Movement - Assisted Sensor Deployment

G. Wang, G. Cao, T. La Porta

Diego Cammarano

Laurea Magistrale in Informatica
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Sapienza, Università di Roma

December 9, 2013

Outline

Localization Techniques

- many applications such as environment monitoring and target tracking depend on knowing the **location** of sensor nodes
- each node must determine its location through a location discovery process (ex. GPS)
- many techniques based on **received signal strenght**, **time difference** of arrival, **angle** of arrival

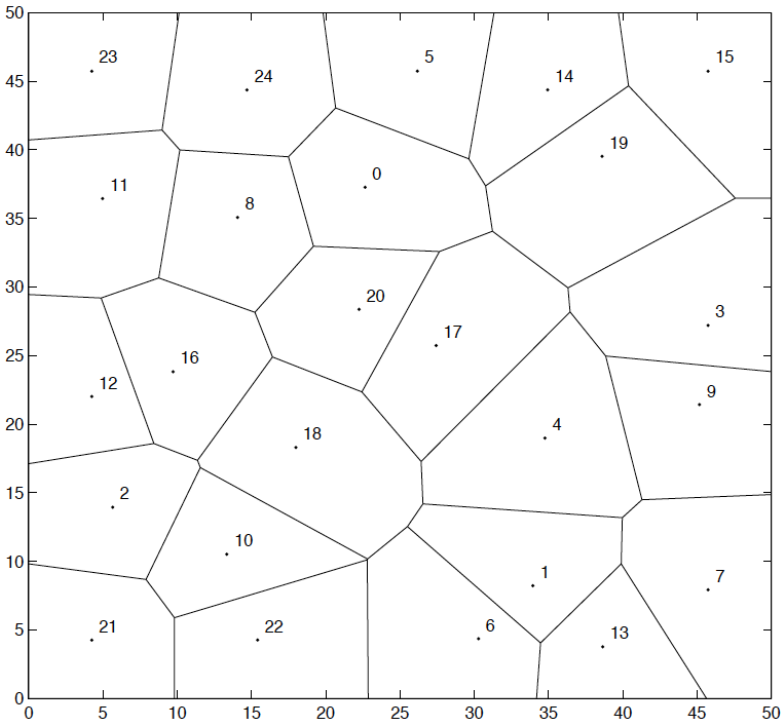
Path Planning

- problem initially studied in the robotic area and recently applied to sensor networks
- combine the known methods to find the best motion path and modified them to exploit the distributed nature of sensor networks
- assume that mobile sensors can move without any limitation using the existing techniques

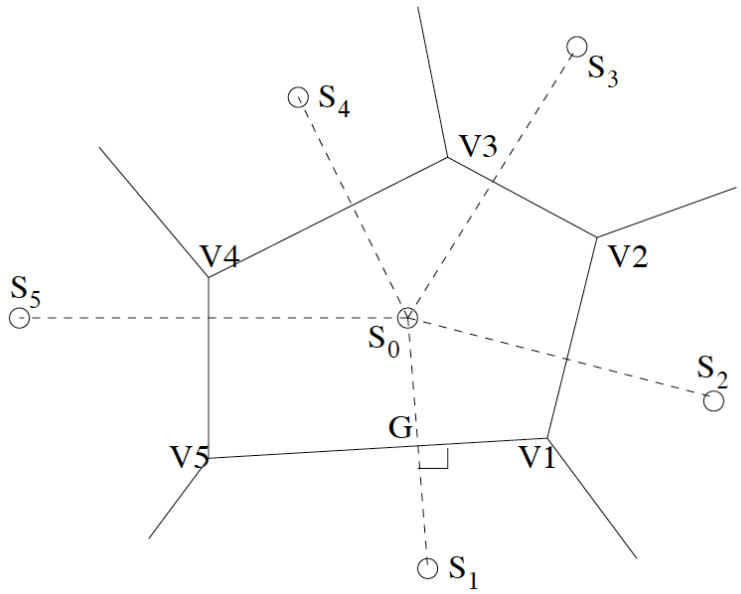
Sensing Model

- each type of sensor has its unique sensing model characterized by its **sensing area**, **resolution** and **accuracy**
- sensing area depends on multiple factors like **strenght** of the signals, **distance** between the source and sensor, desired **confidence level** of sensing
- each sensor node is associated with a sensing area represented by a circle with the same radius (**isotropic sensing models**)

Voronoi Diagram



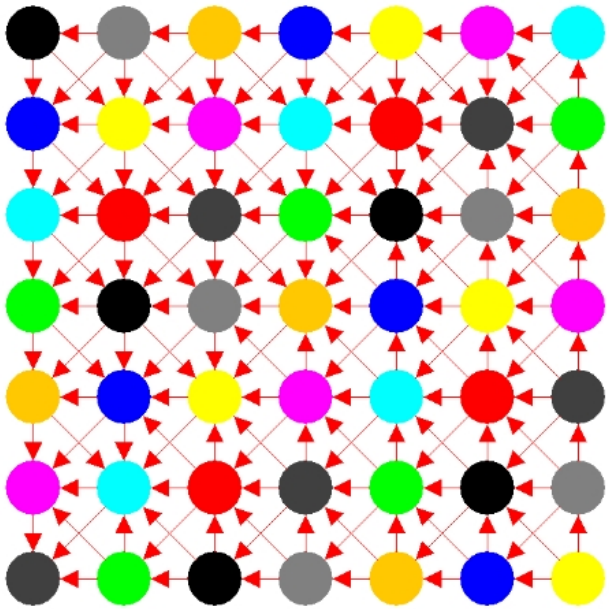
Voronoi Diagram



Voronoi polygon G_0 of s_0

Sensing and Communication Range

- sensors can exchange the location information by broadcasting
- some Voronoi neighbors of a sensor can be out of its **communication range** $\Rightarrow$ the calculated polygon of this sensor in not accurate
- if the **sensing range** $\ll$**communication range** $\Rightarrow$ the inaccurate construction of Voronoi cell will not affect the detection of coverage holes
- if **communication range** $\approx$ **sensing range** $\Rightarrow$sensors may mis-detect coverage holes



Basic deployment protocols

- the deployment protocol runs iteratively in rounds
- in each round sensors broadcast their locations and construct their Voronoi polygons
- if any hole exists, sensors calculate where to move to eliminate or reduce the size of the coverage hole

Three algorithms are proposed to calculate the target locations:

- **VEC** *pushes* sensors away from densely covered area
- **VOR** *pulls* sensors to the sparsely covered area
- **Minimax** moves sensors to the center of their Voronoi polygon

VECtor - based Algorithm

- inspired by the attributes of **electro-magnetic particles**: when two particles are too close to each other, an expelling force pushes them apart
- the **virtual force** will push the sensors away from each other if coverage hole exists in either of their Voronoi polygons
- the final overall force on sensors is the vector summation of virtual forces from the boundary and all Voronoi neighbors
- **VEC** is a "**proactive**" algorithm that tries to relocate sensors to be evenly distributed

VECtor - Pseudocode (1)

Upon *entering Discovery phase;*
    set timer to be *discovery interval*;
    enter *Moving* phase upon timeout;
    broadcast *hello* after a random time slot ;

Upon *entering Moving phase;*
    set timer to be *moving interval*;
    enter *Discovery* phase upon timeout;
if $c_i = false$ **then**
    call *VEC*()
Done when satisfying stop criteria;

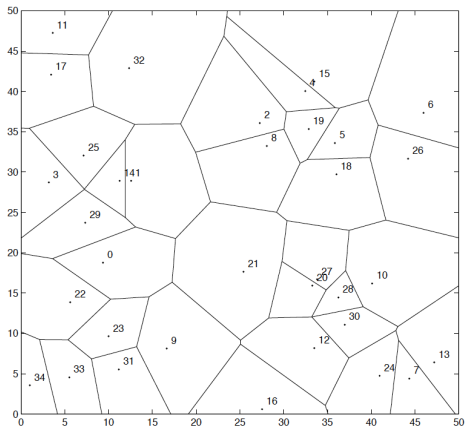
VECtor - Pseudocode (2)

Upon *receiving a hello message from sensor* s_j;
    Update N_i and G_i;
if G_i *is newly covered* **then**
    set $c_i = true$;
    broadcast OK;
Only for *VEC*
Upon *receiving an OK message from sensor* s_j;
    set $c_j = true$;

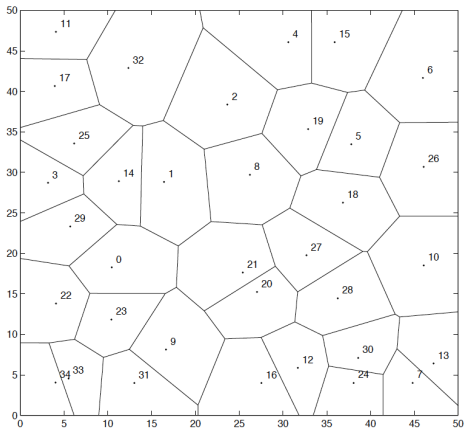
VECtor - Pseudocode (3)

VEC();
$\vec{v_i} = \vec{0}$;
for *each* s_j *in* N_i **do**
  if $c_i \neq true \wedge (d_{ave} > d(s_i, s_j))$ **then**
    $\vec{F_{ij}} = (d_{ave} - d(s_i, s_j))/2$; $\vec{v_i} = \vec{v_i} + \vec{F_{ij}}$;
  if $c_i = true \wedge (d_{ave} > d(s_i, s_j))$ **then**
    $\vec{F_{ij}} = (d_{ave} - d(s_i, s_j))$; $\vec{v_i} = \vec{v_i} + \vec{F_{ij}}$;
end
if $d_{ave}/2 - d_b(s_i)$ **then**
  $\vec{F_b} = d_{ave}/2 - d_b(s_i)$; $\vec{v_i} = \vec{v_i} + \vec{F_b}$;
do movement adjustment

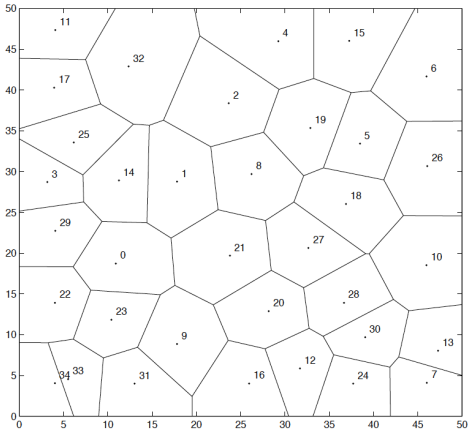
VECtor - Execution



Round 0



Round 1



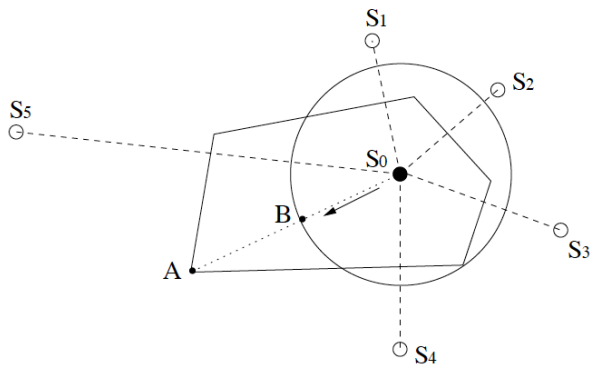
Round 2

VORonoi - Based Algorithm

- algorithm that pulls sensors to cover their local maximum coverage holes
- if a sensor detects the existence of coverage holes, it will move toward its farthest Voronoi vertex (V_{far}) and stop when V_{far} can be covered

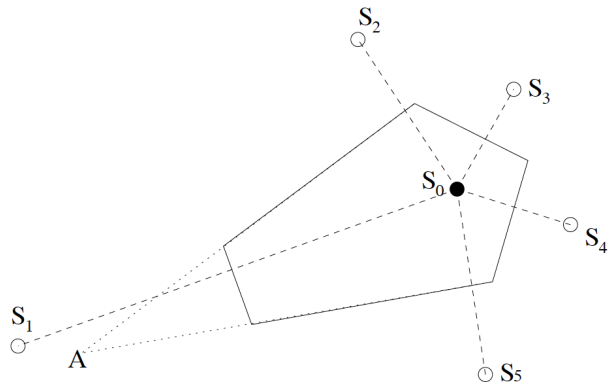
VORonoi - Example

- a sensor needs only to check its own Voronoi polygon
- A is the farthest Voronoi vertex of s_0 and $d(A, s_0)$ is longer than the sensing range
- s_0 moves along line s_0A to point B where $d(A, B)$ is equal to the sensing range



VORonoi - Inaccurate Voronoi polygon

- the maximum moving distance is limited to be at most half of the communication range
- s_0 does not know s_1 so it will calculate its local Voronoi polygon as the dotted one and view area around A as a coverage hole
- it is quite possible it has to move back after it gets to know s_1



VORonoi - Considerations

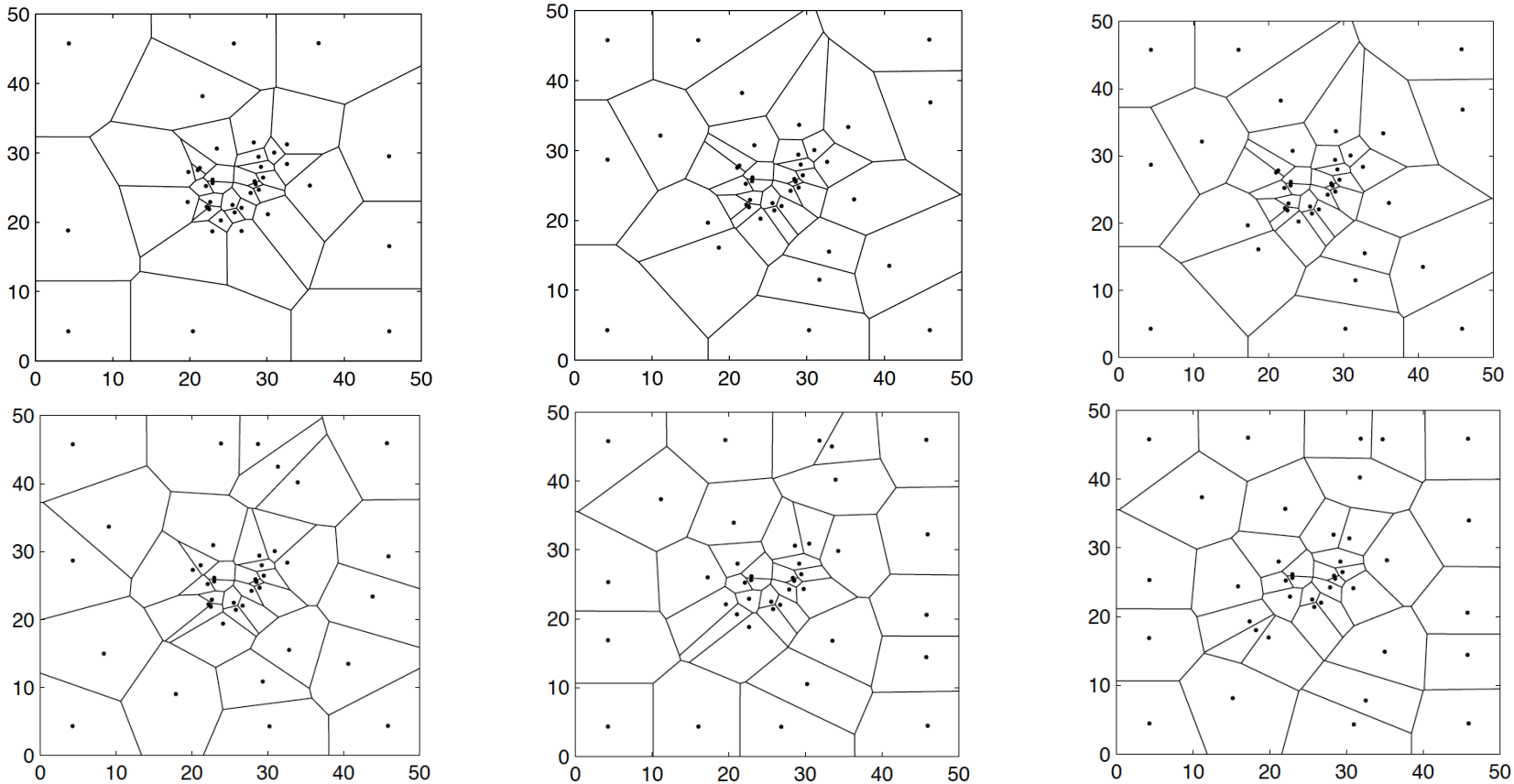
- VOR is a *greedy* algorithm which tries to **fix the largest** hole
- moving oscillations may occur if new holes are generated due to sensor's leaving
- to mitigate this problem is added an **oscillation control** that not allow sensors to move backward immediately
- before a sensor moves, it first check whether its moving direction is opposite to that in the previous round

VORonoi - Pseudocode

Notations:
d_{max} : maximum moving distance
$\vec{v_{i,f}}$:vector from s_i to V_{far}

VOR();
  $\vec{v_i} = \vec{v_{i,f}} -$ sensing range;
  shrink $\vec{v_i}$ to be d_{max} if $\vec{v_i} > d_{max}$;
  do oscillation contol;
  do movement-adjustment

VORonoi - Execution



Coverage during rounds 0-1-2-3-4-5

Minimax Algorithm

- similar to VOR
- chooses the target location as the point inside the Voronoi polygon whose distance to the farthest Voronoi vertex is minimized (Minimax point O_m)
- based on the idea that a sensor should not be too far away from any of its Voronoi vertices

Algorithm Termination - Proof (1)

Notations

$[x_i^{(r)}, y_i^{(r)}]$: location of the sensor s_i in the r^{th} round
$G_i^{(r)}$: Voronoi polygon of s_i in the r^{th} round
$A_i^{(r)}, \hat{A}_i^{(r)}$: area of the covered part of $G_i^{(r)}$

Lemma 1

$$(a)\ A_{total}^{(r)} = \sum_{i=1}^{n} A_i^r; \qquad (b)\ A_{total}^{(r+1)} = \sum_{i=1}^{n} \hat{A}_i^{(r)};$$

Proof.

(a) ✓ Voronoi diagram is a partition of the target field
(b) ✓ the summation of the covered area of the Voronoi polygons in the previous round is also the whole covered area in the current round □

Algorithm Termination - Proof (2)

Lemma 2

$A_i^{(r)} = A_i^{(r)}([x_i^{(r)}, y_i^{(r)}])$

Proof.

This is the direct result of the attribute of Voronoi diagram.

Every point within $G_i{}^{(r)}$ is closer to $[x_i^{(r)}, y_i^{(r)}]$ than to any other sensor.

Any point not covered by s_i is also not be covered by any other sensor. □

Algorithm Termination - Proof (3)

Theorem 1

$A_{total}^{(r+1)} > A_{total}^{(r)}$ before all sensors stop moving.

Proof.

At the $(r+1)^{th}$ round, there may be areas in ${G_i}^{(r)}$ which is not covered by s_i, but is covered by other sensors, because the current Voronoi polygon of s_i is ${G_i}^{(r+1)}$ but not ${G_i}^{(r)}$. Therefore

$$\hat{A}_i^{(r)} \geq {A_i}^{(r)}([x_i^{(r+1)}, y_i^{(r+1)}]) \qquad (1)$$

By enforcing the movement adjustment heuristics, *VEC*, *VOR* and *Minimax* algorithms guarantee that, if s_i moves,

$$A_i^{(r)}([x_i^{(r+1)}, y_i^{(r+1)}]) > A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]) \qquad (2)$$

Algorithm Termination - Proof (4)

Proof (continue).

Certainly, if s_1 does not move,

$$A_i^{(r)}([x_i^{(r+1)}, y_i^{(r+1)}]) = A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]) \qquad (3)$$

because $[x_i^{(r+1)}, y_i^{(r+1)}] = [x_i^{(r)}, y_i^{(r)}]$

From *(1),(2),(3)*

$$\sum_{i=1}^{n} \hat{A}_i^{(r)} \geq \sum_{i=1}^{n} A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]) \qquad (4)$$

if some sensor moves in the r^{th} round

Algorithm Termination - Proof (5)

Proof (continue).

From *Lemma 2* : $A_i^{(r)} = A_i^{(r)}([x_i^{(r)}, y_i^{(r)}])$ *(5)*

From *(4)*,*(5)*: $\sum_{i=1}^{n} \hat{A}_i^{(r)} > \sum_{i=1}^{n} A_i^{(r)}$ *(6)*

By *Lemma 1*: $\begin{cases} A_{total}^{(r)} = \sum_{i=1}^{n} A_i^{(r)} \\ A_{total}^{(r+1)} = \sum_{i=1}^{n} \hat{A}_i^{(r)} \end{cases}$ *(7)*

From *(6)*,*(7)*: $A_{total}^{(r+1)} > A_{total}^{(r)}$ □

Algorithm Termination - Proof (6)

Corollary 1

VEC, *VOR* and *Minimax* algorithms are convergent and thereby terminate naturally

Proof.

Following from *Theorem 1* and the fact that $A^{(r)}_{total}$ total is upper bounded by the total area of the target field, distributed algorithms converge, and terminate naturally.
All sensors stop moving when no coverage increase can happen. □

Dealing with message loss

Problem

- *Hello* messages may be lost due to collisions
- sensors may fail to know the presence of some neighbors and mistakenly determine coverage holes

Solution

- associate each item in a sensor's neighbor list with a number which indicates the **freshness** of this item (i.e for how many rounds this neighbor has not been heard)
- on constructing Voronoi polygon every sensor only consider the sensors in its neighbor list with certain freshness

Dealing with position clustering

Problem

- the initial deployment of sensors may form clusters resulting in a low initial coverage
- sensors located inside the clusters can not move for several rounds
- the deployment time is prolonged and some sensors can still be clustered after several rounds

Solution

- detection of situations in which many sensors are clustered in small area
- during the first round the algorthm "explodes" cluster to scatter the sensor apart
- each sensor after concludes that it is inside a cluster choose a random position within an area centered itself

A first approach

The basic protocols require sensors to move iteratively, eventually reaching the final destination. Other approaches can be envisioned in which the sensors move only once to their destination to minimize the sensor movement.

- sensors calculate their target locations, virtually move there
- exchange these new virtual locations with the sensors which would be their neighbors as if they had actually moved.
- the real movement only happens at the last round after final destinations are determined

Drawbacks

- susceptible to poor performance under network partitions
- high communication overhead

A better approach

Idea

- To get balance between movement and message complexity, sensors do virtual movement when the communication cost to reach the logical Voronoi neighbors is reasonable, and do physical movement otherwise.
- The challenge is to determine if a sensor can reach its logical neighbors with reasonable communication cost.

Heuristic

1. if sensor's distance to its farthest Voronoi vertex is shorter then half of the communication range, it must know all its Voronoi neighbors
   - one hop broadcast is enough to exchange the location information with its logical neighbors
   - physical movement is not necessary

2. some Voronoi neighbors are out of the communication range
   - sensors request their neighbors lists to obtain the logical positions of sensors within two broadcast hops
   - when the distances between the physical locations of sensors and their farthest Voronoi vertices are larger than two times the maximum moving distance, sensors should move physically

Implementation

1. **Discovery phase**
   - sub-phase 1: sensors broadcast *hello* messages
   - sub-phase 2: sensors broadcast the location of known neighbors

2. **Logical moving phase**
   - if a sensor's distance to its farthest Voronoi vertex is larger than half of the communication range, calculate the target location and do logical movement
   - set a flag in the hello messages to require broadcasting of neighbors list
   - any sensor that receives a hello message will broadcast its neighbor list in the second sub-phase of the discovery phase

3. **Physical moving phase**
   - its physical position is two times the maximum moving distance to its farthest Voronoi vertex
   - a sensor's logical position has not changed for several rounds

Virtual Algorithm - Pseudocode (1)

Upon *entering Discovery phase-I;*
    set *timer* to be $discovery - interval/2$;
    enter *Discovery phase-II*;
    broadcast $hello(w_i)$ after a random time slot;
Upon *entering Discovery phase-II;*
    set *timer* to be $discovery - interval/2$;
    enter *Moving phase* upon timeout;
if $l_i = true$ **then**
    broadcast NL_i after a random time slot;
    $l_i = false$;

Virtual Algorithm - Pseudocode (2)

Upon *entering Moving phase;*
    set *timer* to be *moving* − *interval*;
    enter *Discovery phase-I* upon timeout;
if $c_i = false$ **then**
    calculate $loc_i^{'}$ by *VEC* or *VOR* or *Minimax*;
    do *oscillation control*;
    do *movement adjustment*;
    $p_i = 1$ if logically moves;
    if $d(loc_i, V_{far}) \geq 2 * d_{max}$ **then**
        move to $loc_i^{'}$;
        $w_i = false$;
    if $d(loc_i^{'}, V_{far}) \geq d_c/2$ **then**
        move to $loc_i^{'}$;
        $w_i = true$;
    .....

Virtual Algorithm - Pseudocode (3)

Upon *receiving a hello*(w_j) *message from sensor* s_j;
if $w_j = true$ **then**
  $l_i = true$;
Update N_i and G_i;
if G_i *is newly covered* **then**
  set $c_i = true$;
Upon *receiving a* NL_j *message from sensor* s_j;
  Update N_i and G_i;

Objectives

- testing the effectiveness of protocols in providing high coverage
- comparing **VEC**, **VOR** and **Minimax**
- comparing the **Basic protocols** and the **Virtual movement** protocols
- studying the effectiveness of controlling the tradeoff among various metrics by adjusting parameters

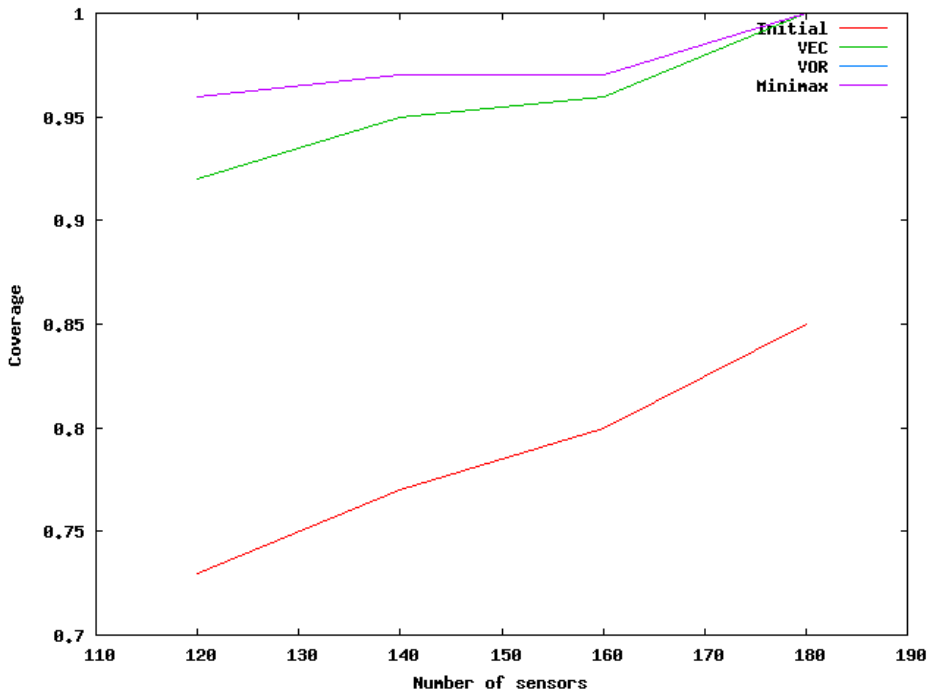
Metrics

1. **Deployment quality**
   - sensor coverage
   - time (number of rounds) to reach this coverage
2. **Energy consumption**
   - mechanical movement (starting/braking energy and moving energy)
   - communication

Methodology

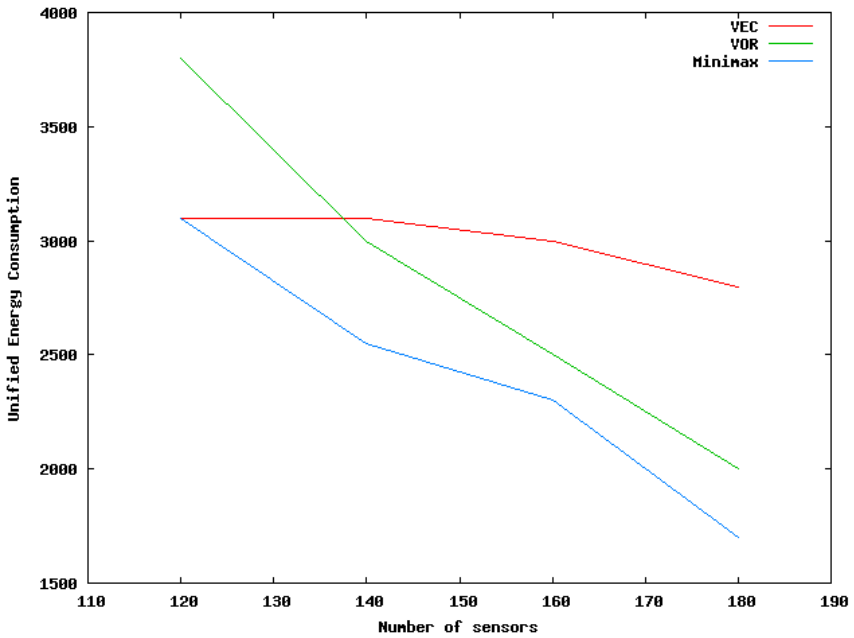
- simulations done under **different sensor density** to determine the sensor coverage that can be reached and the difficulty to reach it
- a **100m*100m** target field, **4** different number of sensors distributed (from 120 to 180)
- run **10 experiments** based on different initial distribution and calculate the average results
- **802.11 MAC** layer protocol, **DSDV** routing protocol based on **Bellman-Ford** algorithm
- **6** meters **sensing range**, **20** meters **communication range**

Coverage

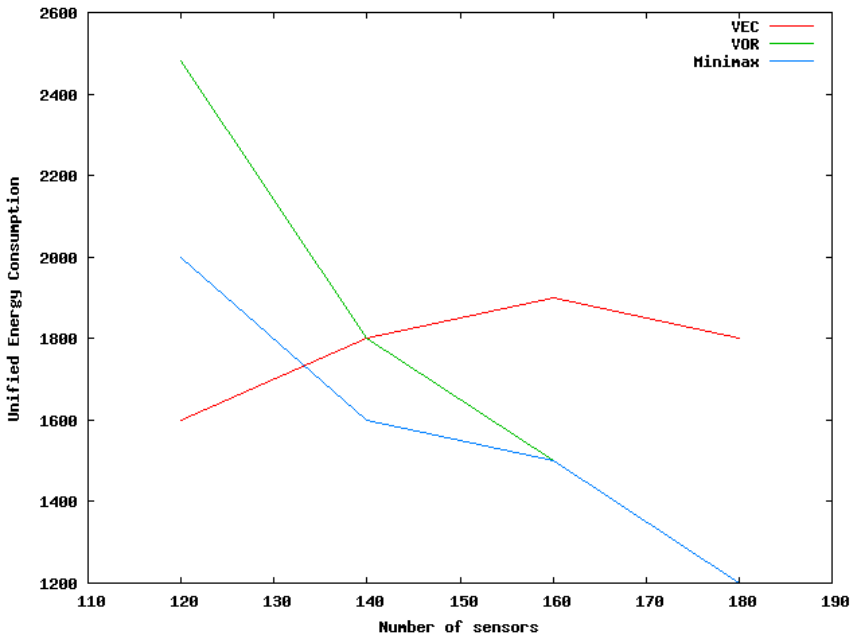


- **VEC** is sensitive to initial deployment
- **VOR** and **Minimax** both move to heal the holes directly and achieve quite similar coverage
- **Basic protocols** and **Virtual movement** protocols achieve almost the same coverage

Energy Consuption - Basic vs Virtual Protocols



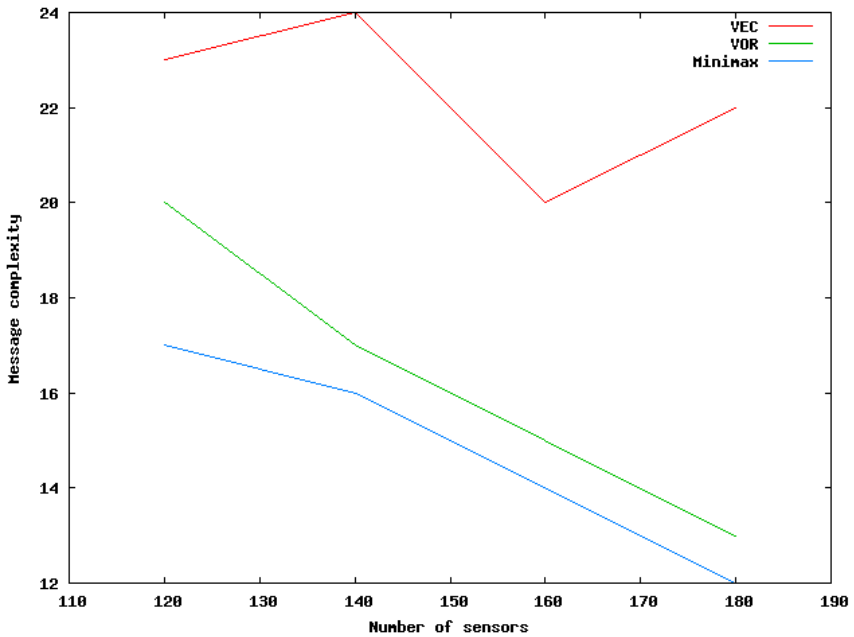
Basic



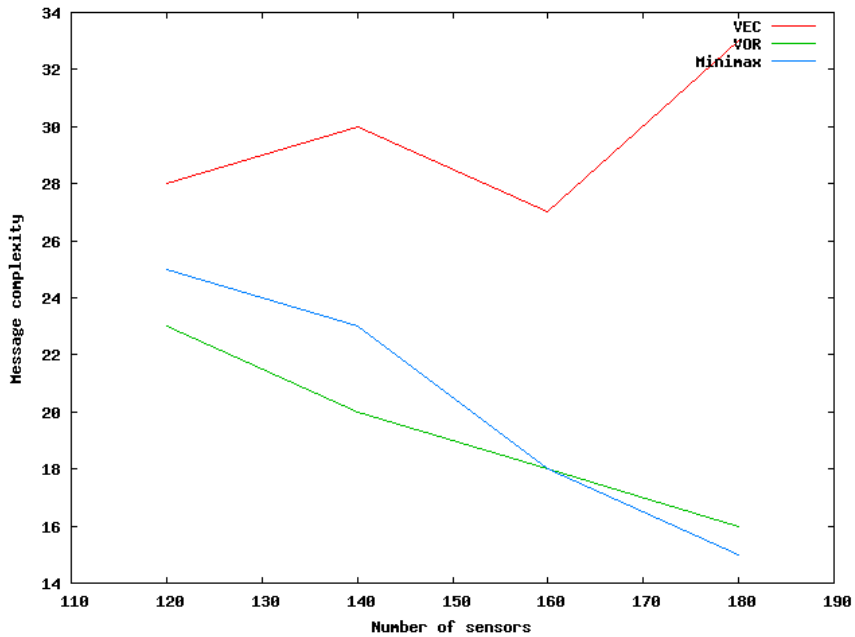
Virtual

Message Complexity - Basic vs Virtual Protocols

Message complexity is the number of messages exchanged when the protocol terminates and represents the normalization of the *moving distance* and the *number of movements*

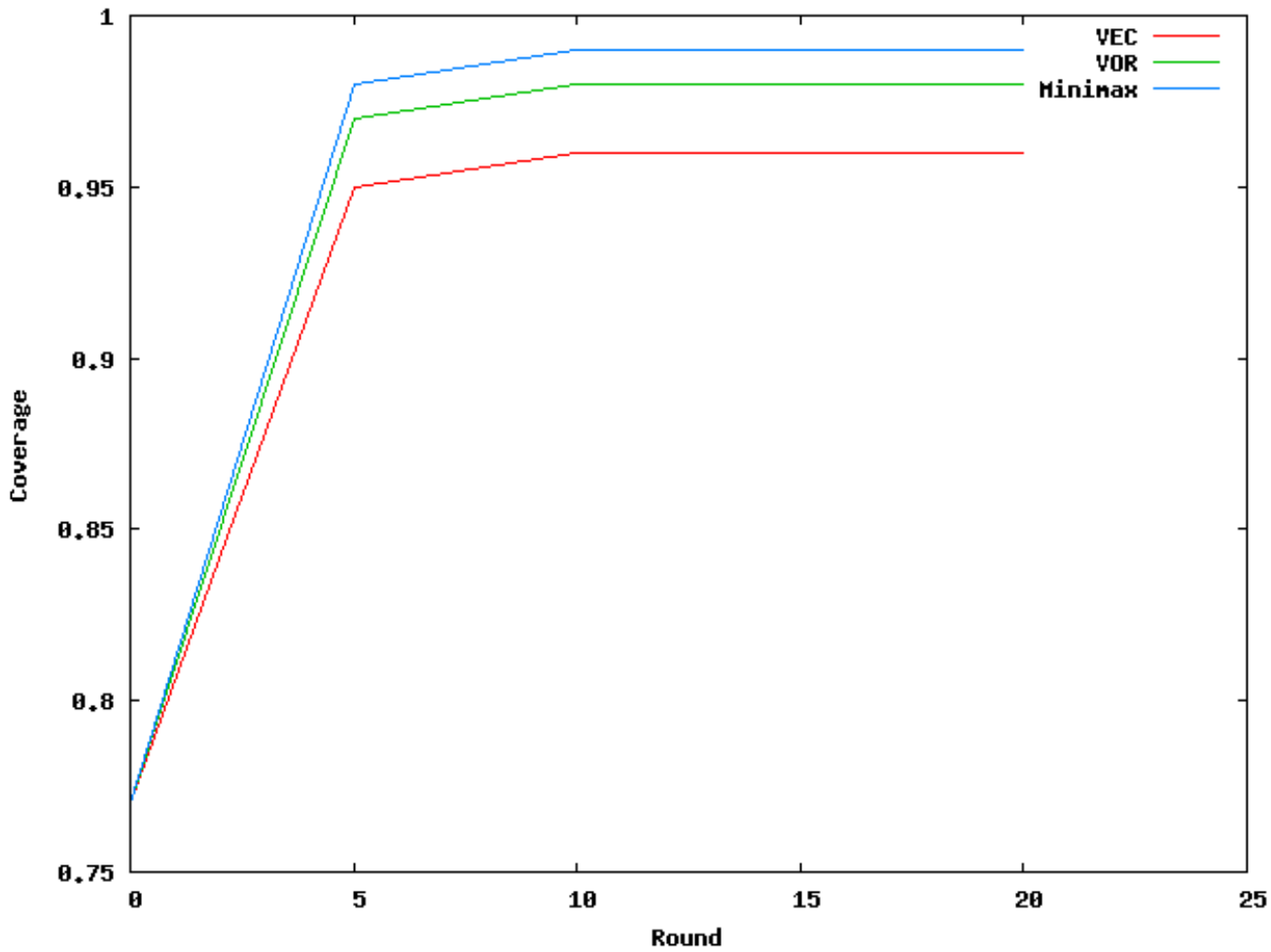


Basic

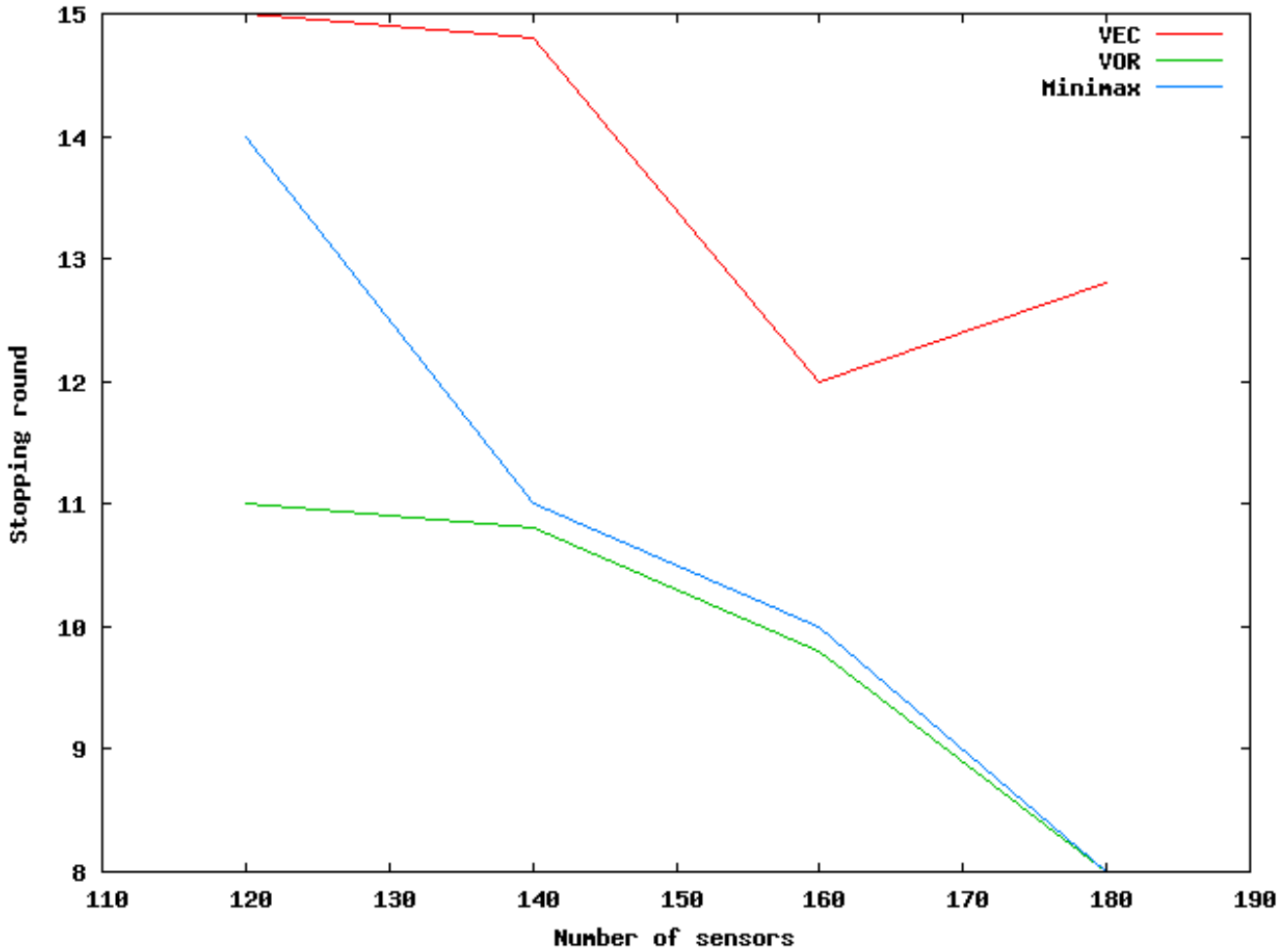


Virtual

Convergence time



Termination



Considerations

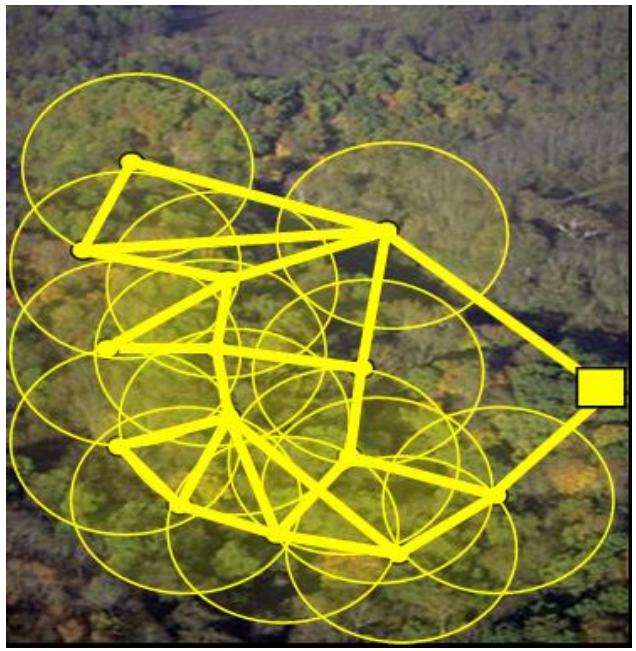
- **VEC** move least at *low sensor density* and can be deployed when the coverage requirements is not high
- **VOR** terminates the *earliest* when the *sensor density* is *not very high* and it can be deployed when both the deployment time requirement and coverage requirement are strict
- **Minimax** is the *best choice* to calculate the *target location*
- **Virtual movement** protocols can significantly *reduce mechanical movement* with a cost of less then two

Distributed scheme

- sensors calculate their target location following a distributed approach
- sensors only move when there are coverage holes
- can't always guarantee the optimal coverage

Centralized scheme

- a centralized approach may not be feisible in some deployments and suffers from the problem of a *single point of failure*
- optimal position to place sensors is decided *a priori* by a *central server* who minimize the *moving distance* of the sensors
- this is a *bipartite matching* problem and the classic *Hungarian method* is used to distribute sensors minimizing their moving distance



Sensing Area

Area shape

- the sensing area of each sensor is a disk with radius 6m
- if the sensing area is uniformly a disk, protocols can deal well with the case of a larger or smaller sensing radius
- if the sensing area is an irregular shape, sensors can still determine the coverage holes by reducing the sensing range

Performance

- performance depends more on the ratio of communication range to sensing range than the absolute sensing range
- on decreasing of the sensing range, protocols can accurately construct the Voronoi diagrams
- on increasing of the sensing range, it is necessary to enlarge broadcast hops to better construct the Voronoi polygons

Questions