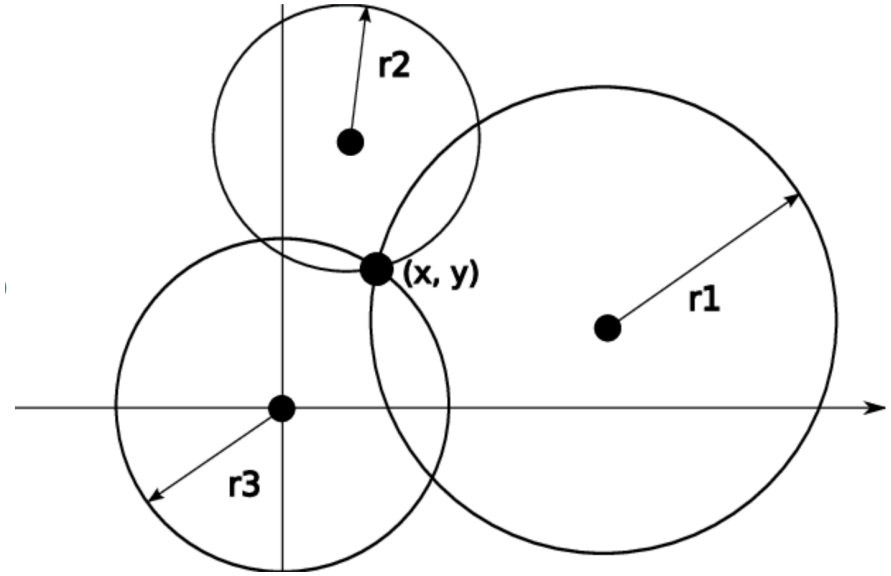
LOCALIZATION

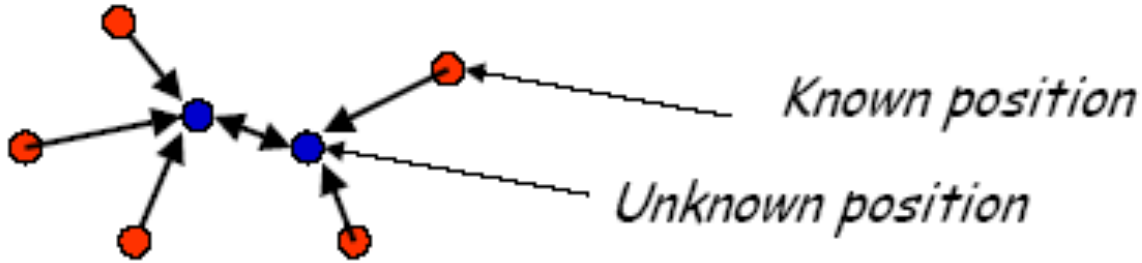
COOPERATIVE MULTILATERATION

COLLABORATIVE MULTILATERATION: JOINT OPTIMIZATION



The Trilateration algorithm.

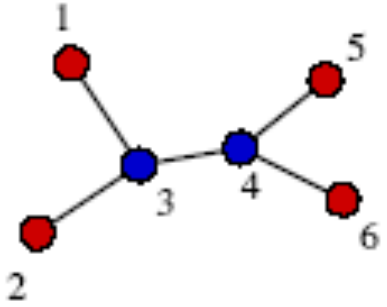
- All available measurements are used as constraints



- Solve for the positions of multiple unknowns simultaneously
- Joint optimization can get better results compared with separate optimizations.

KALMAN FILTER FORMULATION

$$f_{2,3} = r_{2,3} - \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}$$
$$f_{3,5} = r_{3,5} - \sqrt{(x_3 - x_5)^2 + (y_3 - y_5)^2}$$
$$f_{4,3} = r_{4,3} - \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2}$$
$$f_{4,5} = r_{4,5} - \sqrt{(x_4 - x_5)^2 + (y_4 - y_5)^2}$$
$$f_{4,1} = r_{4,1} - \sqrt{(x_4 - x_1)^2 + (y_4 - y_1)^2}$$



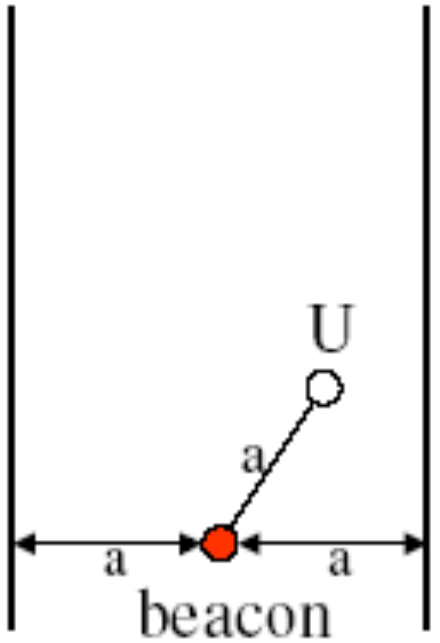
The objective function is

$$F(x_3, y_3, x_4, y_4) = \min \sum f_{i,j}^2$$

Start from some initial estimates, then use a Kalman Filter.

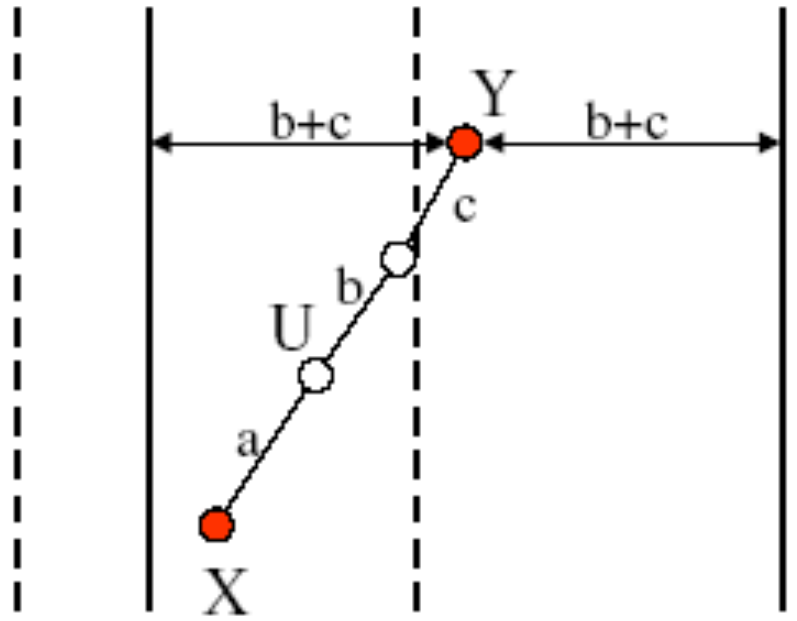
INITIAL ESTIMATES

- Use the distance to a beacon as bounds on the x and y coordinates



INITIAL ESTIMATES CONT

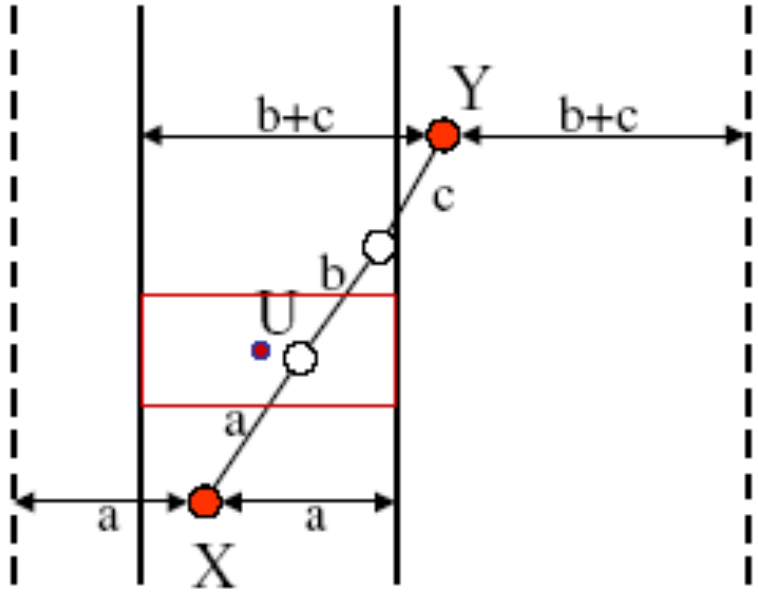
- Use the distance to a beacon as bounds on the x and y coordinates
- Do the same for beacons that are multiple hops away
- Select the most constraining bounds



U is between [Y-(b+c)] and [X+a]

INITIAL ESTIMATES

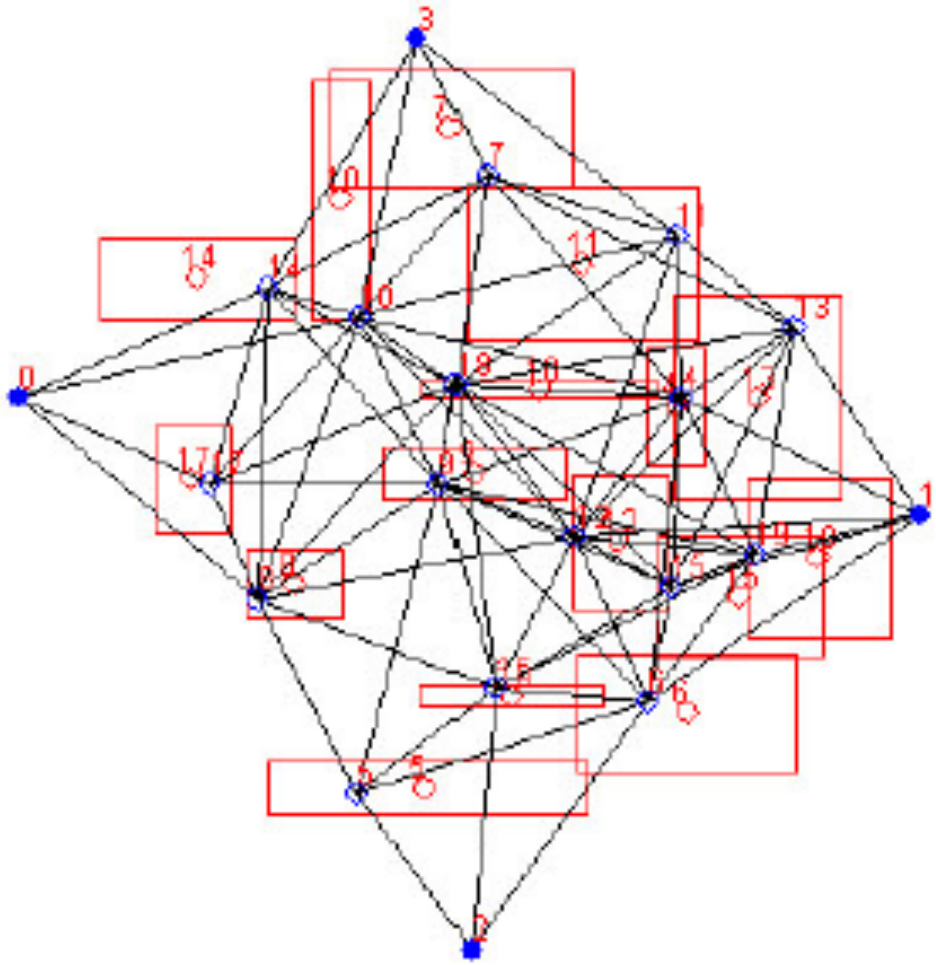
- Use the distance to a beacon as bounds on the x and y coordinates
- Do the same for beacons that are multiple hops away
- Select the most constraining bounds
- **Set the center of the bounding box as the initial estimate**



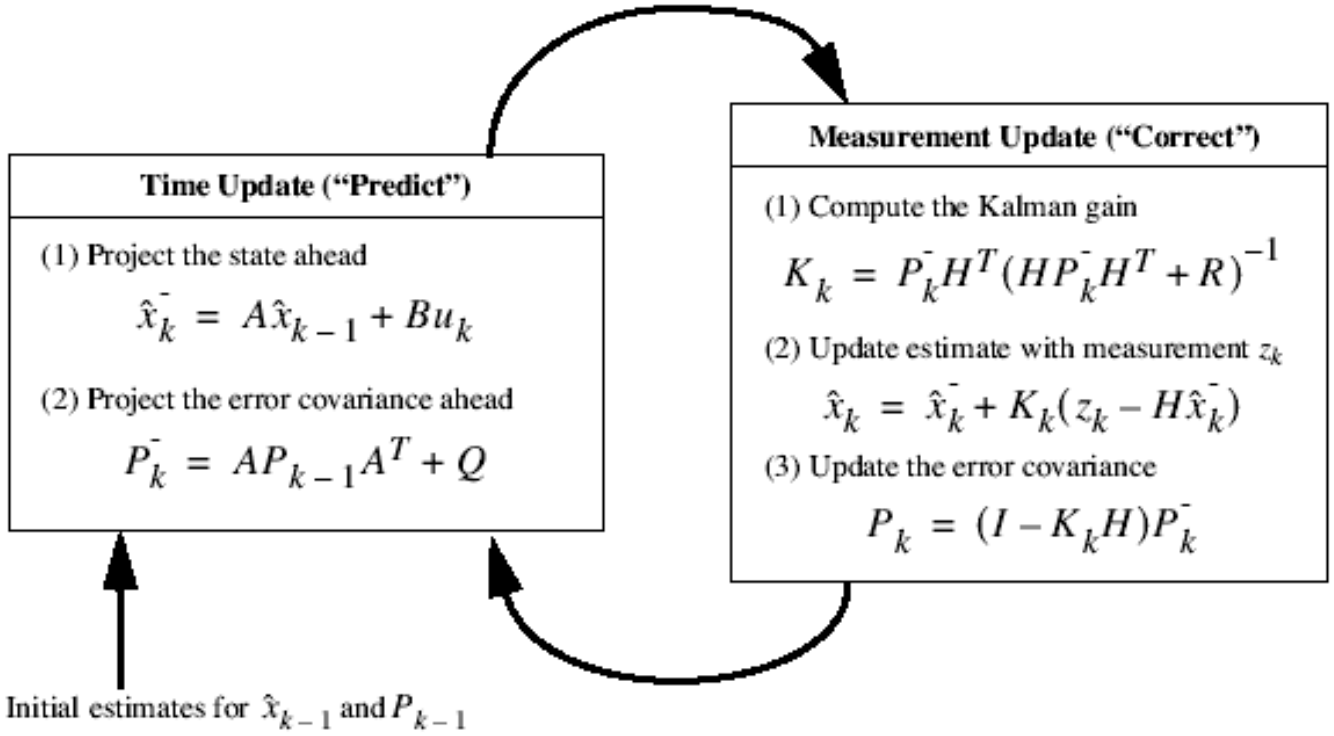
INITIAL ESTIMATION

- Initial estimates give rough location information.

- Use Kalman Filter to refine.
  - Start with prior info.
  - Incorporate new measurement info.
  - Improve the current state.
  - Details omitted.



KALMAN FILTER EQUATIONS



- We only use measurement update since the nodes are static
- We know R (ranging noise distribution)

GLOBAL KALMAN FILTER

$$\hat{z}_k^{\,T} = \begin{bmatrix} \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ \sqrt{(ex_3 - ex_4)^2 + (ey_3 - ey_4)^2} \\ \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \\ \sqrt{(ex_4 - x_5)^2 + (ey_4 - y_5)^2} \end{bmatrix}$$

$$H = \underbrace{\begin{bmatrix} 0 & 0 & \frac{x_2 - ex_3}{\hat{z}_k(1)} & \frac{y_2 - ey_3}{\hat{z}_k(1)} \\ \frac{ex_3 - x_5}{\hat{z}_k(2)} & \frac{ey_3 - y_5}{\hat{z}_k(2)} & 0 & 0 \\ \frac{ex_3 - ex_4}{\hat{z}_k(3)} & \frac{ey_3 - ey_4}{\hat{z}_k(3)} & \frac{ex_4 - ex_3}{\hat{z}_k(3)} & \frac{ey_4 - ey_3}{\hat{z}_k(3)} \\ \frac{ex_4 - x}{\hat{z}_k(4)} & \frac{ey_4 - y}{\hat{z}_k(4)} & 0 & 0 \\ \frac{ex_4 - x_5}{\hat{z}_k(5)} & \frac{ey_4 - y_5}{\hat{z}_k(5)} & 0 & 0 \end{bmatrix}}_{\text{\# of unknown nodes x 2}} \Bigg\} \text{\# of edges}$$

Matrices grow with density and number of nodes => so does computation cost

Computation is not feasible on small processors with limited computation and memory

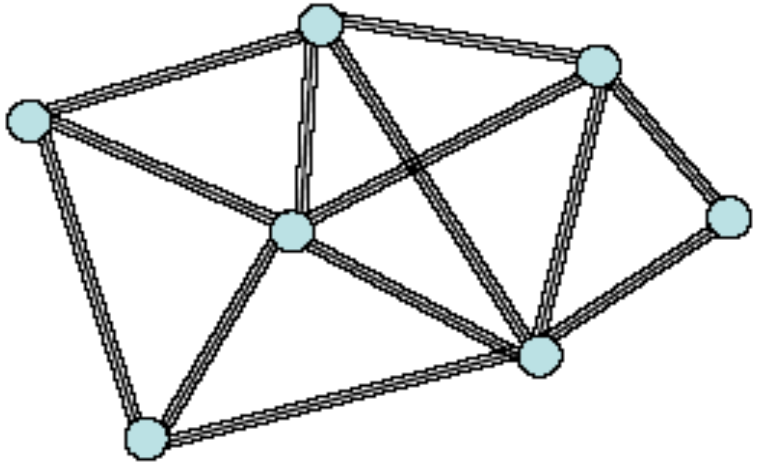
ML

- Need beacons.
- Iterative multi-lateration.
  - Error accumulates.
  - May get stuck when the density is low.
- Collaborative multi-lateration.
  - Still requires a large number of beacon nodes, especially when the network is sparse.
  - Kalman filter computation is expensive on large networks.

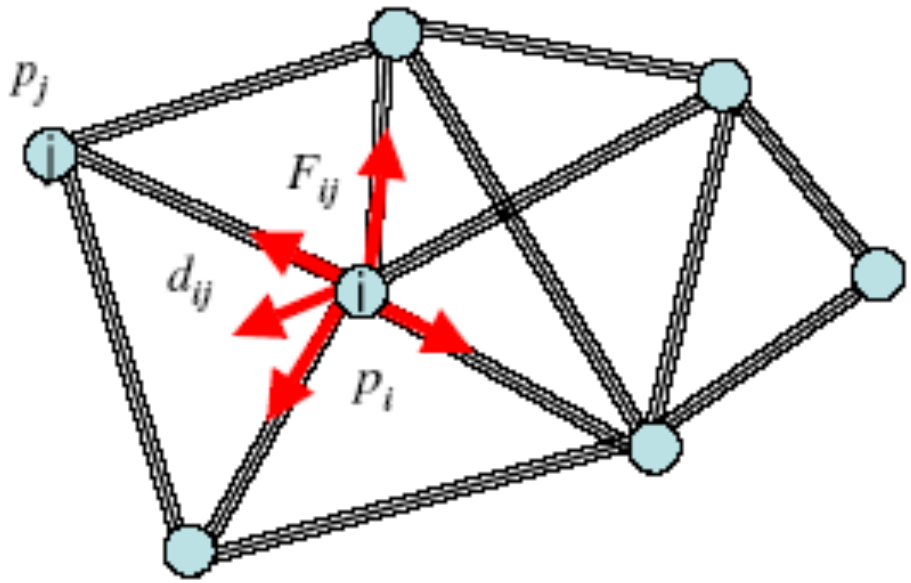
MASS-SPRING LOCALIZATION

Improve the accumulated localization error by a global iterative algorithm ---

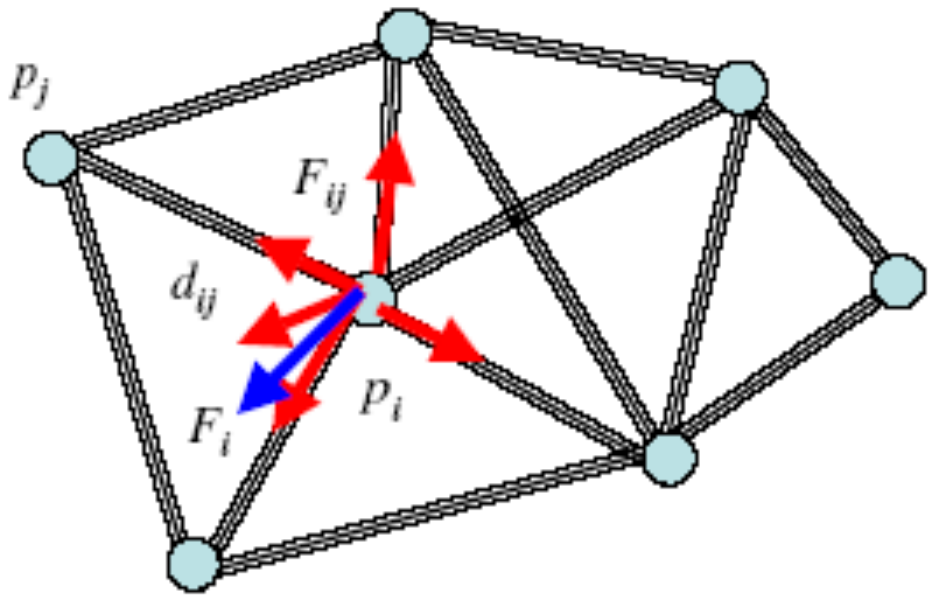
- Nodes are "masses", edges are "springs".
- Length of the spring equals the distance measurement.
- Springs put forces to the nodes.
- Nodes move.
- Until the system stabilizes.



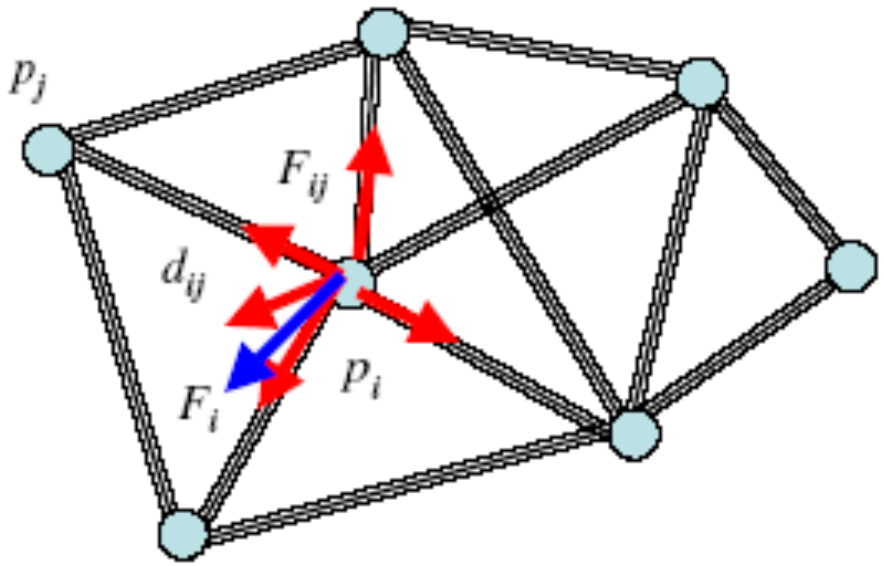
- Node n_i's current estimate of its position: p_i.
- The estimated distance d_{ij} between n_i and n_j.
- The measured distance r_{ij} between n_i and n_j.
- Force: $F_{ij}=d_{ij}- r_{ij}$, along the direction p_ip_j.



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

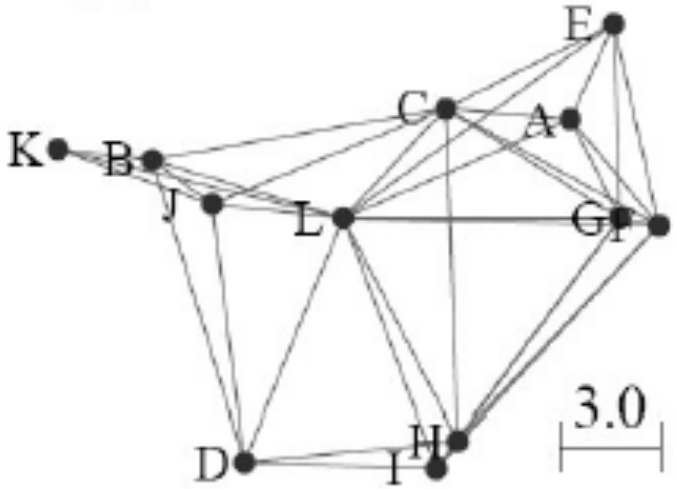


- Total energy n_i: $E_i = \Sigma E_{ij} = \Sigma (d_{ij} - r_{ij})^2$.
- Make sure that the total energy $E = \Sigma E_i$ goes down.
- Stop when the force (or total energy) is small enough.



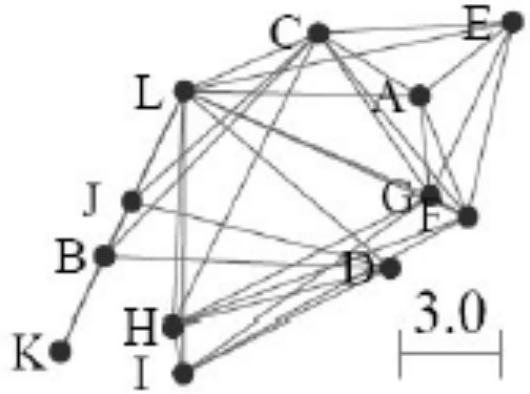
- Naturally a distributed algorithm.
- Problem: may stuck in local minima.
- Need to start from a reasonably good initial estimation, e.g., the iterative multi-lateration.

(a) Ground truth



$\sigma_{err} = 0.37$

(b) Alternate realization



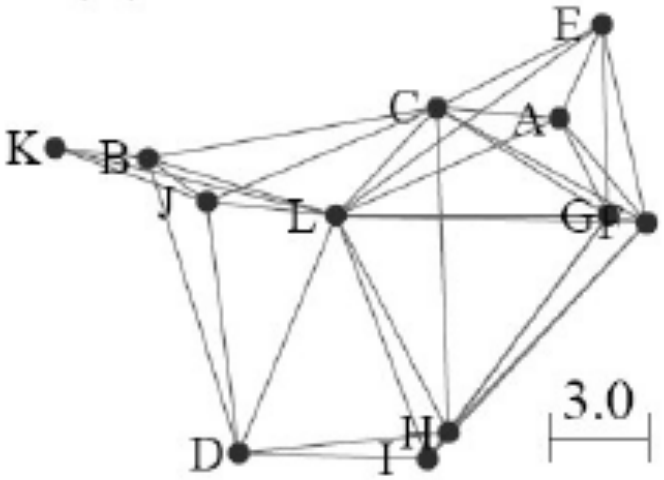
$\sigma_{err} = 0.34$

AMBIGUITY IN LOCALIZATION

For noisy measurements, we use optimization methods…Yet optimization does not solve ---

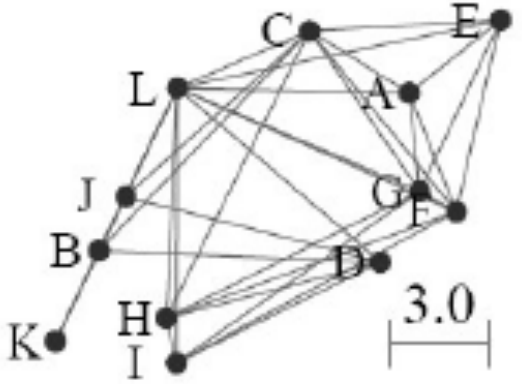
- Same distances, different realization.

(a) Ground truth



$\sigma_{err} = 0.37$

(b) Alternate realization



$\sigma_{err} = 0.34$

RANGE-FREE LOCALIZATION METHODS DV-HOP

RANGE-FREE LOCALIZATION

Range−free localization techniques does not rely on distance estimation using ranging techniques, instead it uses connectivity information for estimating the position.

So, these techniques do not require installation of additional hardware on the nodes that makes them cost−effective alternative to the range−based algorithms.

One such technique is called Ad Hoc Positioning System (APS) which is discussed next.

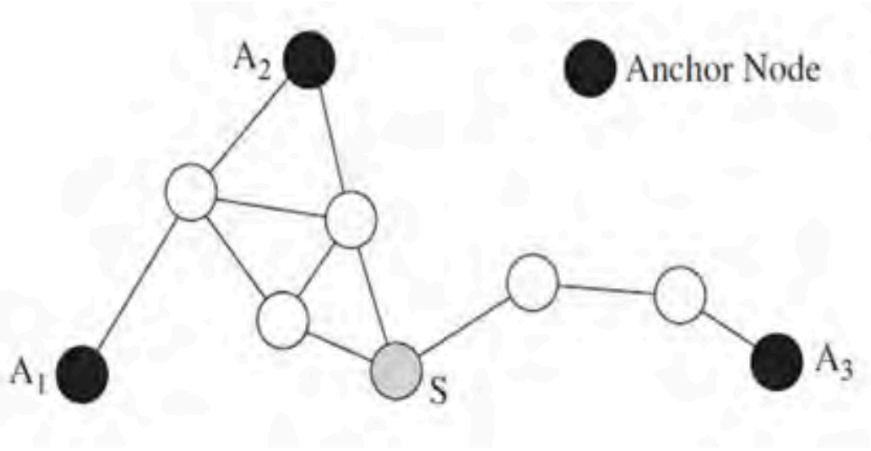


Figure 10.8 APS:DV-hop localization.

BASIC PRINCIPLE

APS is the most typical model of distributed localization based on connectivity between nodes in the network. In this algorithm anchor nodes are also needed to estimate unknown positions, infact, the accuracy of estimation increases by increasing the number of anchors.

At least three anchor nodes are required for localization.

Each anchor node propagates its position to other nodes using Distance Vector.

All the nodes exchange their routing tables with one-hop neighbours. In the most basic scheme of APS, called DV-hop, each node stores a table $\{X_i, Y_i\}$ h_i where $\{X_i, Y_i\}$ means the location of anchor node i and h_i is the distance in hops between this node and node i.

When an anchor node knows the location of other anchor nodes and the distance in hops between them, it can determine the average size of one hop called the correction factor.

Proposed by Niculescu and Nath in [2001] as Ad-Hoc Positioning System

RANGE-FREE LOCALIZATION METHODS DV-HOP [2001]

Uses a distance–vector flooding technique to determine the minimum hop count and average hop distance to known anchors' positions.

- Each anchor broadcasts a packet with its location and a hop count, initialized to one.
- The hop–count is incremented by each node as the packet is forwarded.
- Each node maintains a table of minimum hop–count distances to each anchor.

RANGE-FREE LOCALIZATION METHODS DV-HOP

- Once an anchor gets distance information from all other anchors. It calculates a correction to the average hop distance based on the following equation

$$c_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_i}$$

Where anchor *i* is the anchor that calculates correction, j are other known anchors for *i*

- Individual nodes use the average hop distance calculated from nearest anchor, along with the hop count to known anchors, to calculate their local position using lateration.

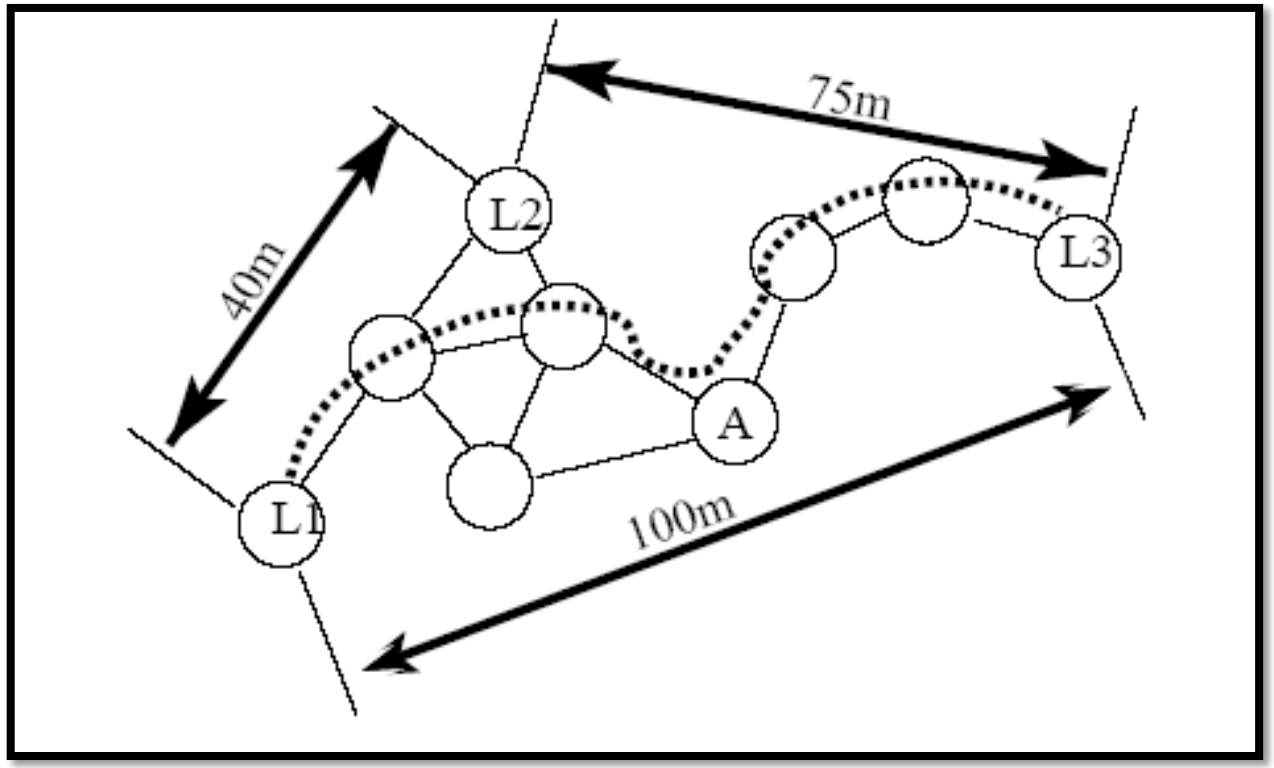
RANGE-FREE LOCALIZATION METHODS DV-HOP

L_1 computes the correction $\frac{100+40}{6+2} = 17.5$

L_2 computes a correction of $\frac{40+75}{2+5} = 16.42$

L_3 a correction of $\frac{75+100}{6+5} = 15.90$

A gets its correction from L_2



Distance to L1 = 3 x 16.42, to L2 = 2 x 16.42, to L3 = 3 x 16.42

RANGE-FREE LOCALIZATION METHODS DV-HOP

Advantages

- Simple

Drawback

- Works only for isotropic (direction insensitive propagation) networks
- Estimation error depends on the number of anchors that a node can hear
- Large overhead

RANGE-FREE LOCALIZATION METHODS AMORPHOUS (GRADIENT) [1999]

Proposed by Nagpal, Shrobe and Bachrach in [1999] independently from the DV−HOP Proposal

Same as DV−HOP except for average hop distance calculation to be equal to

$$r\left(1+e^{-n_{local}}-\int_{-1}^{1}e^{-\frac{n_{local}}{\pi}\left(\arccos t-t\sqrt{1-t^2}\right)}\right)$$

Where n_{local} is the number neighbors, r is the maximum hop distance (communication Range)

IMPROVEMENTS

Localization through APS can have some errors, however, some improvements are suggested in order to minimize the estimation error.

1. When each node gets the one-hop distance, use the formula

$$\text{Hopsize}_{\text{ave}} = \frac{\sum \text{Hopsize}_i}{n},$$

where n is the number of anchor nodes, Hopsize_i denotes the one-hop distance calculated from anchor i.

2. Increase the number of anchor nodes, the more anchors ensures more reliable results.

3. We can analyze the deviation between each anchor's one-hop distance and the average one-hop distance, then modify the average one-hop distance we previously got. The following steps will show how it works. In the beginning, we define the average one-hop distance deviation of anchor i as

$$\text{err\_dis}_i = \sum_{i \neq j} \frac{|d_{\text{true}} - d_{\text{estimate}}|_{ij}}{\text{hops}_{ij} \times n_i} \qquad (10.18)$$

Here, hops_{ij} means number of hops between anchor i and j, n_i means the number of anchors in anchor's data list,

$$d_{\text{true}} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\,,$$

where

$$d_{\text{estimate}} = \text{Hopsize}_{\text{ave}} \times \text{hops}_{ij}\,.$$

First, we calculate the average one-hop distance, then we use Equation (10.18) to compute the average one-hop distance deviation and propagate it to the network in the form of $\{\text{id}, \text{err\_dis}_i\}$. Secondly, every node receiving this data package stores the information in its data list and transmit it to its neighbors, the package with the same id number will be dropped. Thirdly, when every node acquires the average one-hop deviation of each anchor, we can do the calculation $c\_\text{err\_dis} = \sum \text{err\_dis}_i/n$, where n is the number of anchors. Finally, we can recalculate the average one-hop distance:

$$\text{New\_Hopsize}_{\text{ave}} = \text{Hopsize}_{\text{ave}} + k \times c\_\text{err\_dis}\,, \tag{10.19}$$

where k is from -1 to 1.

GENETIC BASED IMPROVED DV-HOP ALGORITHM

Suppose that there are $m+n$ sensor nodes in a sensor network, which includes m anchor nodes, whose locations are known, and n unknown nodes whose locations are unknown. The coordinate of anchor node is (x_i, y_i), $(i = 1, 2, \ldots, m)$, and the coordinate of unknown node is (x, y).

According to the second phase of DV-Hop (Section "Estimating the distance among nodes"), we get the distance d_i between an unknown node and the anchor nodes i, $(i = 1, 2, \ldots, m)$. So we can get the following equations:

$$\begin{cases} \sqrt{(x-x_1)^2+(y-y_1)^2} = d_1 \\ \sqrt{(x-x_2)^2+(y-y_2)^2} = d_2 \\ \cdots \\ \sqrt{(x-x_m)^2+(y-y_m)^2} = d_m \end{cases} \tag{10}$$

An improved localization algorithm based on genetic algorithm in wireless sensor networks

Bo Peng · Lei Li

Because the distance is an estimated value, so there must be error. Therefore, we proposed an improved DV-Hop based on GA to minimize the error of localization problem in WSN, and the localization problem can be formulated as:

$$F(x,y) = Min\left(\Sigma_{i=1,\ldots,m} \mid \sqrt{(x-x_i)^2 + (y-y_i)^2} - d_i \mid\right) \tag{11}$$

According to the objective function shown in Eq. (11), we set the fitness function as follows:

$$fitness(x,y) = \frac{\alpha}{F(x,y)} \tag{12}$$

where α is a positive real coefficient.

The general steps of our proposed algorithm GADV-Hop are summarized as follows:

Step 1 Calculate the minimum hop-count value from each node to each anchor node;

Step 2 Estimate the average size for one hop of each anchor node according to Eq. (1);

Step 3 By using the minimum hop-count value and the average size for one hop, each unknown node estimates the distance to the anchor node according to Eq. (2);

Step 4 Determine the population feasible region of each unknown node and generate initial population in the feasible region;

Step 5 Evaluate the fitness of each individual;

Step 6 Three genetic operations (crossover, mutation, and selection) are performed to generate the next generation. This procedure is repeated until the termination criterion is satisfied.

Creation of the initial population

For improving localization accuracy or convergence rate, we restrain a population feasible region as shown in Fig. 2. Assume that the unknown node N has three nearest anchor nodes A_1, A_2, A_3, whose coordinates are (x_1, y_1), (x_2, y_2) and (x_3, y_3), and (H_1, H_2, H_3) are minimum hop-count value between anchor nodes A_1, A_2, A_3 and unknown node N respectively. The communication radius of all the sensor nodes is R.

We construct three squares by using the three anchor nodes to be the center, and using $(2RH_1, 2RH_2, 2RH_3)$ to be the length of the three squares respectively. Intersection of the three squares (shaded zone) is the population feasible region of unknown node N. Initial population is randomly generated in the feasible region, and the coordinate upper and lower bounds of the unknown node N can be defined as follows:

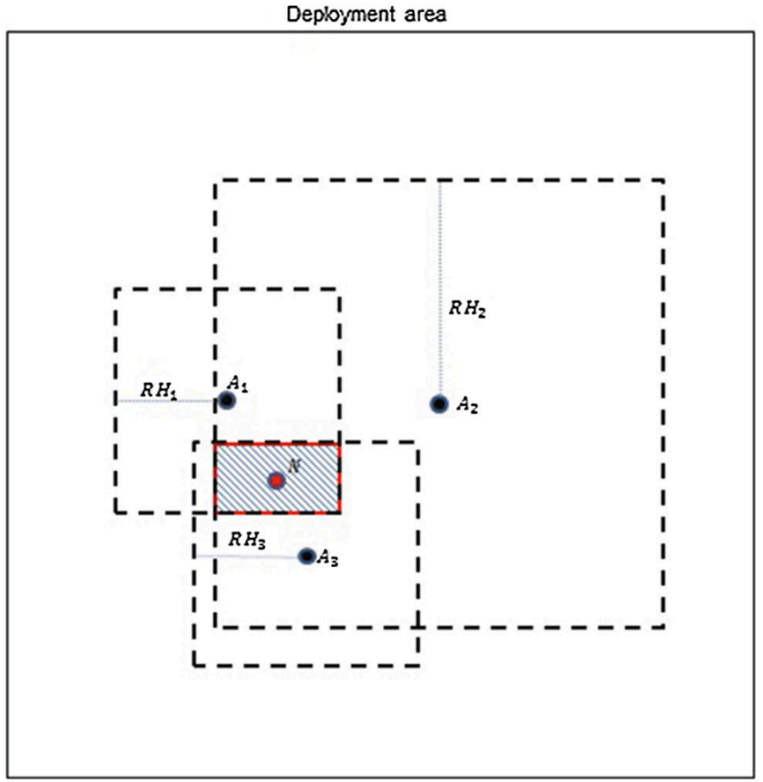


Fig. 2 Population feasible region

$$\begin{cases} \max\limits_{i=1,2,3}(x_i - RH_i) \le x_N \le \min\limits_{i=1,2,3}(x_i + RH_i) \\ \max\limits_{i=1,2,3}(y_i - RH_i) \le y_N \le \min\limits_{i=1,2,3}(y_i + RH_i) \end{cases} \tag{13}$$

where (x_N, y_N) is the coordinate of unknown node N.

Crossover

In this paper, we use an arithmetic crossover operator that defines a linear combination of two chromosomes. Two parents chromosomes (x_1, y_1) and (x_2, y_2) that selected randomly for crossover, may produce two offspring (x_1', y_1') and (x_2', y_2').

$$\begin{cases} x_1' = x_1\alpha + x_2(1-\alpha) \\ y_1' = y_1\alpha + y_2(1-\alpha) \end{cases} \tag{14}$$

$$\begin{cases} x_2' = x_2\alpha + x_1(1-\alpha) \\ y_2' = y_2\alpha + y_1(1-\alpha) \end{cases} \tag{15}$$

where α is a random number in range of $[0, 1]$.

Mutation

The mutation operator performs a random move to the neighbor in the population feasible region. An individual (x, y) is chosen for mutation, then the new coordinate (x', y') is defined as follow:

- $cos(\theta) \ge 0$

$$\begin{cases} x' = x + |x_{max} - x| \cdot cos(\theta) \\ y' = y + |y_{max} - y| \cdot sin(\theta) \end{cases} \tag{16}$$

- $cos(\theta) < 0$

$$\begin{cases} x' = x + |x_{min} - x| \cdot cos(\theta) \\ y' = y + |y_{min} - y| \cdot sin(\theta) \end{cases} \tag{17}$$

where $x_{min}, x_{max}, y_{min}, y_{max}$ are the upper and lower bounds of coordinates, $\theta \in [0, 2\pi]$ is an angle randomly produced.

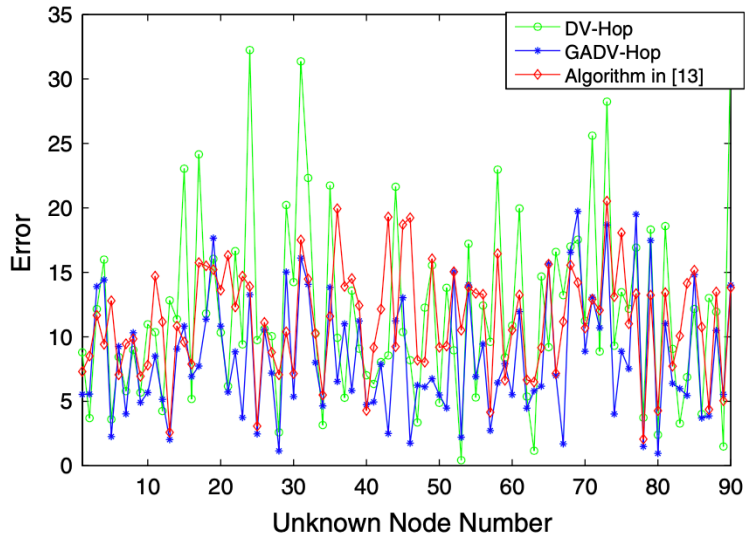


Fig. 4 Localization error of each unknown node. **a** R = 30, Anchor node=10. **b** R = 30, Anchor node = 20

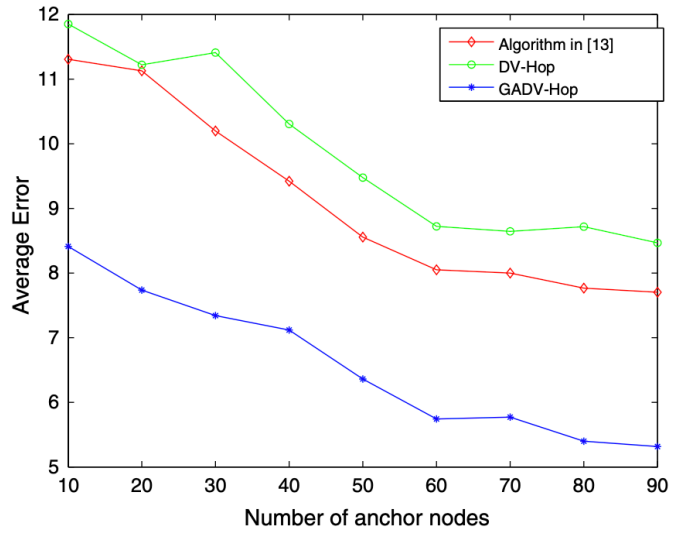


Fig. 5 Average localization error in different anchor node density

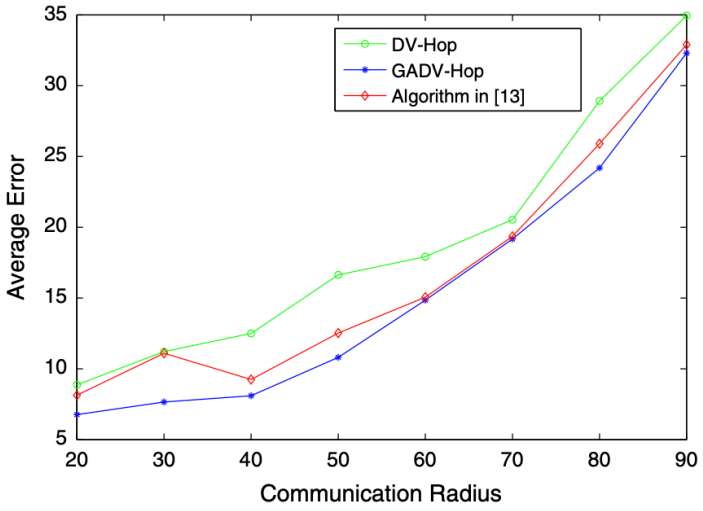


Fig. 6 Average localization error with the change of communication radius