Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

Δημήτριος Κατσαρός, Ph.D.

Χειμώνας 2005

Διάλεξη 11η

Ιστοσελίδα του μαθήματος

- **http://skyblue.csd.auth.gr/~dimitris/courses/mpc_fall05.htm**
- **http://skyblue.csd.auth.gr/~dimitris/courses/mpc_fall05/**
  - **books/**
  - **lectures/**
  - **papers/**
  - **proj_papers/**
  - **present_papers/**
- Τοποθετούνται οι διαφάνειες του επόμενου μαθήματος
- Τοποθετούνται τα research papers που αντιστοιχούν σε κάθε διάλεξη. Τα σημαντικά με πρόθεμα **MUST_BE_READ**

Περιεχόμενα

- Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks)
  - Πρόβλημα κάλυψης και συνδεσμικότητας σε ασύρματα δίκτυα αισθητήρων

Coverage Problems

- In general
  - Determine how well the sensing field is monitored or tracked by sensors.
- Possible Approaches
  - Geometric Problems
  - Level of Exposure
  - Area Coverage
    - Coverage
    - Coverage and Connectivity
    - Coverage-Preserving and Energy-Conserving Problem

Review: Art Gallery Problem

- Place the minimum number of cameras such that every point in the art gallery is monitored by at least one camera.

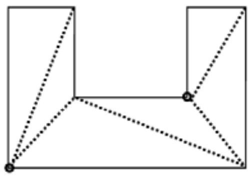


Figure 1: An example of triangulating a polygon and a possible deployment of cameras. Circles represent positions of cameras.

Review: Circle Covering Problem

- Given a fixed number of identical circles, the goal is to minimize the radius of circles.

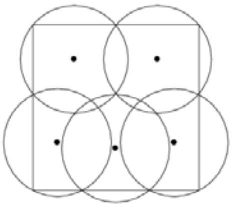
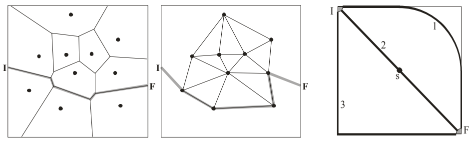


Figure 2: An example of an optimal covering with 5 circles. The radius of each circle is about 0.3621605.

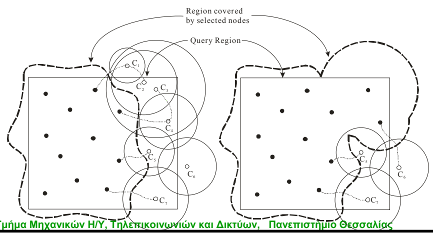
Level of Exposure

- Breach and support paths
  - paths on which the distance from any point to the closest sensor is maximized and minimized
  - Voronoi diagram and Delaunay triangulation
- Exposure paths
  - Consider the duration that an object is monitored by sensors



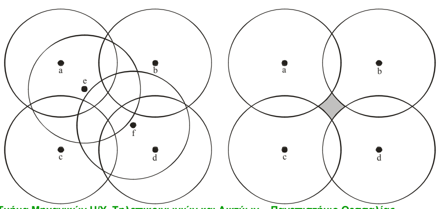
Coverage and Connectivity

- A region is *k*-covered, then the sensor network is *k*-connected if $\mathbf{R_C \geq 2R_S}$
- Extending the coverage such that connectivity is maintained.



Coverage-Preserving and Energy-Conserving Protocols

- Sensors' on-duty time should be properly scheduled to conserve energy.
  - thus extending the lifetime of the network.
  - This can be done if some nodes share the common sensing region.



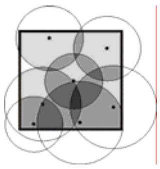
10

The Coverage Problems in 2D Spaces

11

Coverage Problems

- In general
  - To determine how well the sensing field is monitored or tracked by sensors
  - Sensors may be randomly deployed

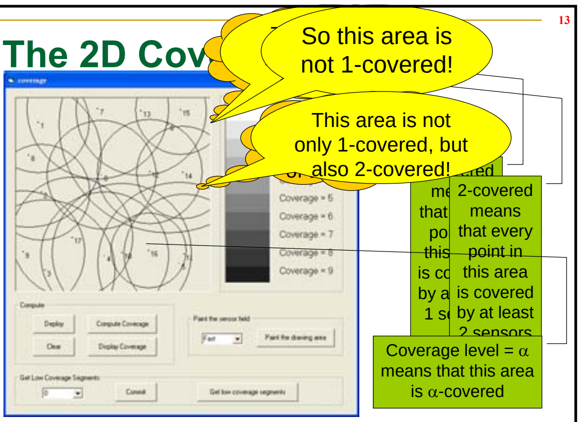


12

Coverage Problems

- We formulate this problem as
  - Determine whether every point in the service area of the sensor network is covered by at least α sensors
  - Why α sensors?
    - Localization, positioning, and tracking applications
    - Fault-tolerance

The 2D Cov



Sensing and Communication Ranges

Radio transmission range of Berkeley Motes

Product	Transmission Range
MPR300*	30m
MPR400CB	150m
MPR410CB	300m
MPR420CB	300m
MPR500CA	150m
MPR510CA	300m
MPR520CA	300m

Table 2: Sensing range of several typical sensors

Product	Sensing Range	Functions
HMC1002 Magnetometer sensor[8]	5m	Detecting disturbance from Automobiles
Reflective type photoelectric sensor [2]	1m	Detecting targets of virtually any material
Thrubeam type photoelectric sensor [2]	10m	Detecting targets of virtually any material
Pyroelectric infrared sensor (RE814S) [18]	30m	Detecting moving objects
Acoustic sensor Berkeley Motes * [8]	~ 1m	Detecting acoustic on sound sources

* This result is based on our own measurement on Berkeley motes [8]

[1]Honghai Zhang and Jennifer C. Hou, ``On deriving the upper bound of α-lifetime for large sensor networks," Proc. *ACM Mobihoc 2004*, June 2004

Assumptions

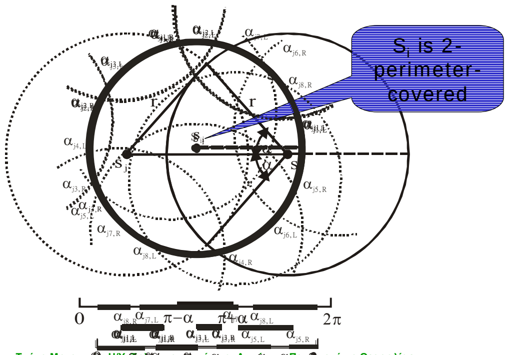
- Each sensor is aware of its geographic location and sensing radius.
- Each sensor can communicate with its neighbors.

- Difficulties:
  - $O(n^2)$ regions divided by n circles
  - How to determine boundaries of these regions?

The Proposed Solution

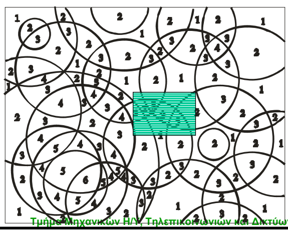
- We try to look at how the **perimeter** of each sensor's sensing range is covered.
  - How a perimeter is covered implies how an area is covered
  - … by the continuity of coverage of a region
- By collecting **perimeter coverage** of each sensor, the level of **coverage of an area** can be determined.
  - a distributed solution

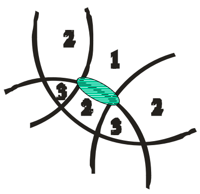
How to calculate the perimeter cover of a sensor?



Relationship between k-covered and k-perimeter-covered

- THEOREM. *Suppose that no two sensors are located in the same location. The whole network area A is* ***k-covered*** *iff each sensor in the network is* ***k-perimeter-covered***.



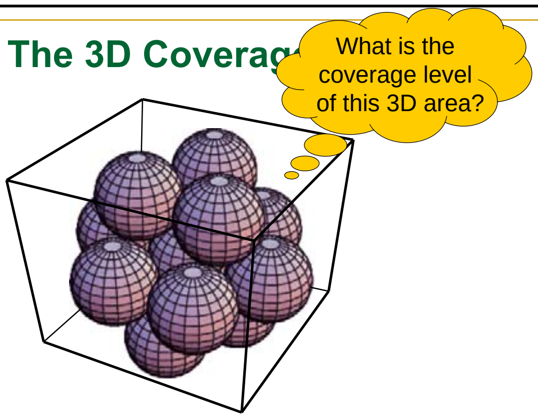


Detailed Algorithm

- Each sensor independently calculates its perimeter-covered.
  - k = min{each sensor's perimeter coverage}
- Time complexity: nd *log*(d)
  - n: number of sensors
  - d: number of neighbors of a sensor

The Coverage Problem in 3D Spaces

The 3D Coverag



The 3D Coverage Problem

- Problem Definition
  - Given a set of sensors in a 3D sensing field, is every point in this field covered by at least α sensors?
- Assumptions:
  - Each sensor is aware of its own location as well as its neighbors' locations.
  - The sensing range of each sensor is modeled by a **3D ball**.
  - The sensing ranges of sensors can be non-uniform.

Overview of Our Solution

- The Proposed Solution
  - We reduce the geometric problem from a 3D space to one in a 2D space, and further to one in a 1D space.

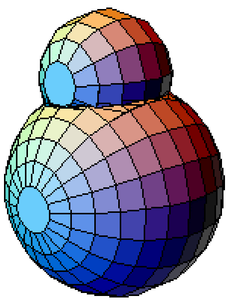
Reduction Technique (I)

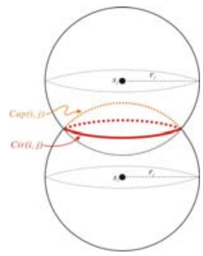
- 3D => 2D
  - To determine whether the whole sensing field is sufficiently covered, we look at the **spheres** of all sensors
  - *Theorem 1:* If each sphere is ***α-sphere-covered***, then the sensing field is *α-covered*.
    - Sensor s_i is ***α-sphere-covered*** if all points on its sphere are ***sphere-covered*** by at least α sensors, i.e., on or within the spheres of at least α sensors.

Reduction Technique (II)

- 2D => 1D
  - To determine whether each sensor's sphere is sufficiently covered, we look at how each **spherical cap** and how each **circle** of the intersection of two spheres is covered.
    - (refer to the next page)
  - Corollary 1: Consider any sensor s_i. If each point on S_i is *α-cap-covered*, then sphere S_i is **α-sphere-covered**.
    - A point *p* is *α-cap-covered* if it is on at least *α* caps.

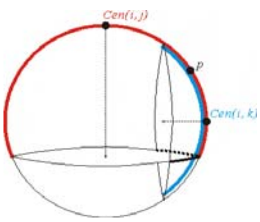
Cap and Circle





k-cap-covered

- *p* is *2-cap-covered* (by *Cap(i, j)* and *Cap(i, k)*).

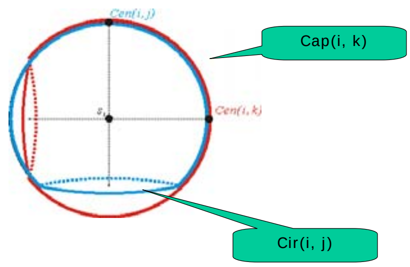


Reduction Technique (III)

- 2D => 1D
  - *Theorem 2:* Consider any sensor s_i and each of its neighboring sensor s_j. If each circle *Cir(i, j)* is ***α-circle-covered***, then the sphere S_i is ***α-cap-covered***.
  - A circle is ***α-circle-covered*** if every point on this circle is covered by at least α caps.

k-circle-covered

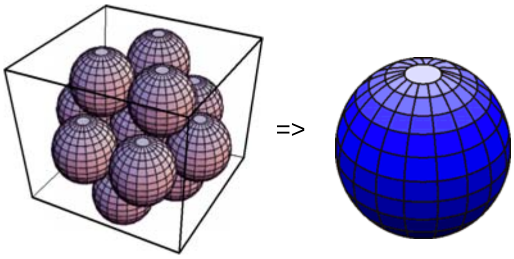
- *Cir(i, j)* is *1-circle-covered* (by *Cap(i, k)*).



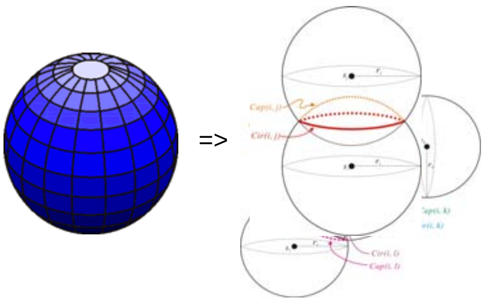
Reduction Technique (IV)

- 2D => 1D
  - By stretching each circle on a 1D line, the level of circle coverage can be easily determined.
  - This can be done by our 2-D coverage solution.

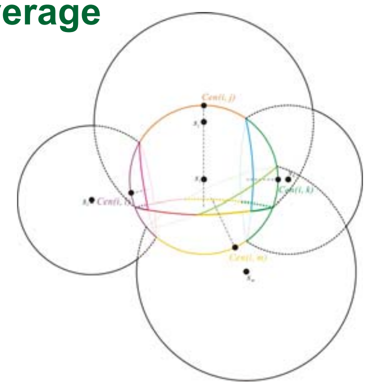
Reduction Example



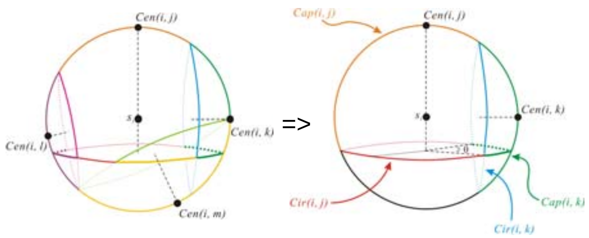
Reduction Example



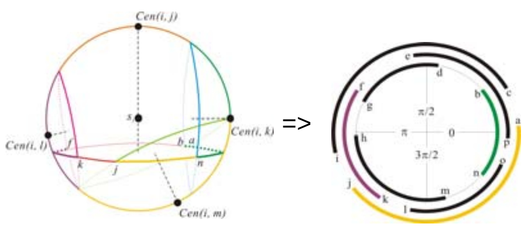
Calculating the Circle Coverage



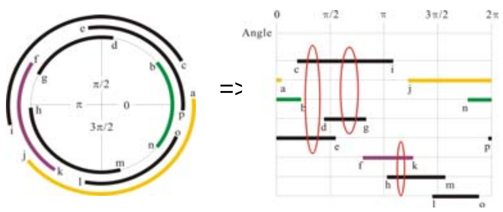
Calculating the Circle Coverage



Calculating the Circle Coverage



Calculating the Circle Coverage



37

The Complete Algorithm

- Each sensor s_i independently calculates the **circle coverage** of each of the circle on its sphere.
  - **sphere coverage of s_i** = min{ circle coverage of all circles on S_i }
- overall coverage = min{ sphere coverage of all sensors }

38

Complexity

- To calculate the sphere coverage of one sensor: $O(d^2 \log d)$
  - d is the maximum number of neighbors of a sensor
- Overall: $O(nd^2 \log d)$
  - n is the number of sensors in this field

39

Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network

Outline

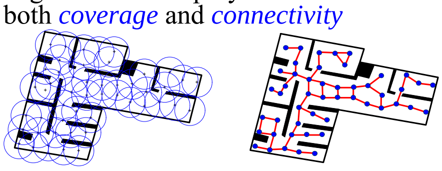
- **Introduction**
- Sensor Placement
- Sensor Dispatch
- Conclusions

Introduction

- Wireless sensor networks (WSN)
  - Tiny, low-power devices
  - Sensing units, transceiver, actuators, and even mobilizers
  - Gather and process environmental information
- WSN applications
  - Surveillance
  - Biological detection
  - Monitoring

Introduction

- Sensor deployment is a critical issue because it affects the *cost* and *detection capability* of a wireless sensor network
- A good sensor deployment should consider both *coverage* and *connectivity*



Coverage Connectivity

Review

- The *art gallery problem (AGP)* asks how to use a minimum set of guards in a polygon such that every point of the polygon is watched by at least one guard.
- However, the results cannot be directly applied to sensor deployment problem because
  - AGP typically assumes that a guard can watch a point as long as line-of-sight exists
    - Sensing distance of a sensor is normally *finite*
  - AGP does NOT address the communication issue between guards
    - Sensor deployment needs to address the *connectivity* issue

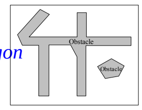
Two Issues in Sensor Deployment

- Sensor placement problem:
  - Ask how to place the *least number of sensors* in a field to achieve desired coverage and connectivity properties.
- Sensor dispatch problem:
  - Assume that sensors are *mobilized*
  - Given a set of mobile sensors and an area of interest **I** inside the sensing field **A**, to choose *a subset of sensors* to be delegated to **I** with *certain objective functions* such that the coverage and connectivity properties can be satisfied

Outline

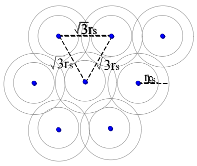
- Introduction
- **Sensor Placement**
- Sensor Dispatch
- Conclusions

Sensor Placement Problem



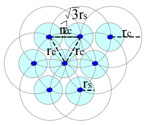
- Input: sensing field **A**
  - **A** is modeled as an *arbitrary-shaped polygon*
  - **A** may contain several obstacles
    - Obstacles are also modeled by *polygons*
    - Obstacles do NOT partition **A**
- Each sensor has a sensing distance r_s and communication distance r_c
  - But we do NOT restrict the relationship between r_s and r_c
- Our goal is to place sensors in **A** to ensure both sensing coverage and network connectivity *using as few sensors as possible*

Two Intuitive Placements



Consider coverage first

Need to add extra sensors to maintain *connectivity* when $r_c < \sqrt{3} r_s$



Consider connectivity first

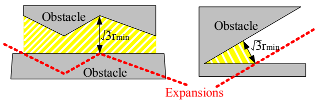
Need to add extra sensors to maintain *coverage* when $r_c > \sqrt{3} r_s$

Proposed Placement Algorithm

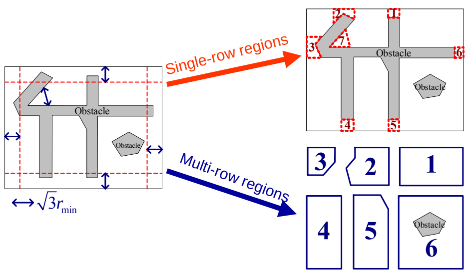
- Partition the sensing field **A** into two types of sub-regions:
  - *Single-row regions*
    - A belt-like area between obstacles whose width is NOT larger than $\sqrt{3} r_{min}$, where $r_{min} = \min(r_s, r_c)$
    - We can deploy *a sequence of sensors* to satisfy both coverage and connectivity
  - *Multi-row regions*
    - We need multi-rows sensors to cover such areas
    - Note: Obstacles may exist in such regions.

Step 1: Partition the Sensing Field

- From the sensing field **A**, we identify all *single-row regions*
  - Expand the perimeters of obstacles *outwardly* and **A**'s boundaries *inwardly* by a distance of r_{min}
  - If the expansion overlaps with other obstacles, then we can take a *projection* to obtain single-row regions
- The remaining regions are multi-row regions.

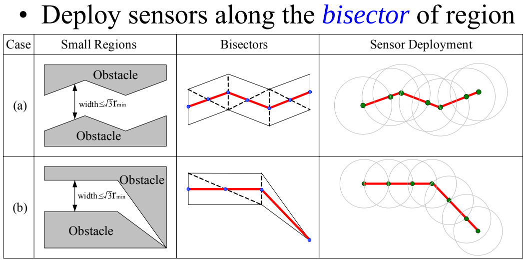


An Example of Partition



Step 2: Place Sensors in a Single-row Region

- Deploy sensors along the *bisector* of region

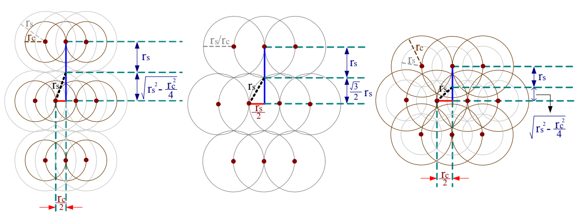


Step 3: Place Sensors in a Multi-row Region

- We first consider a 2D plane without boundaries & obstacles
  - Deploy sensors *row by row*
  - A row of sensors needs to guarantee *coverage and connectivity*
  - Adjacent rows need to guarantee *continuous coverage*
- Case 1: $r_c \le \sqrt{3} r_s$
  - Sensors on each row are separated by r_c
  - Adjacent rows are separated by $r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}$
- Case 2: $r_c > \sqrt{3} r_s$
  - Each sensor is separated by $\sqrt{3} r_s$

Case 1:

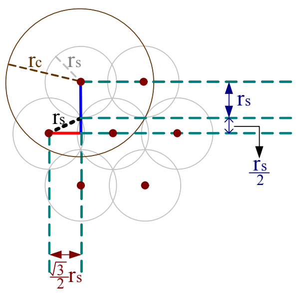
$$r_c \le \sqrt{3} r_s$$



$r_s > r_c$ $r_s = r_c$ $r_s < r_c \le \sqrt{3} r_s$

Case 2:

$$r_c > \sqrt{3} r_s$$

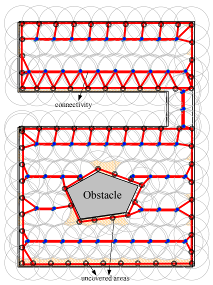


Refined Step 3:

- For a multi-row region with boundaries and obstacles,
  - We can place sensors one by one according to the following locations (if it is not inside an obstacle or outside the region)

Neighbor	$r_c \le \sqrt{3}r_s$	$r_c > \sqrt{3}r_s$
N_1	$(x + r_c, y)$	$(x + \sqrt{3}r_s, y)$
N_2	$(x + \frac{r_c}{2}, y - \sqrt{r_s^2 - \frac{r_c^2}{4}} - r_s)$	$(x + \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2})$
N_3	$(x - \frac{r_c}{2}, y - \sqrt{r_s^2 - \frac{r_c^2}{4}} - r_s)$	$(x - \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2})$
N_4	$(x - r_c, y)$	$(x - \sqrt{3}r_s, y)$
N_5	$(x - \frac{r_c}{2}, y + \sqrt{r_s^2 - \frac{r_c^2}{4}} + r_s)$	$(x - \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2})$
N_6	$(x + \frac{r_c}{2}, y + \sqrt{r_s^2 - \frac{r_c^2}{4}} + r_s)$	$(x + \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2})$

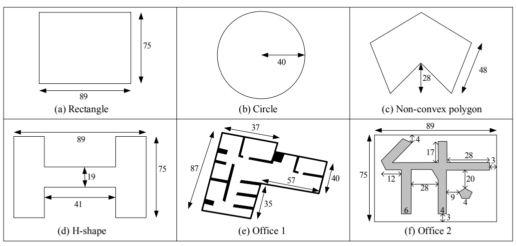
Step 4:



- Three unsolved problems
  - Some areas *near the boundaries* are uncovered
  - Need extra sensors between *adjacent rows* to maintain connectivity when $r_c < \sqrt{3}r_s$
  - Connectivity to *neighboring regions* needs to be maintained
- Solutions
  - Sequentially place sensors along the boundaries of the regions and obstacles

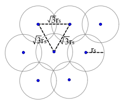
Simulation Results

- Sensing fields

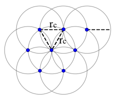


Simulation Parameters

- We use (r_s, r_c) = (7,5), (5,5), (3.5,5), (2,5) to reflect the four cases $r_s > r_c, r_s = r_c, r_s < r_c \le \sqrt{3}r_s, \sqrt{3}r_s < r_c$
- Comparison metric
  - Average number of sensors used to deploy
  - Compare with two deployment methods

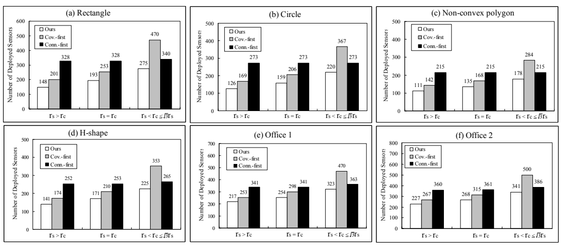


Coverage-first

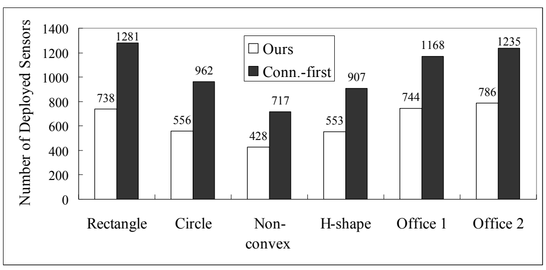


Connectivity-first

Simulations (r_s vs. r_c)



Simulations (Shapes of A)



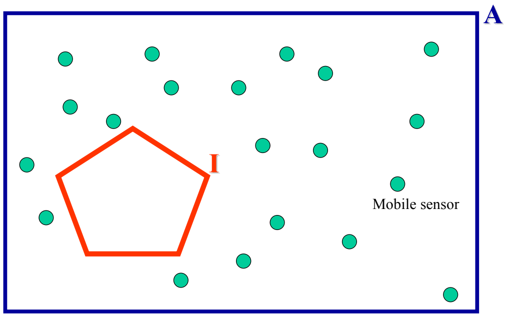
Outline

- Introduction
- Sensor Placement
- Sensor Dispatch
- Conclusions

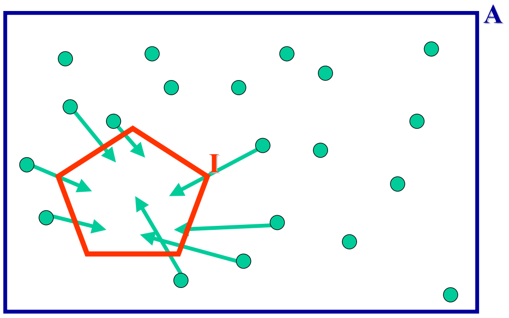
Problem Definition

- We are given
  - A sensing field **A**
  - An area of interest **I** inside **A**
  - A set of mobile sensors **S** resident in **A**
- The sensor dispatch problem asks how to find a *subset* of sensors **S'** in **S** to be moved to **I** such that after the deployment, **I** satisfies *coverage* and *connectivity* requirements and the movement cost satisfies some *objective functions.*

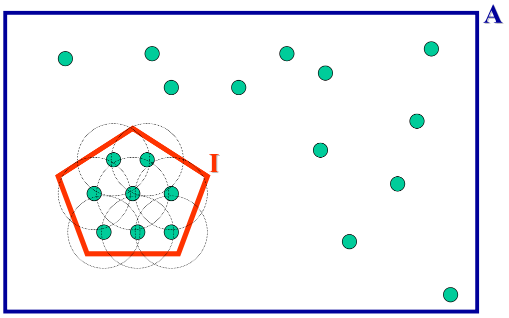
Example



Example



Example



Two Objective Functions

- Minimize the *total energy consumption* to move sensors

  $$\min \sum_{i \in S'} \Delta_m \times d_i$$

  - Δ_m : unit energy cost to move a sensor in one step
  - d_i : the distance that sensor *i* is to be moved
- Maximize the *average remaining energy* of sensors in S' after the movement

  $$\max \frac{\sum_{i \in S'} (e_i - \Delta_m \times d_i)}{|S'|}$$

  - e_i : initial energy of sensor *i*

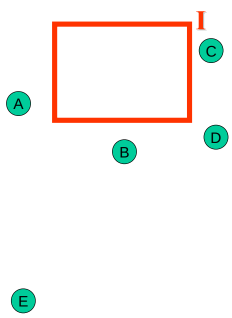
Proposed Dispatch Algorithm (I)

- Run any sensor placement algorithm on **I** to get the target locations $L=\{(x_1, y_1),\ldots,(x_m, y_m)\}$
- For each sensor $s_i \in S$, determine the energy cost $c(s_i, (x_j, y_j))$ to move s_i to each location (x_j, y_j)
  - $c(s_i,(x_j,y_j)) = \Delta_m \times d(s_i,(x_j,y_j))$
- Construct a weighted complete bipartite graph $G=(S\cup L, S\times L)$ such that the weight of each edge is
  - $w(s_i, (x_j, y_j)) = - c(s_i, (x_j, y_j))$, if objective function (1) is used; or as
  - $w(s_i, (x_j, y_j)) = e_i - c(s_i, (x_j, y_j))$, if objective function (2) is used

Proposed Dispatch Algorithm (II)

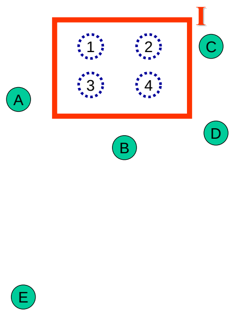
- Construct a new graph $\hat{G}=(S\cup L\cup \hat{L}, S\times\{L\cup \hat{L}\})$ from G, where $\hat{L}$ is a set of $|S|-|L|$ elements, each called a *virtual location*. The weights of edges incident to $\hat{L}$ are set to w_{min}, where w_{min} ={min. weight in G}-1.
- Find the maximum-weight perfect-matching M on graph $\hat{G}$ by using the *Hungarian method.*
- For each edge $c(s_i, (x_j, y_j))$ in M such that $(x_j,y_j)\notin \hat{L}$, move sensor s_i to location (x_j, y_j) via the shortest path.
  - If $e_i - c(s_i,(x_j,y_j)) \le 0$, it means that we do not have sufficient energy to move all sensors. Then the algorithm terminates.

An Example of Dispatch



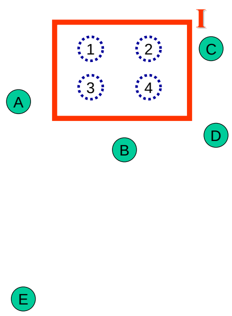
- Initially, there are five mobile sensors A, B, C, D, and E

An Example of Dispatch



- Run sensor placement algorithm on **I** to get the target locations $L=\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$

An Example of Dispatch

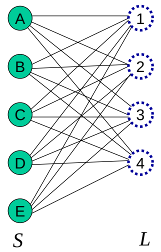


- Compute energy cost (assume $\Delta_m=1$)

$c(A,(x_1,y_1))=9 \quad c(A,(x_2,y_2))=12$
$c(A,(x_3,y_3))=8 \quad c(A,(x_4,y_4))=11$
$c(B,(x_1,y_1))=11 \quad c(B,(x_2,y_2))=11$
$c(B,(x_3,y_3))=9 \quad c(B,(x_4,y_4))=9$
$c(C,(x_1,y_1))=10 \quad c(C,(x_2,y_2))=6$
$c(C,(x_3,y_3))=11 \quad c(C,(x_4,y_4))=8$
$c(D,(x_1,y_1))=14 \quad c(D,(x_2,y_2))=13$
$c(D,(x_3,y_3))=12 \quad c(D,(x_4,y_4))=10$
$c(E,(x_1,y_1))=33 \quad c(E,(x_2,y_2))=35$
$c(E,(x_3,y_3))=30 \quad c(E,(x_4,y_4))=31$

An Example of Dispatch

- Construct the weighted complete bipartite graph G and assign weight on each edge

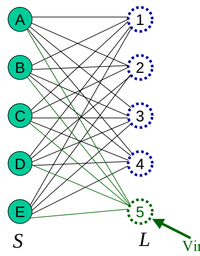


Weights of edges (assume that all sensors have the same initial energy 40 & 1st objective function is used)

	A	B	C	D	E
1	31	29	30	26	7
2	28	29	36	27	5
3	32	31	29	38	10
4	29	31	32	30	9

An Example of Dispatch

- Construct the new graph $\tilde{G}$ from G by adding $|S|-|L|$ virtual locations



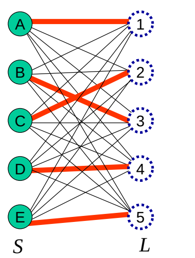
Weights of edges

	A	B	C	D	E
1	31	29	30	26	7
2	28	29	34	27	5
3	32	31	29	28	10
4	29	31	32	30	9
5	4	4	4	4	4

Min.

An Example of Dispatch

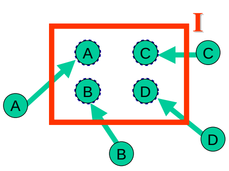
- Use the Hungarian method to find a maximum-weighted perfect-matching M



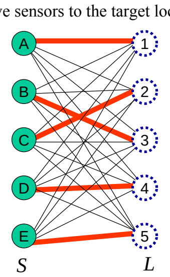
Weights of edges

	A	B	C	D	E
1	31	29	30	26	7
2	28	29	34	27	5
3	32	31	29	28	10
4	29	31	32	30	9
5	4	4	4	4	4

An Example of Dispatch



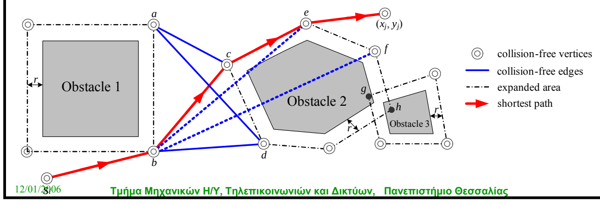
- Move sensors to the target locations



E Do not move

Find the Shortest Distance $d(s_i, (x_j, y_j))$

- Find *collision-free shortest* path
  - A sensor is modeled as a *circle* with a radius *r*
  - Expand the perimeters of obstacles by the distance of *r* to find the *collision-free vertices.*
  - Connect all pairs of vertices, as long as the corresponding edges do not cross any obstacle.
  - Using *Dijkstra's algorithm* to find the shortest path.



Find the Maximum-Weight Perfect-Matching

Definition 1. *Given $\hat{G} = (S \cup L \cup \hat{L}, S \times \{L \cup \hat{L}\})$, a **feasible vertex labeling** of $\hat{G}$ is a real-valued function f on $S \cup L \cup \hat{L}$ such that for all $s_i \in S$ and $(x_j, y_j) \in \{L \cup \hat{L}\}$,*

$$f(s_i) + f((x_j, y_j)) \geq w(s_i, (x_j, y_j)).$$

Definition 2. *Given a feasible vertex labeling of $\hat{G}$, an **equality subgraph** $\hat{G}_f = (S \cup L \cup \hat{L}, E_f)$ is the subgraph of $\hat{G}$ in which E_f contains all edges $(s_i, (x_j, y_j))$ in $\hat{G}$ such that*

$$f(s_i) + f((x_j, y_j)) = w(s_i, (x_j, y_j)).$$

Theorem 1. *Let f be a feasible vertex labeling of $\hat{G}$ and M be a perfect matching of $\hat{G}_f$, then M is a maximum-weight perfect matching of $\hat{G}$.*

The Hungarian Method

Step 1: Find a maximum matching M in $\hat{G}_f$. If M is perfect, we find out the solution and the method terminates. Otherwise, there must be an unmatched vertex $s_i \in S$. We then assign two sets $A = \{s_i\}$ and $B = \emptyset$.

Step 2: In the graph $\hat{G}_f$, if $N_{\hat{G}_f}(A) \neq B$, where $N_{\hat{G}_f}(A)$ is the set of vertices in $\{L \cup \hat{L}\}$ that are adjacent to the vertices in A, then go to step 3. Otherwise, we set

$$\alpha = \min_{s_i \in A,\ (x_j, y_j) \in \{L \cup \hat{L}\} - B} \{f(s_i) + f((x_j, y_j)) - w(s_i, (x_j, y_j))\},$$

and construct a new labeling f' for $\hat{G}$ by

$$f'(v) = \begin{cases} f(v) - \alpha & \text{for } v \in A \\ f(v) + \alpha & \text{for } v \in B \\ f(v) & \text{otherwise} \end{cases}.$$

Then we replace f by f', reconstruct the equality subgraph $\hat{G}_{f'}$, and go to step 1. Note that we have to satisfy the conditions of $\alpha > 0$ and $N_{\hat{G}_{f'}}(A) \neq B$; otherwise, we need to reselect another α value that can satisfy the above conditions.

Step 3: Choose a vertex (x_l, y_l) in $N_{\hat{G}_f}(A)$ but not in B. If (x_l, y_l) is matched with $s_k \in S$ in M, then we update $A = A \cup \{s_k\}$ and $B = B \cup \{(x_l, y_l)\}$, and go back to step 2.

Time complexity

- The time complexity of our sensor dispatch algorithm is $O(mnk^2 + n^3)$
  - m: number of target locations in **I**
  - n: number of mobile sensors
  - k: number of vertices of the polygons of all obstacles and **I**