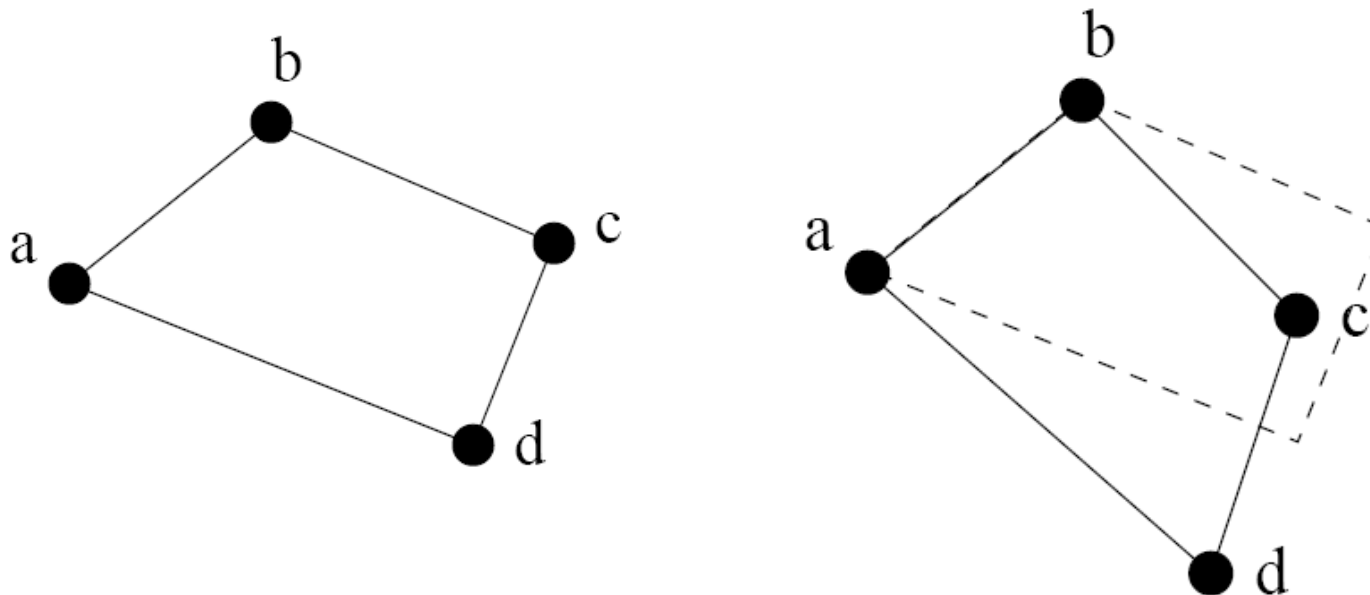

Localization of Sensor Networks II

Jie Gao
Computer Science Department
Stony Brook University

Rigidity theory

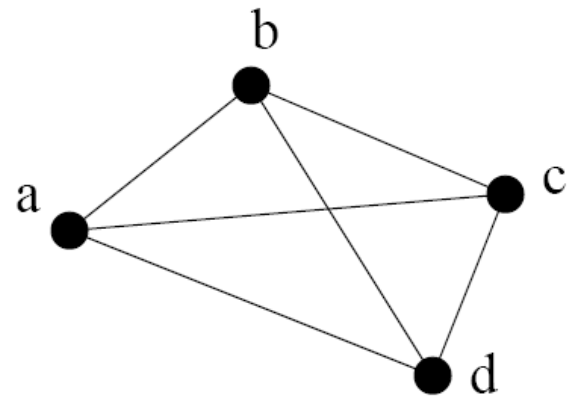
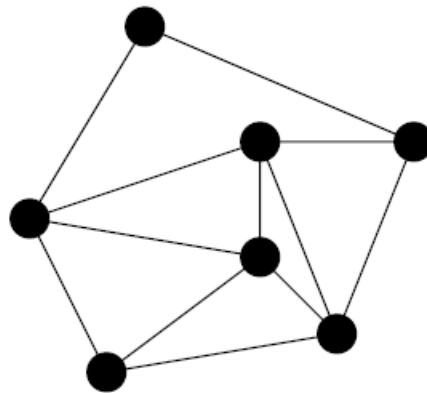
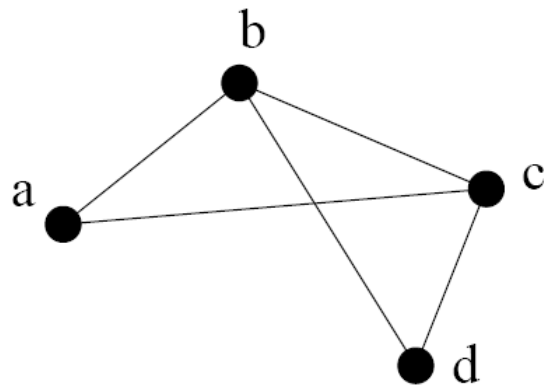
- Given a set of rigid bars connected by hinges, rigidity theory studies whether you can move them continuously.



Laman condition

Laman graph: it has $2n-3$ edges and no subgraph of k vertices has more than $2k-3$ edges.

Laman condition: A graph is rigid if it contains a Laman graph.



Algorithm to test rigidity

- Laman's condition taken literally leads to poor algorithm, as it involves checking all subgraphs!
- Efficient and intuitive algorithm exists, based on counting degrees of freedom.

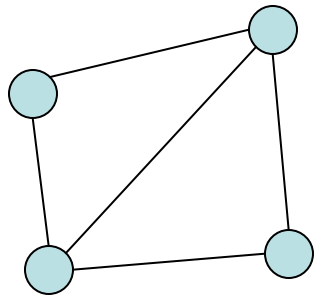
Pebble game – test graph rigidity

- D. J. Jacobs and B. Hendrickson, [An Algorithm for two dimensional rigidity percolation: The pebble game.](#) J. Comput. Phys., 137:346-365, 1997.

Intuition

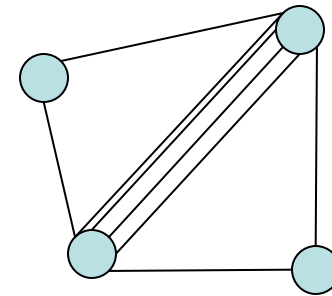
- Find a subset of independent edges.
- Use an incremental algorithm: pick an edge, test if it's independent of the current subset.
- **Alternate Laman theorem:** The edges in G are independent in 2D if and only if for each edge (a, b) , the graph formed by **quadrupling** (a, b) has no induced subgraph of k nodes and $>2k$ edges.

Illustration



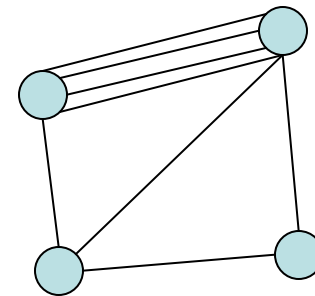
G

“quadruple an edge”



no subgraph with $>2k$ edges

***G* is rigid.**



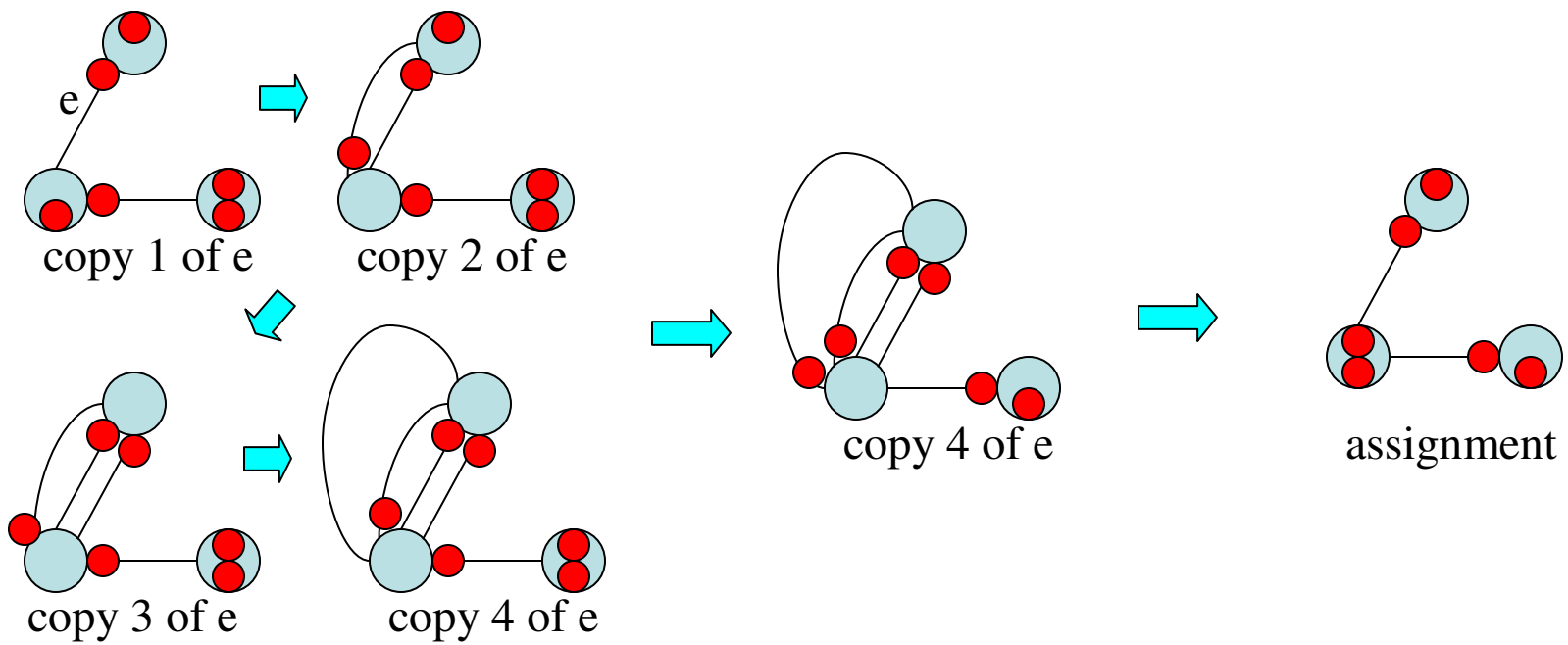
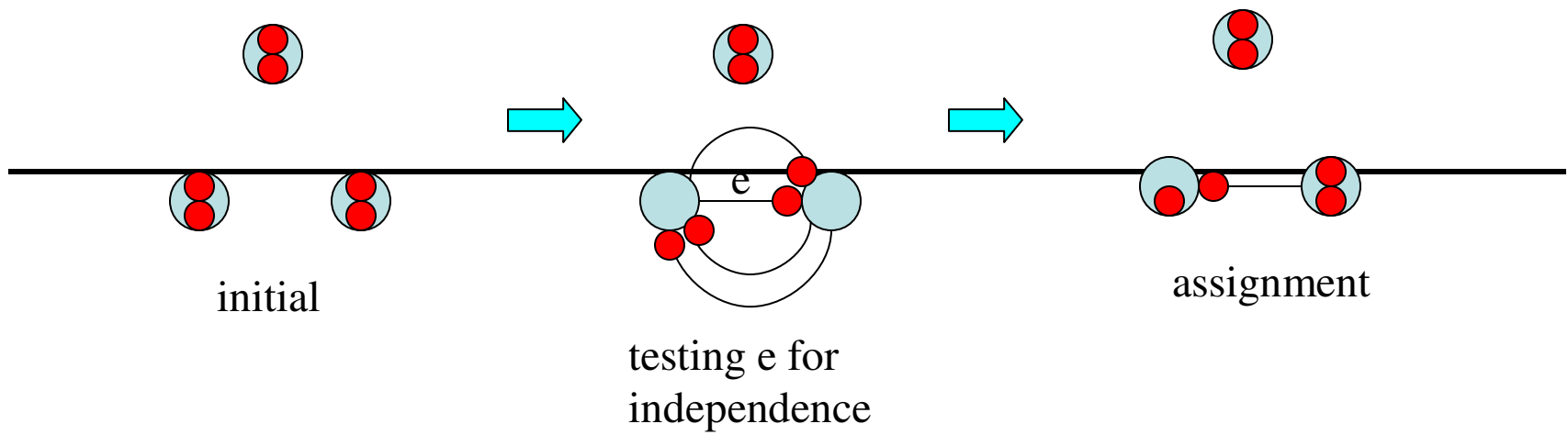
no subgraph with $>2k$ edges

Test an independent edge

- Grow a maximal set of independent edges one at a time.
- New edge is added if it is independent of existing set.
 - Quadruple the new edge and test the Laman condition.
- If Laman failed, then output “not rigid”.
- If $2n-3$ independent edges are found, then graph is rigid.
- How to test Laman condition quickly?

The Pebble Game

- Each node is assigned 2 pebbles \rightarrow $2n$ pebbles in total.
- An edge is **covered** by having one pebble placed on either of its ends
- A **pebble covering** is an assignment of pebbles so that all edges in graph are covered
- Test if a new edge is independent of the existing set: quadruple the edge; find a pebble covering for the 4 new edges.

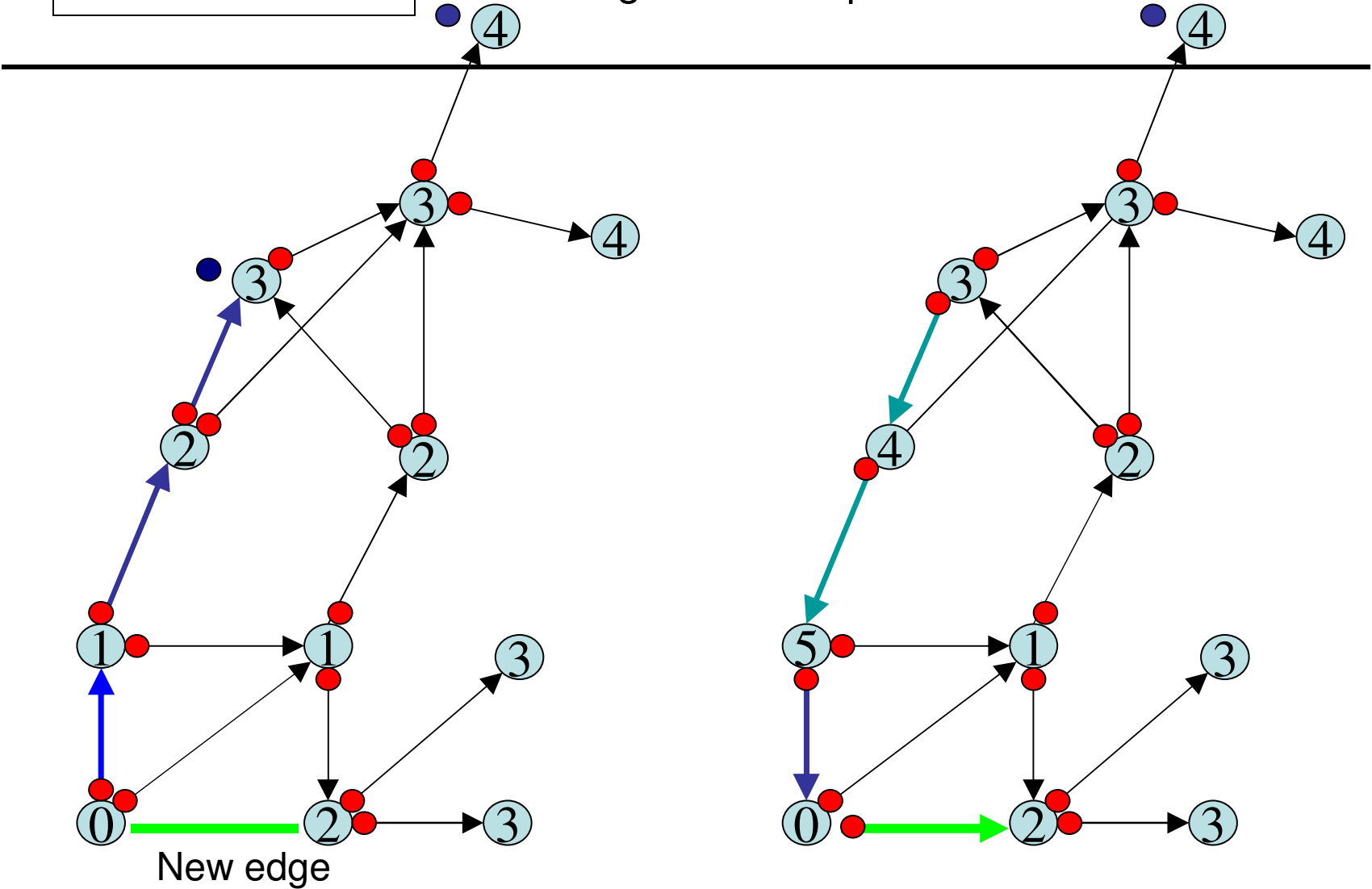


Pebble Game Algorithm

- Assume we have a set of edges covered with pebbles and we want to add a new edge.
- First, look at vertices incident to a new edge.
 - If either has a free pebble, use it to cover the edge and we are done.
 - Otherwise, their pebbles are covering existing edges. If a vertex at other end of one of these edges has a free pebble, then use that pebble to cover existing edge, freeing up pebble to cover new edge.
- Search for free pebbles in a **directed** graph.
 - If edge $e_{a,b}$ is covered by a pebble from vertex a , the edge is directed from a to b .
- Search until a pebble is found, then swap pebbles and reverse the edges until the new edge is covered, else fail.

● unassigned pebble

Do a depth-first search following the directed edges for free pebbles.



Pebble game properties

- Testing an edge for independence takes $O(n)$ time: we do 3 depth-first search in a graph with $O(n)$ edges.
- At most m edges will be tested. The total running time is $O(nm)$.
- Algorithm is amenable to distributed implementation
- Intuitive appeal: Each pebble corresponds to a degree of freedom. A pebble covering always has at least 3 free pebbles.

Additional applications

- Pebble game can also identify redundantly rigid section of the graph.
- Using this tool, along with algorithms for discovering 3-connected components, it is possible to decompose networks into uniquely localizable regions.

Open questions

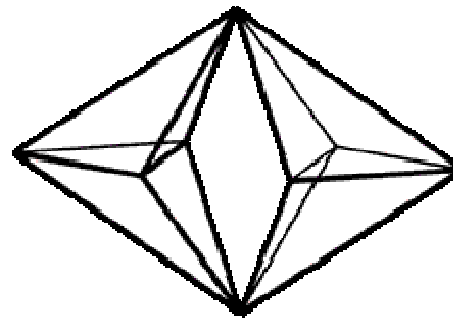
- Find an algorithm with running time better than $O(nm)$?
- Test graph rigidity with edge insertion and deletion, avoid re-run the Pebble Game.

Laman theorem in 3D?

Laman condition in 3D?

A graph is generically minimally rigid in 3D if and only if it has $3n-6$ edges and no subgraph of k vertices has more than $3k-6$ edges?

Double banana!



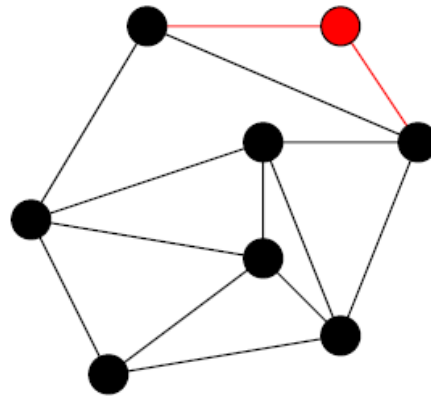
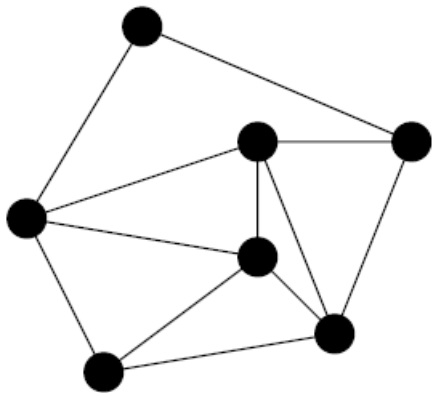
Unfortunately, the condition is necessary but not sufficient.

It's a long open problem what is the **combinatorial** condition for rigidity in 3D.

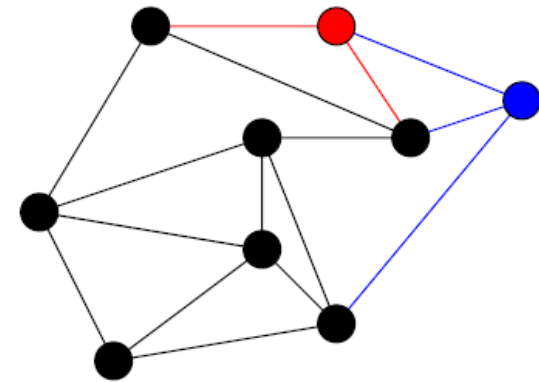
Now, use rigidity theory in
localization algorithms

Recall Henneberg constructions

- Type I step: join the vertex to two old vertices via two edges
- Type II step: join the vertex to three old vertices with at least one edge in between, via three edges. Remove an old edge between the three endpoints.



Type I

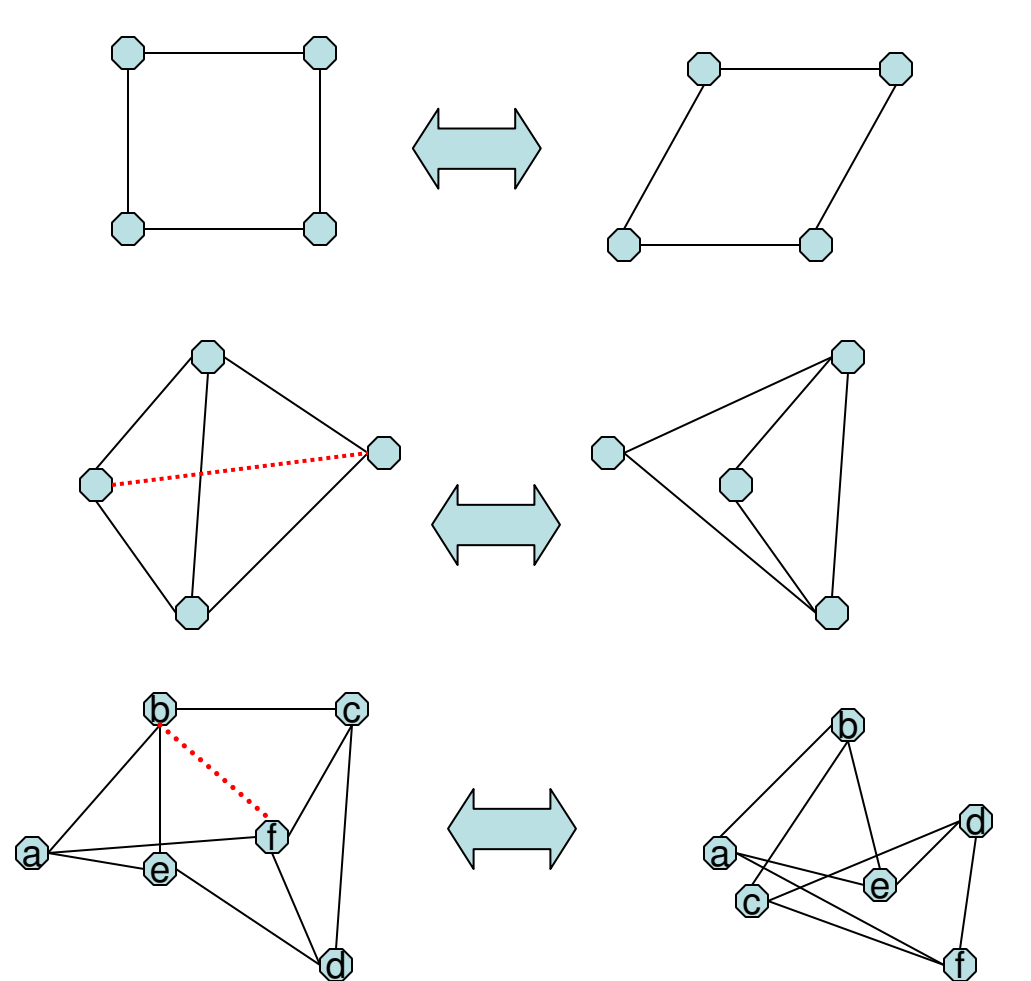


Type II

Hennerberg construction implies...

- The subgraph examined by iterative multilateration is rigid.
 - Start with three nodes (with known locations).
 - Add 1 new node with **3** edges to existing nodes.
- Such a graph is named “trilateration graph”.
- How about global rigidity?

Global rigidity



Solution:

G must be *rigid*

G must be 3-connected, i.e. Connected after removal of 2 Vertices.

G must be *redundantly rigid*: It must remain rigid upon removal of any single edge

Hennerberg construction implies...

- The subgraph examined by iterative multilateration is **globally rigid**.
 - Start with three nodes (with known locations).
 - Add 1 new node with **3** edges to existing nodes.
- Such a graph is named “trilateration graph”.

Localize trilateration graphs

- Find a sequence of nodes such that each node x has edges to 3 nodes already localized.
- Seed triangle: enumerate all possible $\binom{n}{3}$ nodes.
- Use a greedy algorithm to add new nodes.
- Note that trilateration graph has $3n$ edges (much more than $2n-3$).

Required papers

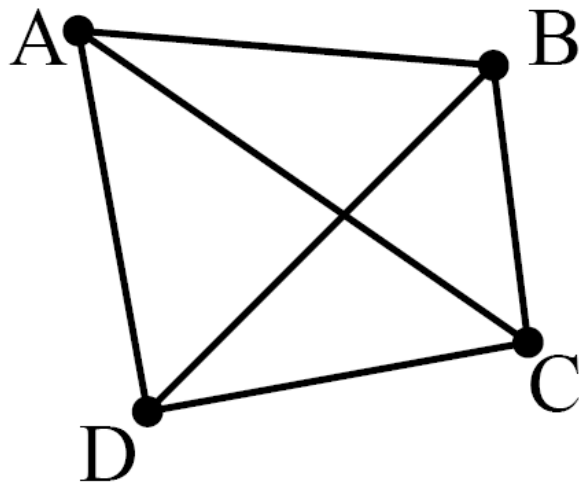
- D. Moore, J. Leonard, D. Rus, S. Teller, [Robust distributed network localization with noisy range measurements](#), Proc. ACM SenSys 2004.
- Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P.J. Fromherz, [Localization from Mere Connectivity](#), MobiHoc'03.

Two approaches

- Local optimization:
 - Avoid flip ambiguity of iterative trilateration.
- Global optimization:
 - multi-dimensional scaling.

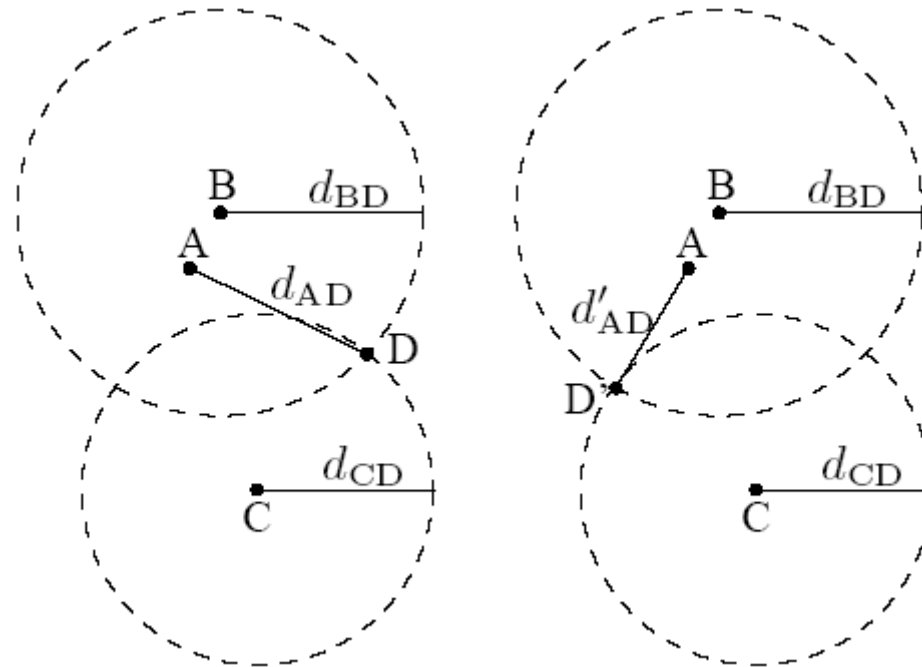
Trilateration is a quadrilateral

- Four nodes with fixed pair-wise distances
- It is the smallest 3-connected redundantly rigid graph \rightarrow globally rigid.



Trilateration with noise...

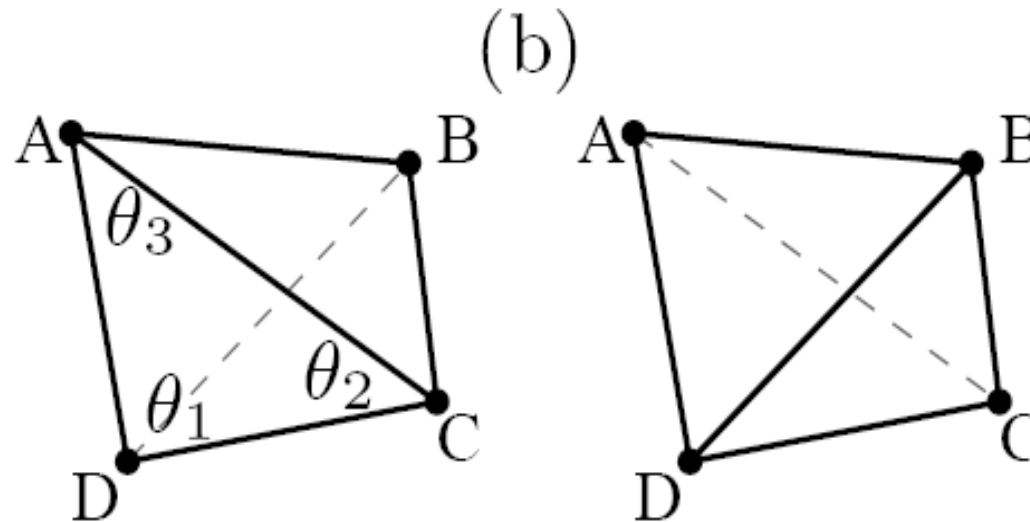
- With noisy measurements, trilateration can have flip ambiguity.



Why flip ambiguity?

- There are 4 triangles in a quadrilateral.

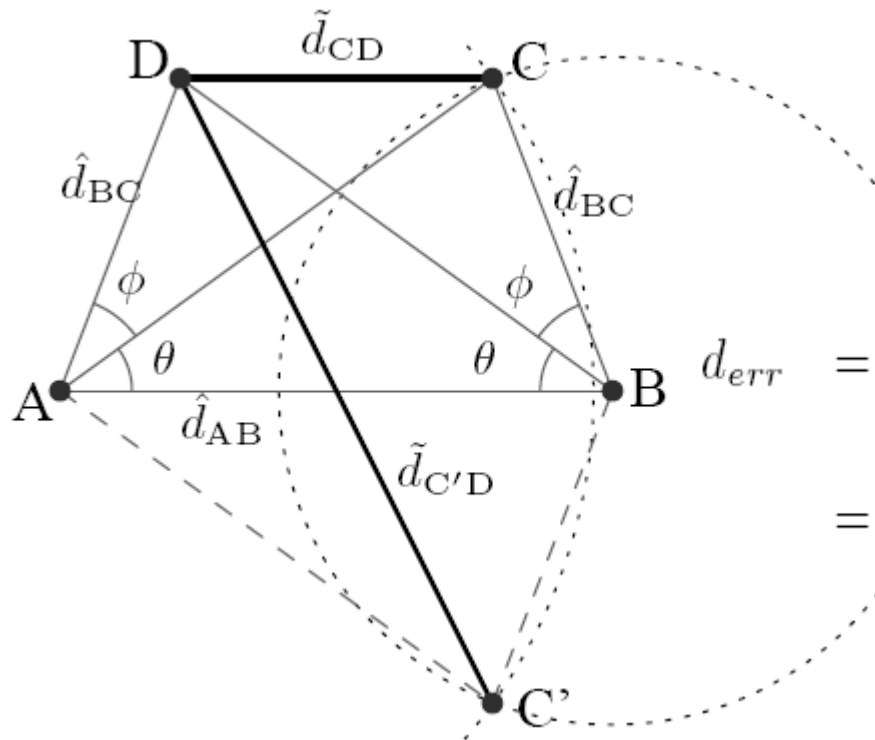
**Flip ambiguity,
when the
smallest angle
is too small!**



- **Assume noise has a normal distribution.**
- **Only accept quadrilaterals with sufficient large min angle!**

Robust quadrilaterals

- Assume we do trilateration to locate C.
- First ignore D, then C and C' are possible locations.
- If CD and C'D are too different, then we can verify which of C, and C' is correct.



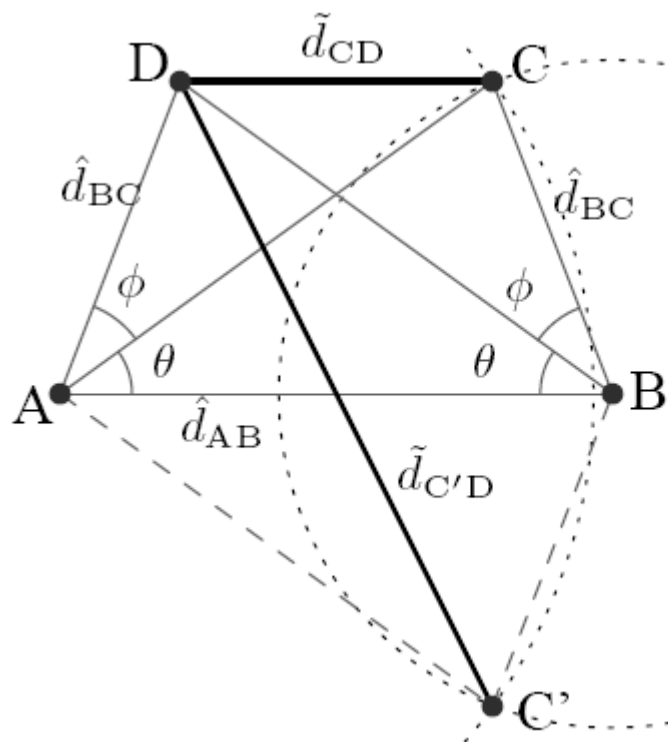
To have ambiguity, the error in measuring CD distance is at least

$$d_{err} = \frac{\tilde{d}_{C'D} - \tilde{d}_{CD}}{2}$$

$$= \hat{d}_{AB} \frac{\sqrt{\sin^2 \phi + 4 \sin^2(\theta + \phi) \sin^2 \theta} - \sin \phi}{2 \sin(2\theta + \phi)}$$

Robust quadrilaterals

- If measurement noise is bounded, the quadrilateral has no incorrect flip.



Take derivative to minimize the error: when $\phi = \pi/2 - 2\theta$.

$$d_{err} = \hat{d}_{AB} \sin^2 \theta$$

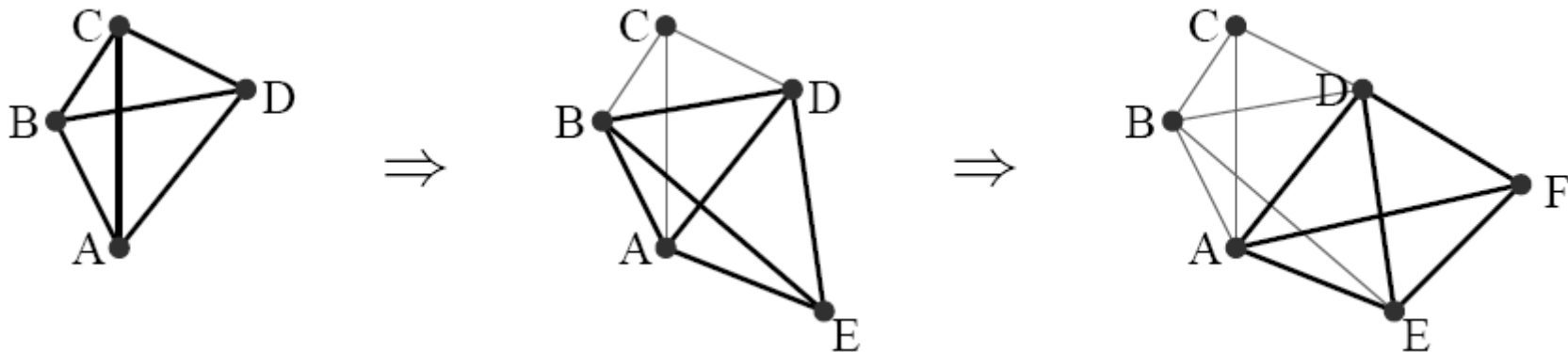
Robust quadrilateral satisfies

$$d_{err} < b \sin^2 \theta$$

Where b is the shortest side and θ is the smallest angle.

Clusters

- Robust quads that share three nodes can be merged into clusters.
- The cluster is still a 3-connected redundantly rigid graph \rightarrow globally rigid.
- Actually it has $3n-6$ edges.

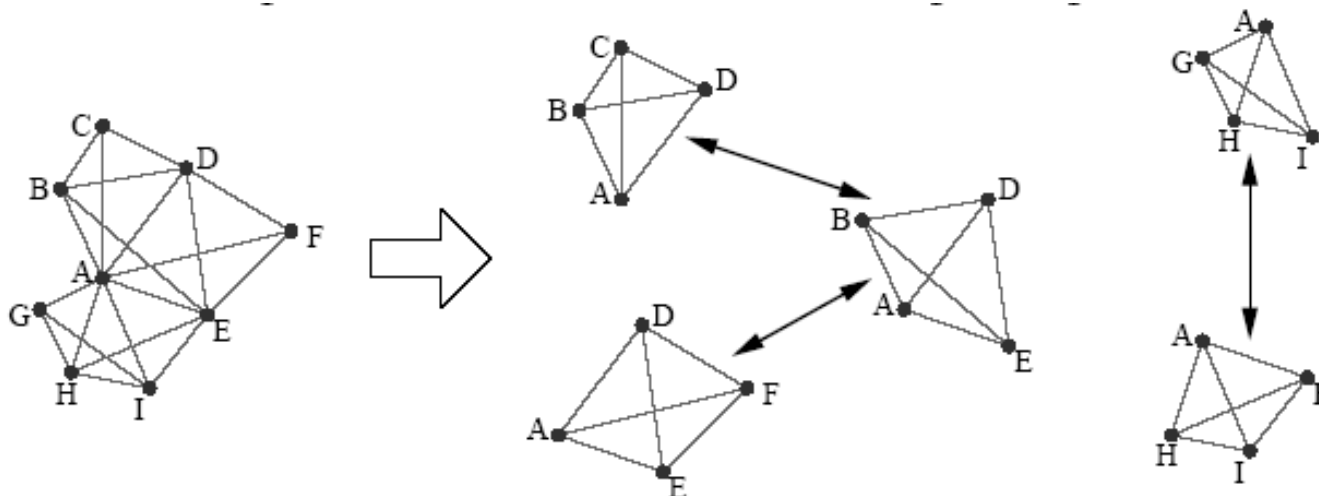


The algorithm

- Cluster localization
 - Each node x finds its local robust quadrilaterals.
 - Merge them to a robust cluster around x .
- (optional) cluster optimization
 - Refine the nodes' location inside a cluster.
- Cluster transformation
 - Glue the local quadrilaterals together
 - Transformation to a global coordinate system.

Algorithm phase I: cluster localization

1. Each node x gets the distance measurements between each pair of 1-hop neighbors.
 2. Identify the set of robust quadrilaterals.
 3. Merge the quads if they share 3 nodes.
 4. Estimate the positions of as many nodes as possible by iterative trilateration.
- Note: **Local coordinate system** rooted at x .

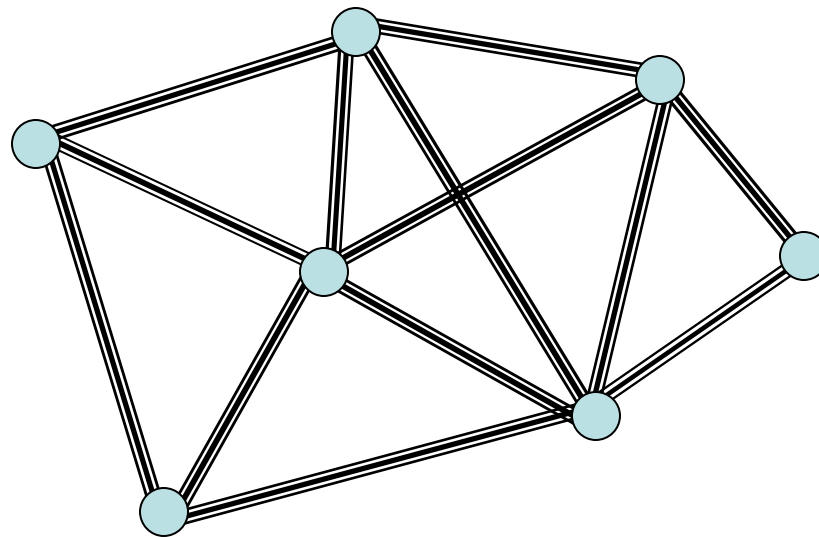


Algorithm phase II: cluster optimization

- For each cluster around node x , refine the position estimates, for example, by mass-spring relaxation.
- Optional.
- Reduce accumulated error.

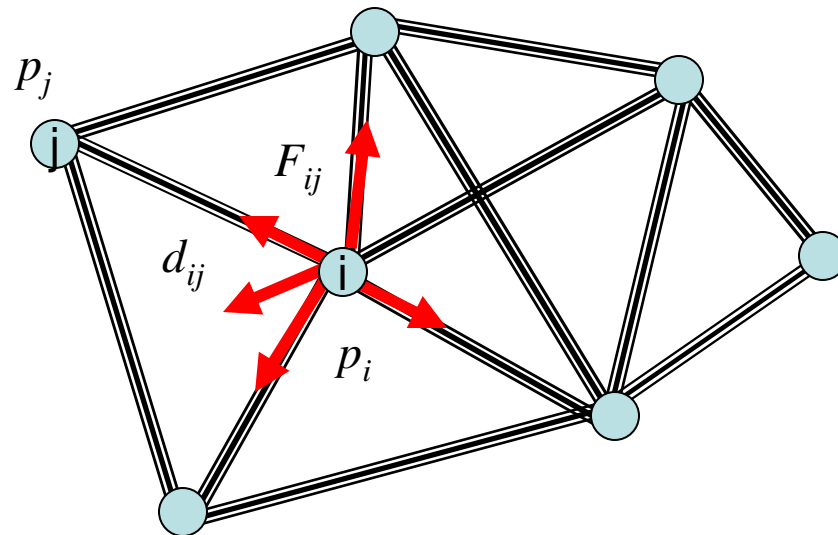
Mass-spring system

- Nodes are “masses”, edges are “springs”.
- Length of the spring equals the distance measurement.
- Springs put forces to the nodes.
- Nodes move.
- Until the system stabilizes.



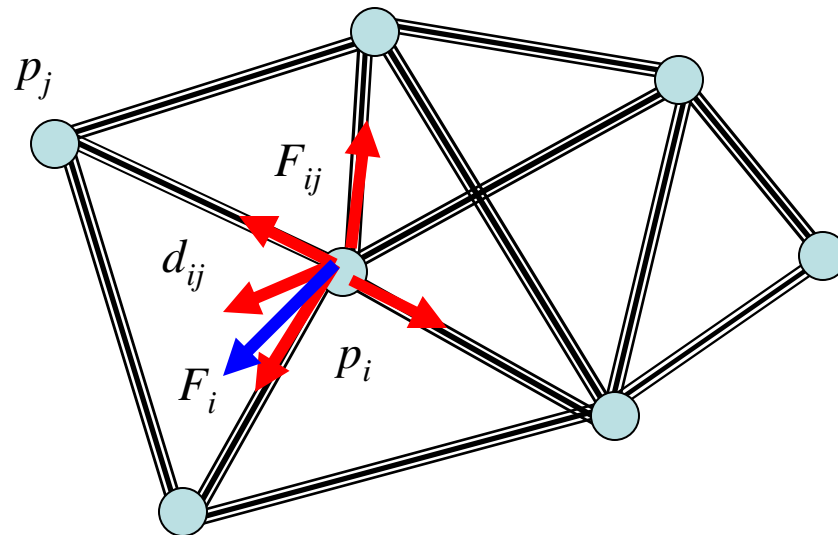
Mass-spring system

- Node n_i 's current estimate of its position: p_i .
- The estimated distance d_{ij} between n_i and n_j .
- The measured distance r_{ij} between n_i and n_j .
- Force: $F_{ij} = d_{ij} - r_{ij}$, along the direction $p_i p_j$.



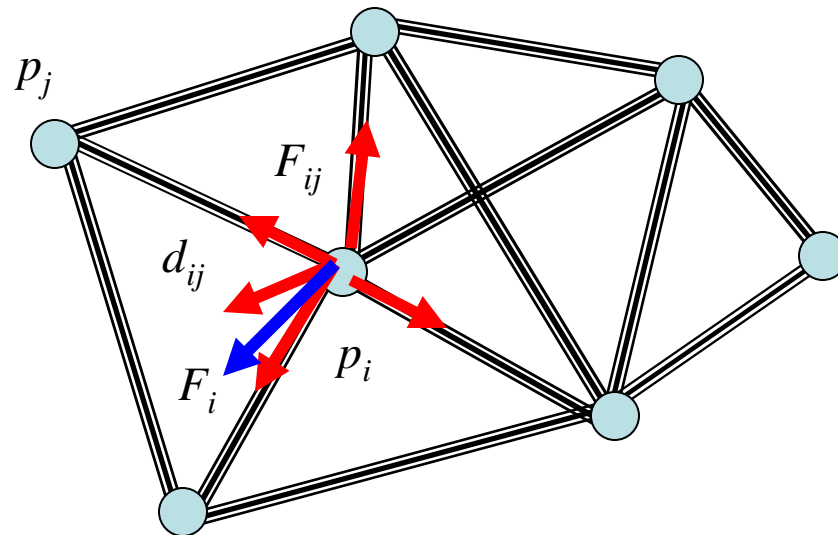
Mass-spring system

- Total force on n_i : $F_i = \sum F_{ij}$.
- Move the node n_i by a small distance (proportional to F_i).
- Recurse.



Mass-spring system

- Total energy n_i : $E_i = \sum E_{ij} = \sum (d_{ij} - r_{ij})^2$.
- Make sure that the total energy $E = \sum E_i$ goes down.
- Stop when the force (or total energy) is small enough.



Mass-spring system

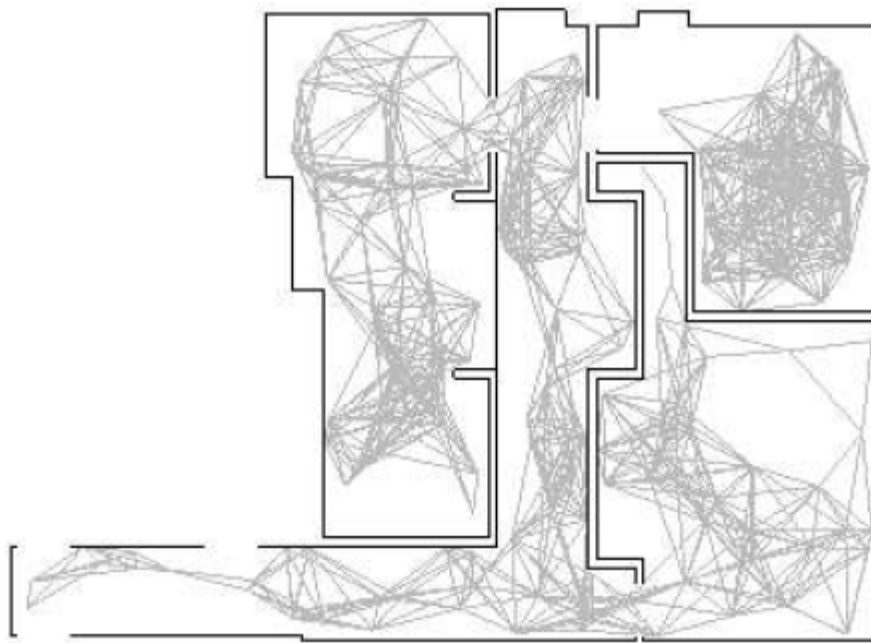
- Advantage: Naturally a distributed algorithm.
- Problem 1: may stuck in local minima.
 - Need to start from a reasonably good initial estimation, e.g., the iterative multi-lateration.
 - Typically not used alone.
- Problem 2: not robust to outliers.
 - If one measurement is off too much, the error gets distributed everywhere in the system.
 - **A possible class project to deal with outliers.**

Algorithm phase III: cluster transformation

- Align neighboring local coordinates systems.
- Find the set of nodes in common between two clusters.
- Compute the translation, rotation that best align them.

Simulations

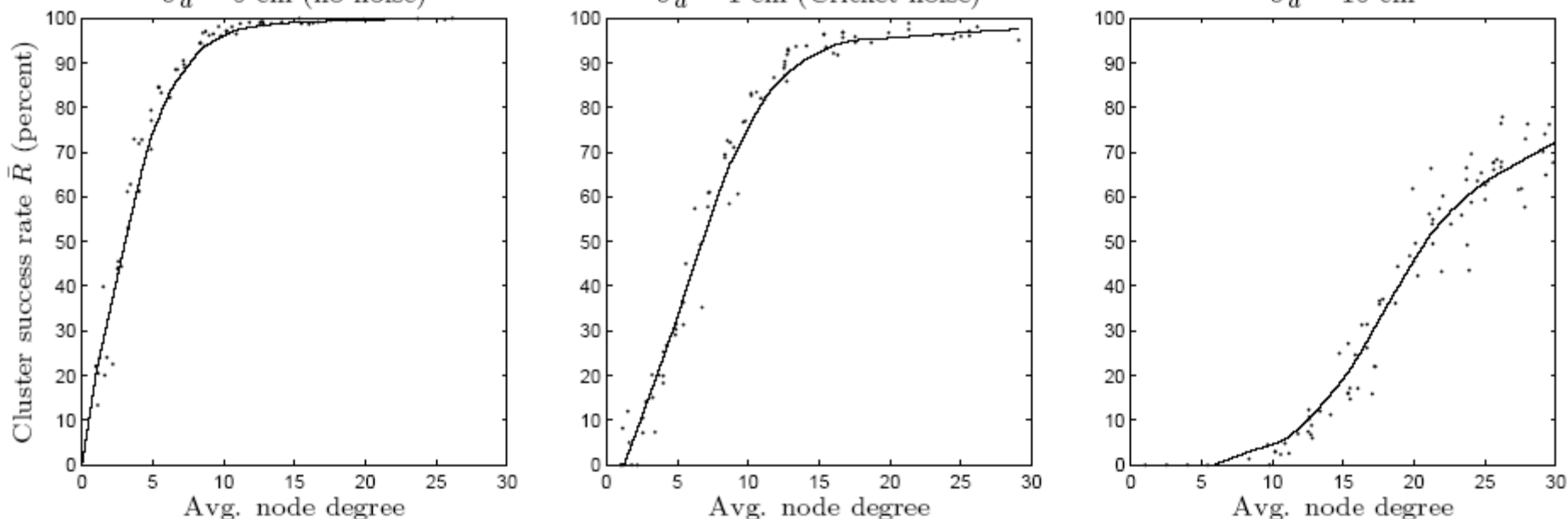
- 183 nodes uniformly inside a building.
- Connectivity is only between nodes not obstructed by walls.



Simulations

- Cluster success rate v.s. node degree.
- Each plot represents a simulation run.

(a) PLOTS OF CLUSTER SUCCESS RATE, \bar{R} , VERSUS NODE DEGREE FOR THE BUILDING ENVIRONMENT
 $\sigma_d = 0$ cm (no noise) $\sigma_d = 1$ cm (Cricket noise) $\sigma_d = 10$ cm



Algorithm properties

- Nodes not included in the robust quadrilaterals are not localized.
 - A wrong location is worse than no location.
- Even as noise goes to 0, avg degree ≥ 15 to achieve 100% localization.
- Not good for sparse networks.
- The avg degree $\cong 6$ for best throughput of the network.

Observations

- Localization algorithm performs poorly when the graph is **sparse**.
- Negative constraints are not effectively used --- non-neighbors should be sufficiently far.
- Localization is a difficult problem: Unit disk graph embedding is NP-hard.
 - Given measurements of edges of a unit disk graph, it is NP-hard to find a realization in the plane.

Paper presentation

- **[Lederer08]** Sol Lederer, Yue Wang, Jie Gao, [Connectivity-based Localization of Large Scale Sensor Networks with Complex Shape](#), INFOCOM'08.
- **[Lee08]** HyungJune Lee, Martin Wicke, Branislav Kusy, and Leonidas Guibas, [Localization of Mobile Users Using Trajectory Matching](#), ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments, MELT 2008.
- **[Zhong07]** Ziguo Zhong, Tian He, [MSP: Multi-Sequence Positioning of Wireless Sensor Nodes](#), Sensys07.

Paper presentation

- 2/10 in class
- 20 minutes
- The audience will evaluate the presentation.
 - Clarity
 - Enthusiasm
 - Engage the audience
 - The score will not be taken into final grades.