

# $\epsilon$ -Net Approach to Sensor $k$ -Coverage

**Giordano Fusco and Himanshu Gupta**

Stony Brook University

fusco@cs.sunysb.edu

WASA '09

August 17, 2009

# Outline

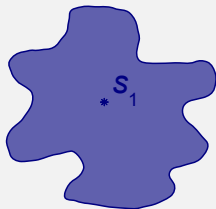
- 1 The problem
- 2 Classical  $\varepsilon$ -net technique
- 3 Extension to  $(k, \varepsilon)$ -net
- 4 Conclusions and open problems

# Outline

- 1 The problem
- 2 Classical  $\varepsilon$ -net technique
- 3 Extension to  $(k, \varepsilon)$ -net
- 4 Conclusions and open problems

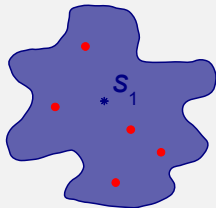
# Sensing regions and target points

- **Sensing regions** can have any *closed* and *connected* shape
- Goal: cover **target points**



# Sensing regions and target points

- **Sensing regions** can have any *closed* and *connected* shape
- Goal: cover **target points**



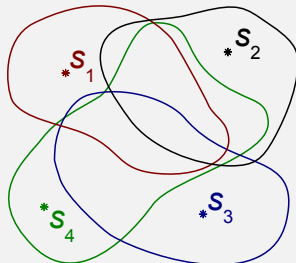
# $k$ -SC problem

## Sensor $k$ -Coverage ( $k$ -SC)

Given sensors

and target points,

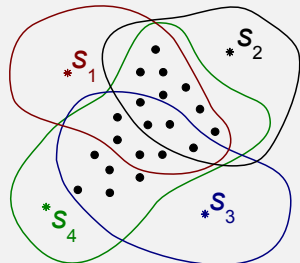
select the **minimum** number of sensors  
to cover each point  $k$  times



# $k$ -SC problem

## Sensor $k$ -Coverage ( $k$ -SC)

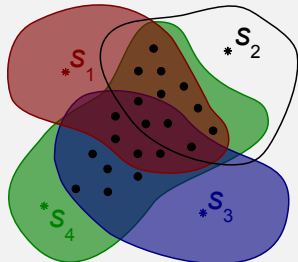
Given sensors  
and target points,  
select the **minimum** number of sensors  
to cover each point  $k$  times



# $k$ -SC problem

## Sensor $k$ -Coverage ( $k$ -SC)

Given sensors  
and target points,  
select the **minimum** number of sensors  
to cover each point  $k$  times



# Motivations

- Higher fault tolerance
- Robustness
- Better position detection
- Better intrusion detection
- Save energy on sleeping sensors
- etc.

# Why $\varepsilon$ -nets?

- Easy to decentralize...  
can be obtained by random sampling
- Our  $\varepsilon$ -net extension gives a  $O(\log M)$ -approximation,  
while the classical greedy algorithm gives  
 $O(k \log n)$ -approximation  
( $M =$  size of optimal solution,  $n = \#$  of target points)

# Outline

- 1 The problem
- 2 Classical  $\varepsilon$ -net technique
- 3 Extension to  $(k, \varepsilon)$ -net
- 4 Conclusions and open problems

# SC and HS

- **Set Cover (SC):** Select the minimum number of **sets** to *cover* all **points**
- **Hitting Set (HS):** Select the minimum number of **points** to *hit*<sup>1</sup> all **sets**
- They are each other's **dual**

<sup>1</sup> A set is *hit* if one of its points has been selected

# $\epsilon$ -nets and weighted $\epsilon$ -nets

## $\epsilon$ -net

**Intuitively:** An  $\epsilon$ -net *hits* all “large-enough” sets

**Formally:** Given a set of points  $X$  and a collection of sets, an  $\epsilon$ -net is a subset of  $X$  that *hits* sets of size  $\geq \epsilon|X|$

## Weighted $\epsilon$ -net

**Intuitively:** Similar concept, but with weighted points

**Formally:** Given a set of weighted points  $X$  and a collection of sets, a **weighted  $\epsilon$ -net** is a subset of  $X$  that *hits* sets of weight  $\geq \epsilon w(X)$

# $\epsilon$ -nets and weighted $\epsilon$ -nets

## $\epsilon$ -net

**Intuitively:** An  $\epsilon$ -net *hits* all “large-enough” sets

**Formally:** Given a set of points  $X$  and a collection of sets, an  $\epsilon$ -net is a subset of  $X$  that *hits* sets of size  $\geq \epsilon|X|$

## Weighted $\epsilon$ -net

**Intuitively:** Similar concept, but with weighted points

**Formally:** Given a set of weighted points  $X$  and a collection of sets, a **weighted  $\epsilon$ -net** is a subset of  $X$  that *hits* sets of weight  $\geq \epsilon w(X)$

# Using $\epsilon$ -nets to solve HS — Why it works

- Given:  $X$  = set of points  
 $\mathcal{C}$  = collection of set
  - **Suppose** we have a black-box to compute weighted  $\epsilon$ -nets
  - **Suppose** we know the optimal hitting set  $H^*$
  - Define the weight function:  $w^*(x) = \begin{cases} 1 & \text{if } x \in H^* \\ 0 & \text{o.w.} \end{cases}$
  - Set  $\epsilon = 1/M$ , where  $M = |H^*|$
- $\Rightarrow$  The black-box will retrieve  $H^*$   
because  $w^*(S) \geq \epsilon w^*(X)$  for all sets  $S \in \mathcal{C}$

# Using $\epsilon$ -nets to solve HS — Practical use

## Ideas of Brönnimann and Goodrich (BG) Algorithm

- Since we don't actually know  $H^*$  or its size  $M$ , we do the following
  - Guess  $M$  (i.e. start with  $M = 1$ , and iteratively double it)
  - For each  $M$ 
    - set  $\epsilon = 1/M$  and find a  $\epsilon$ -net
    - if this  $\epsilon$ -net is a hitting set, then stop  
else, double the weights of all points of a set not hit
    - repeat
  - Doubling the weights helps us converge to  $H^*$
  - It can be shown that only  $O(|X|)$  iterations are needed

# How to obtain weighted $\varepsilon$ -nets

- Weighted  $\varepsilon$ -nets of **size**

$$O\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)$$

can be obtained by **random sampling**,  
if the “*VC dimension*” is constant

- This gives a  $O(\log M)$ -approximate solution  
( $M$  = size of the optimal solution)
- For *disks*, there are methods to build  $\varepsilon$ -net of size  $O(1/\varepsilon)$ ,  
which gives a  $O(1)$ -approximation

# How to obtain weighted $\epsilon$ -nets

- Weighted  $\epsilon$ -nets of **size**

$$O\left(\frac{1}{\epsilon} \log \frac{1}{\epsilon}\right)$$

can be obtained by **random sampling**,  
if the “*VC dimension*” is constant

- This gives a  **$O(\log M)$ -approximate** solution  
( $M$  = size of the optimal solution)
- For *disks*, there are methods to build  $\epsilon$ -net of **size  $O(1/\epsilon)$** ,  
which gives a  **$O(1)$ -approximation**

# Outline

- 1 The problem
- 2 Classical  $\varepsilon$ -net technique
- 3 Extension to  $(k, \varepsilon)$ -net
- 4 Conclusions and open problems

# $k$ -SC and $k$ -HS

- **Sensor  $k$ -Coverage ( $k$ -SC)**: Select the minimum number of **sensors** to  $k$ -cover all **target points**
- **$k$ -Hitting Set ( $k$ -HS)**: Select the minimum number of **points** to  $k$ -hit<sup>1</sup> all **sets**
- They are each other's **dual**

<sup>1</sup> A set is  $k$ -hit if at least  $k$  of its points have been selected

# Weighted $(k, \epsilon)$ -net

**Intuitively:** It *k-hits* all sets with “big-enough” weight

**Formally:** Given a set of weighted points  $X$  and a collection of sets, a **weighted  $\epsilon$ -net** is a subset of  $X$  that *k-hits* sets of **weight  $\geq \epsilon w(X)$**

# Challenges in extending the $\epsilon$ -net result

**Challenge:** Points **cannot** be picked independently at random

**Why:** There might be sets hit by  $< k$  points, if there are duplicates

**Our solution:** Pick a random subset of points **at once** (without duplicates)

**Challenge:** Weights **cannot** be emulated by duplicating points

**Why:** Again, we need  $k$  **distinct** points for each set

**Our solution:** Include the weights directly in the sampling process

# Challenges in extending the $\epsilon$ -net result

**Challenge:** Points **cannot** be picked independently at random

**Why:** There might be sets hit by  $< k$  points, if there are duplicates

**Our solution:** Pick a random subset of points **at once** (without duplicates)

**Challenge:** Weights **cannot** be emulated by duplicating points

**Why:** Again, we need  **$k$  distinct** points for each set

**Our solution:** Include the weights directly in the sampling process

# Centralized $\varepsilon$ -net approach

## Generalized BG Algorithm

- Apart from using  $(k, \varepsilon)$ -nets, the rest of the algorithm remains the same:
  - 1 guess  $M$
  - 2 set  $\varepsilon = k/(2M)$
  - 3 obtain a  $(k, \varepsilon)$ -net
  - 4 change weights if needed
  - 5 repeat
- The challenge lies in proving the approximation ratio

# Distributed $\epsilon$ -net approach

## Challenges

- Distributed randomized selection of  $m$  sensors:

Each sensors keeps an estimate of *total\_weight*, and activates himself with probability

$$m \times \text{own\_weight} / \text{total\_weight}$$

$\Rightarrow$   $m$  sensors are activated an average

- Distributed coverage verification:

Each sensor verifies locally and terminates when he and his neighbors *declare*  $k$ -coverage for 10 consecutive rounds

- Select 1 target point to double the weights:

points are selected with probability  $1/((1 - \epsilon)n)$

$\Rightarrow$  1 point gets selected an average

# Distributed $\epsilon$ -net approach

## Challenges

- **Distributed randomized selection of  $m$  sensors:**  
Each sensors keeps an estimate of *total\_weight*, and activates himself with probability
$$m \times \text{own\_weight} / \text{total\_weight}$$
  
 $\Rightarrow m$  sensors are activated an average
- **Distributed coverage verification:**  
Each sensor verifies locally and terminates when he and his neighbors *declare*  $k$ -coverage for 10 consecutive rounds
- **Select 1 target point to double the weights:**  
points are selected with probability  $1/((1 - \epsilon)n)$   
 $\Rightarrow 1$  point gets selected an average

# Distributed $\epsilon$ -net approach

## Challenges

- **Distributed randomized selection of  $m$  sensors:**  
Each sensors keeps an estimate of *total\_weight*, and activates himself with probability
$$m \times \text{own\_weight} / \text{total\_weight}$$
  
 $\Rightarrow m$  sensors are activated an average
- **Distributed coverage verification:**  
Each sensor verifies locally and terminates when he and his neighbors *declare*  $k$ -coverage for 10 consecutive rounds
- **Select 1 target point to double the weights:**  
points are selected with probability  $1/((1 - \epsilon)n)$   
 $\Rightarrow 1$  point gets selected an average

# Generalization to $k$ -coverage of an area

- Divide the given area into **subregions**
- **Subregion** = set of points covered by the same set of sensors
- Consider each **subregion** as a target point

# Outline

- 1 The problem
- 2 Classical  $\varepsilon$ -net technique
- 3 Extension to  $(k, \varepsilon)$ -net
- 4 Conclusions and open problems

# Conclusions and open problems

## In summary

- We gave a  $O(\log OPT)$ -approximation for sensor  $k$ -coverage, which is an improvement over the classical greedy algorithm

## Related problem

- **Directional sensors** have associated *several* sensing regions, but *only one* can be active at any time (e.g. orienting a camera)
- **$k$ -Coverage with directional sensors** entails both selecting sensors and, for each sensor, selecting the active sensing region
- We gave a greedy algorithm for this problem in SECON '09

## Open problem

- Is it possible to extend the  $\epsilon$ -net technique to  $k$ -coverage with directional sensors?

# Conclusions and open problems

## In summary

- We gave a  $O(\log OPT)$ -approximation for sensor  $k$ -coverage, which is an improvement over the classical greedy algorithm

## Related problem

- **Directional sensors** have associated *several* sensing regions, but *only one* can be active at any time (e.g. orienting a camera)
- **$k$ -Coverage with directional sensors** entails both selecting sensors and, for each sensor, selecting the active sensing region
- We gave a greedy algorithm for this problem in SECON '09

## Open problem

- Is it possible to extend the  $\epsilon$ -net technique to  $k$ -coverage with directional sensors?

# Questions ?

Giordano Fusco

[fusco@cs.sunysb.edu](mailto:fusco@cs.sunysb.edu)