exchange their positions.

Note that *4-flip* is uninteresting as it gives the same robust quadrilateral, rotated from the original configuration. Also *2-flip* in which diagonally opposite vertices get flipped gives the reflection of the original configuration. Refer figure 2



**Figure 2:** (a) Original Robust Quadrilateral (b) 4-flip resulting in same quadrilateral, rotated (c) 2-flip resulting in reflection of the original quadrilateral

All other flips, i.e. the *1-flip* as shown in figure 3, *2-flip* in which the adjacent vertices get flipped and the *3-flip* result in swapping the intersecting edges and hence by bounding the error such flips can be avoided.



**Figure 3:** Example of 1-flip : A configuration in which the vertex C is flipped giving alternate realization for the robust quadrilateral

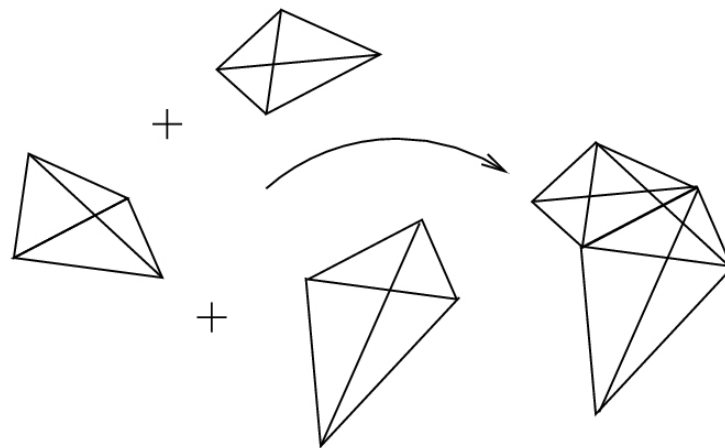
$$d_{err} = \frac{d_{C'D} - d_{CD}}{2} = d_{AB} \frac{\sqrt{\sin^2 \phi + 4 \sin^2(\theta + \phi) \sin^2 \theta} - \sin \phi}{2 \sin(2\theta + \phi)}$$

The error can be minimized by choosing  $\phi = \pi/2 - 2\theta$  resulting in  $d_{err} = d_{AB} \sin^2 \theta$

A more rigorous bound of  $d_{err} < b \sin^2 \theta$  assures that no flip takes place and hence assures the robust nature of the quadrilateral.

### 2.1.2 Assigning local coordinates

The robust quadrilaterals assure unique shape realization and hence a local coordinate system can be fitted according to the distance measurements. The robust quadrilaterals sharing three vertices are glued to form clusters and the local coordinates are extended. Hence each such cluster is assigned local coordinates as shown in figure 4.



**Figure 4:** Assigning local coordinates to robust quadrilaterals and the corresponding clusters formed by gluing quadrilaterals that share three vertices.

Optionally the position estimates can be refined by using *mass-spring model* for cluster optimization.

## 2.2 Local to Global

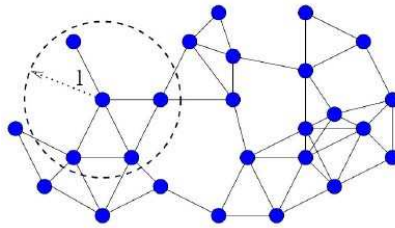
The clusters which share three or more vertices are joined and the local coordinates are transformed to global by suitable rotation translation and reflection as needed. Since in sensor networks most of the times, wrong location information is worse than no location information, the nodes that are not part of robust quadrilaterals are not assigned any coordinates.

## 2.3 Final Remarks

The algorithm does not perform well for sparse networks. Even if there is no noise in distance measurement, the experiments show that average degree of the node should be greater than ten for 100 percent localization.

## 3 Unit Disk Graph embedding is NP Hard

[1] shows that the Unit Disk Graph (UDG) embedding is NP hard. Before we describe the proof of hardness we explain a few basic terms.

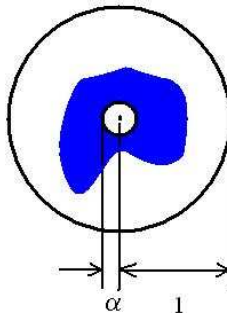


**Figure 5:** Unit Disk Graph : The nodes are connected if and only if they are less than unit distance apart

### 3.1 Basic Definitions

A *Unit Disk Graph* as shown in fig 5 is one in which the nodes are connected to each other if and only if the Euclidian distance between them is less than or equal to one unit, else they remain unconnected. This is a *theoretical model* of the connectivity in a sensor network.

A more practical version, *Quasi Unit Disk Graph* as shown in figure 6 associates a parameter  $\alpha$  which is strictly less than one with the connectivity such that two nodes which are less than  $\alpha$  units apart must be connected and the nodes that are more than 1 unit apart must not be connected. The connectivity information for distances between  $\alpha$  and 1 is unclear.



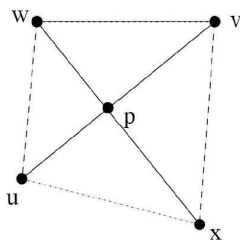
**Figure 6:** Quasi Unit Disk Graph : The nodes must be connected if they are less than  $\alpha$  distance apart and must be disconnected if more than unit distance apart

An *Embedding* of a planar graph is assignment of coordinates to the graph which satisfies the connectivity according to one of the above two models.

The input of an embedding problem can be just connectivity information or connectivity with either local distances or local angles or both.

**Lemma 1** *If two edges cross in a UDG as shown in figure 7, then one node has edges to the three other nodes in the UDG. This is called Crossing Lemma.*

$$\begin{aligned} |uw| &\leq |wp| + |up| \\ |vx| &\leq |vp| + |xp| \end{aligned}$$



**Figure 7:** Example to illustrate crossing lemma

Hence  $|wu| + |vx| \leq |wx| + |uv| \leq 2$   
 Also,  $|wv| + |ux| \leq |wx| + |uv| \leq 2$

Thus there must be 2 edges on the quadrilateral adjacent to the same node.

### 3.2 A few hardness results for UDG embedding

With only *connectivity* information both UDG and Quasi UDG( $\alpha > \sqrt{2/3}$ ) are NP hard. Also with only *local angle* information both UDG and Quasi UDG( $\alpha > 1/\sqrt{2}$ ) are NP hard. If only *local distance* information is available UDG embedding is NP hard but in a special case where the graph has  $\Omega(n^2)$  edges the embedding solution can be found by semidefinite programming.

It should be easy to see that with both *local distance* and *local angle* information the problem is easy to solve. (just start with any vertex, construct the star formed and continue building the graph in an incremental way)

### 3.3 Reduction from 3-SAT

A 3-SAT instance is made up of conjunction of clauses where each clause is a disjunction of at most 3 variables. A special instance of 3-SAT where each variable can take part in at most 3 clauses is also shown to be hard.

[1] shows that UDG embedding is hard by reduction from this special instance of 3-SAT.

#### 3.3.1 Graph Representation of a 3-SAT

As shown in figure 8, the 3-SAT instance is embedded on a grid where there is an edge between a literal and a clause if the literal appears in that clause.

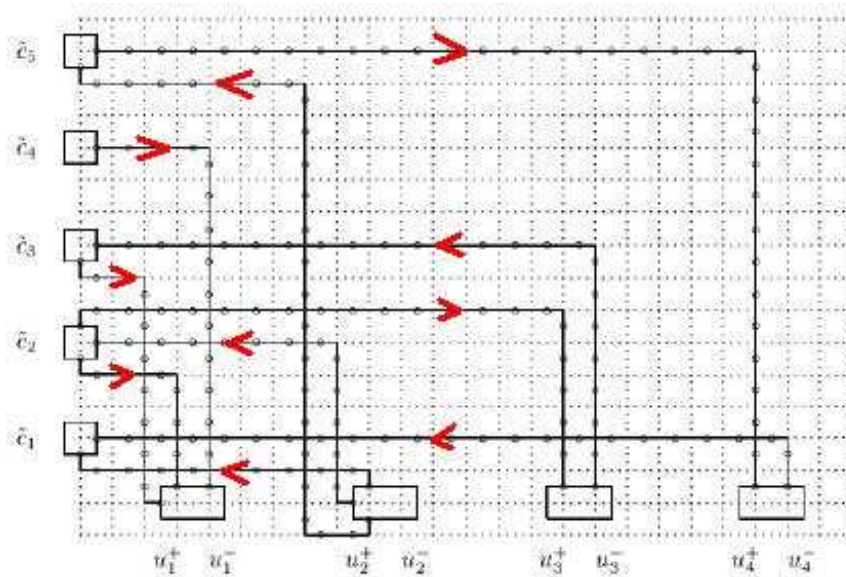
The edge is *directed* from a clause  $c$  to a literal  $u$  if  $c$  chooses  $u$  to satisfy it. Also if a literal has incoming edges then its negated version has outgoing edges. Finally in order for the 3-SAT instance to be satisfied, every clause has at least one outgoing edge.

**Claim 2** A 3-SAT graph is orientable if and only if the 3-SAT instance is satisfiable.

The idea is to realize the 3-SAT graph by UDG such that a valid embedding in 2D gives a valid orientation of the edges.

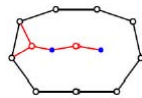
#### 3.3.2 Cage with Beads

The idea of cage with beads is used to represent the orientation. A cage as shown in figure 9, is a closed chain of 2-connected vertices hence by crossing lemma there is no alterante realization of the cage and also the chain of beads



**Figure 8:** Graph representation of 3-SAT instance. Horizontal Axis : Variables, Vertical Axis : Clauses

is completely inside or outside the cage. The chain of beads can be thought as a small arrow indicating the orientation.



**Figure 9:** A Cage with a chain of Beads inside it

Since the beads are independent vertices they must be at least unit distance apart. Also each bead should be at least unit distance away from the cage vertices, hence by packing argument as shown in figure 10 you can not put too many beads inside a fixed size cage.

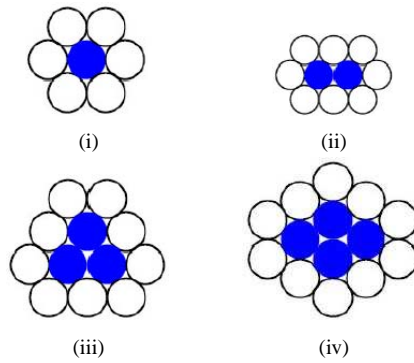
### 3.3.3 Constructing the 3-SAT graph using chains of cages

Figure 11 shows how a few of the building blocks can be made using the cage with beads. In (i) all the cages must have the same orientation because of the packing argument since no two beads can be inside one cage. (ii) represents an oriented path that turns while (iii) shows how a cross-over can be constructed. These building blocks are used to build the 3-SAT graph.

Figure 12 shows how a single *variable* is constructed. There are three edges for both the variable and its compliment. The reader should convince himself that if any of the edges of a variable are incoming then the all the edges of its compliment must be outgoing and vice versa.

Figure 13 shows how a single *clause* with two or three variables is constructed. In both, atleast one of the edges must be an outgoing edge thus a valid embedding will satisfy the clause.

Finally, these clauses and variables are connected according to the given 3-SAT problem and a UDG embedding will result in defining the orientations in the 3-SAT graph and thus solving 3-SAT. But since 3-SAT is NP-hard, UDG



**Figure 10:** (i) 6-vertex cage (ii) 8-vertex cage (iii) 9-vertex cage (iv) 10-vertex cage .

embedding is NP-hard.

## References

- [1] H. Brey and D. G. Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry, Theory and Applications*, 9(1-2):3–24, 1998.
- [2] D. Rus D. Moore, J. Leonard and S. Teller. Robust distributed network localization with noisy range measurements. *Proc. ACM SenSys*, 2004.



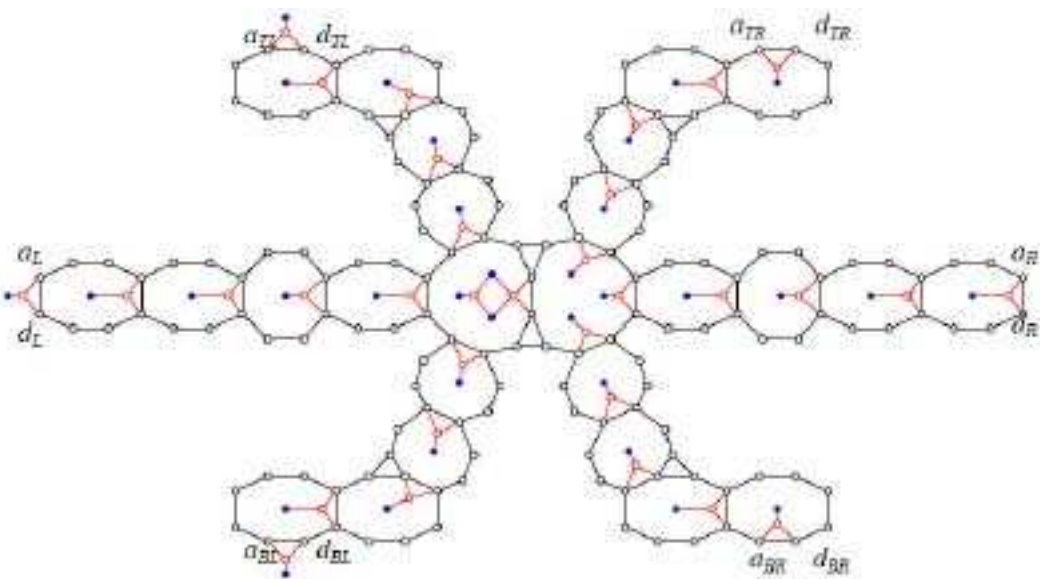


Figure 12: 3-SAT variable construction using cage and beads

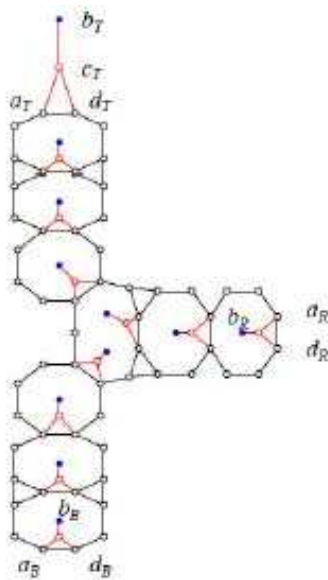


Figure 13: 3-SAT clause construction using cage and beads