

WPI

RBE 510 – Multi-Robot Systems

Lecture 7: Coverage

Kevin Leahy

September 12, 2025

Combinatorics in the News

[**FEATURE**] Current Issue

The Schedule Tamer

Using AI, John Stewart '97 solves the combinatorial explosion problem of sports scheduling.



orcester Polytechnic Institute

Admin

- HW 2 Out
 - Due at Midnight
 - Questions
- HW 3 posted this afternoon
- Paper presentation 2 on Tuesday
 - Paper is on canvas – read it!

HW2 Pointer

- Algorithm for the programming problem references θ_{smpld}
- This is for systems with continuous dynamics
- Discrete updates require us to sample the states
- *Distributed algorithm in python assumes discrete time dynamics, so no sampling is required*

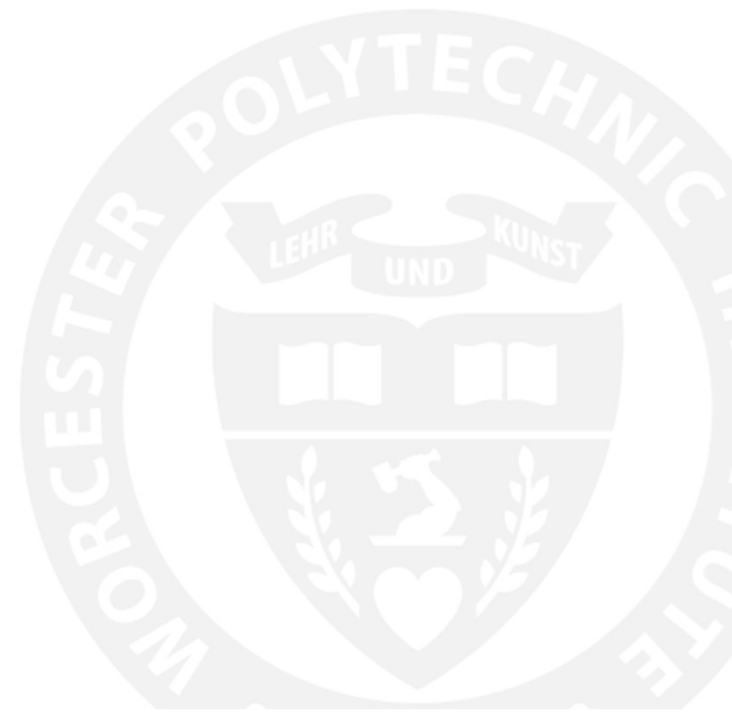
Recap

- Sensor placement
 - Problem definition
 - Complexity
- Modeling sensors and beliefs
- Information theory and mutual information

Today

- Coverage metrics
 - Homogeneous sensor placement
 - Information theory for sensing quality
- Heuristics
 - Submodular set functions (guaranteed greedy approach)
 - Geometric approaches/packing
- Geometric approaches
 - Voronoi diagrams
 - Geometric optimization
 - Graph-based coverage

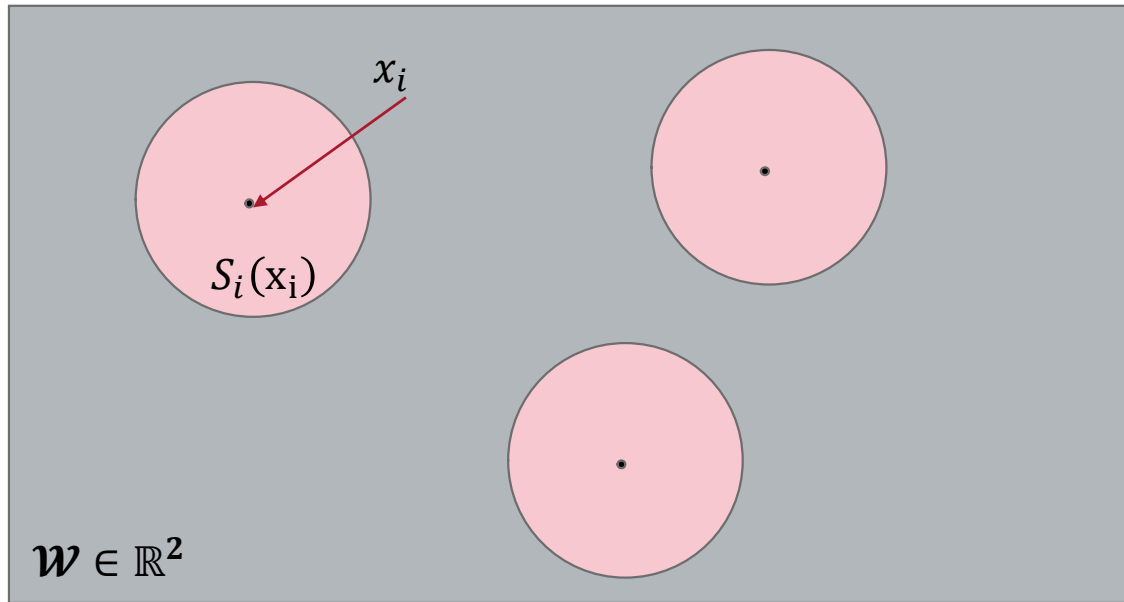
Coverage Metrics



Coverage

- For uniform coverage (i.e., all areas equally “interesting”)
- Coverage criterion is $\frac{\text{Total area of sensor footprints}}{\text{Total ROI}}$
- Can be weighted by “interest”
- Let’s formalize a bit

Problem Setup



- Goal maximize coverage in \mathcal{W}
- Area of \mathcal{W} denoted $A(\mathcal{W})$
- Area covered by sensors is $A[\cup_{i=1}^N S_i(x_i)]$
- Objective function
$$J \triangleq \frac{A[\cup_{i=1}^N S_i(x_i)]}{A(\mathcal{W})}$$

Computing the Coverage

- Consider disjoint sensors

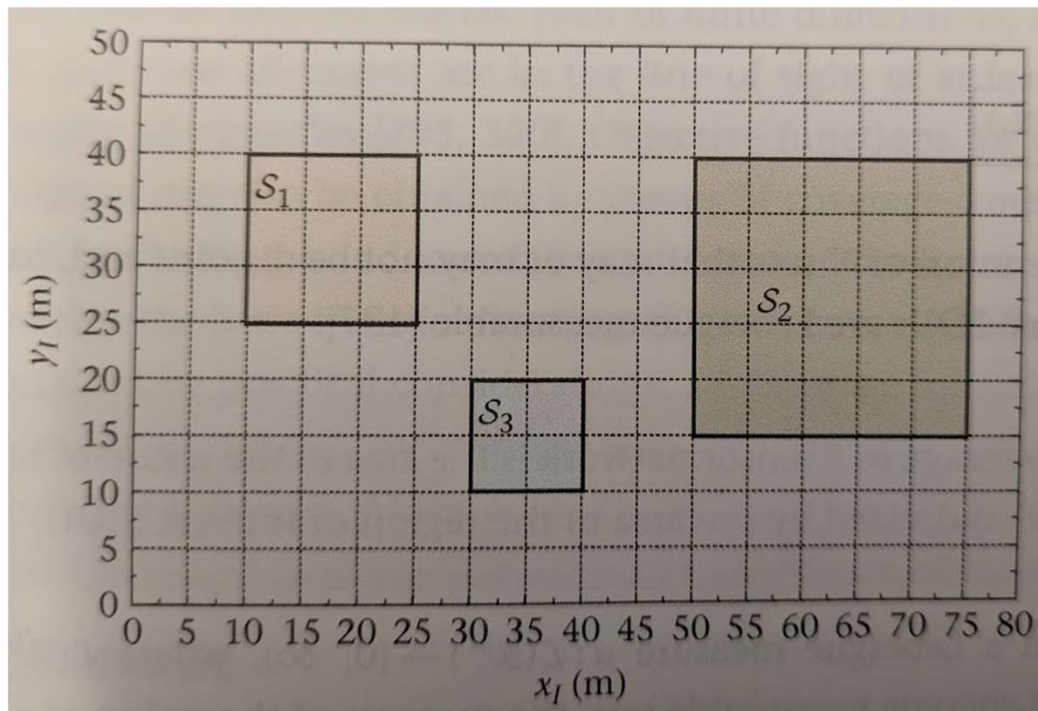
$$S_i(x_i) \cap S_j(x_j) = \emptyset \quad \forall i, j = 1, \dots, N; \quad i \neq j$$

- Then $A\left[\bigcup_{i=1}^N S_i(x_i)\right]$ is trivial

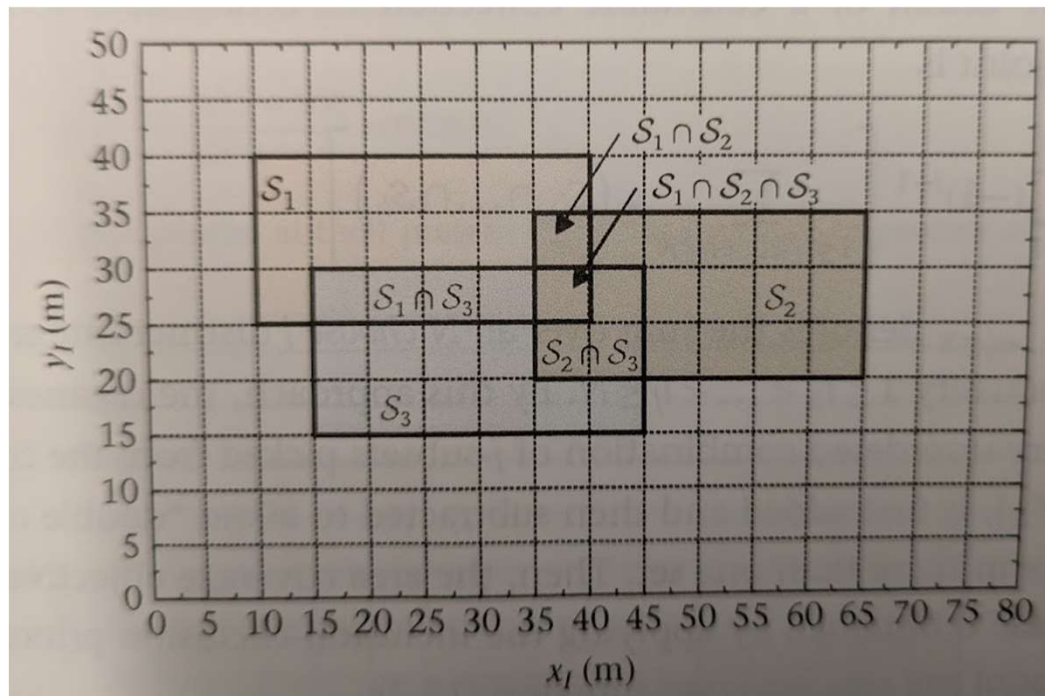
$$A\left[\bigcup_{i=1}^N S_i(x_i)\right] = \sum_{i=1}^N S_i(x_i)$$

- Identically sized sensors, reduces to $NS_i(x_i)$

Example - Disjoint



Example – Not disjoint



Inclusion-Exclusion

- Main idea
 - Add everything up
 - Loop over all of the intersections, and subtract for however many times they were counted

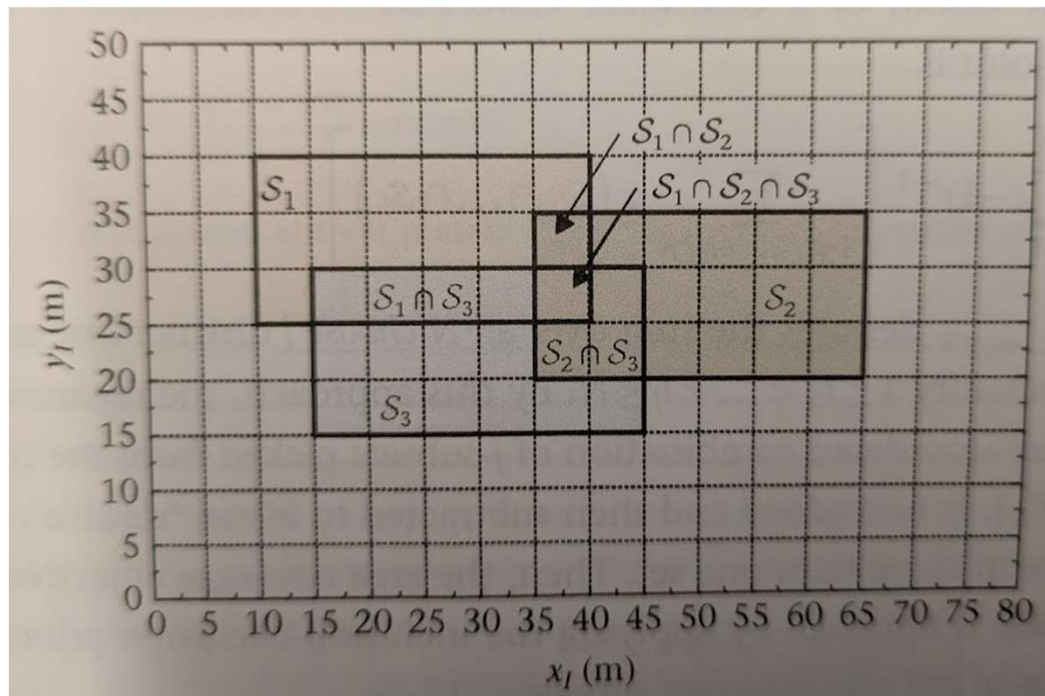
$$A\left[\bigcup_{i=1}^N S_i(x_i)\right] \neq \sum_{i=1}^N S_i(x_i)$$

Inclusion-Exclusion

$$A\left[\bigcup_{i=1}^N S_i(x_i)\right] = \sum_{j=1}^N (-1)^{j+1} \left[\sum_{1 \leq i_1 < \dots < i_j \leq N} A(S_{i_1} \cap \dots \cap S_{i_j}) \right]$$

- This counts N choose j intersections of j possible subsets
- Subset i_j counts the i^{th} subset of intersections of j choices

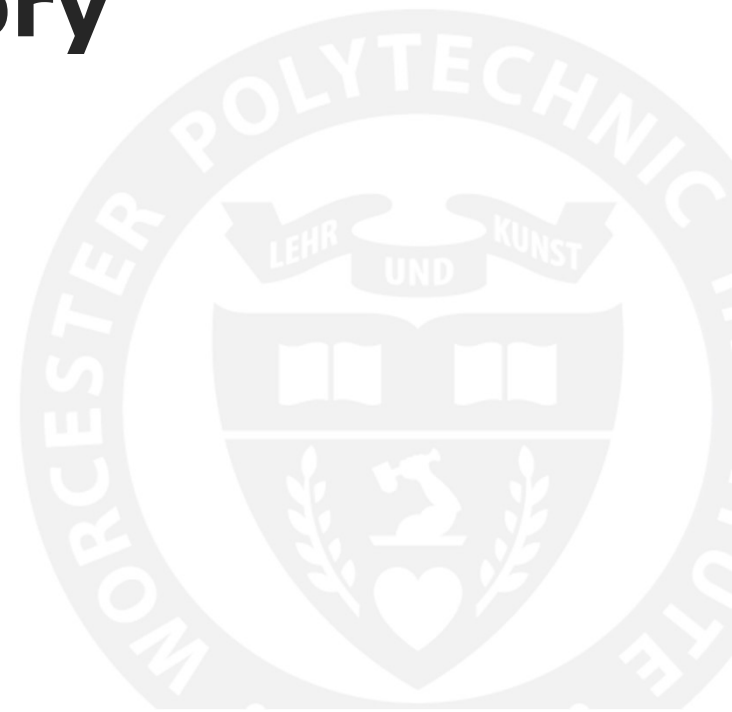
Example Revisited



Scalability

- Computable for “reasonable” number of N
- Increasing intersections blows up combinatorially
- For larger systems or complex areas, we need other approaches
- Let’s look at both through the lens of information theory

Back to Information Theory



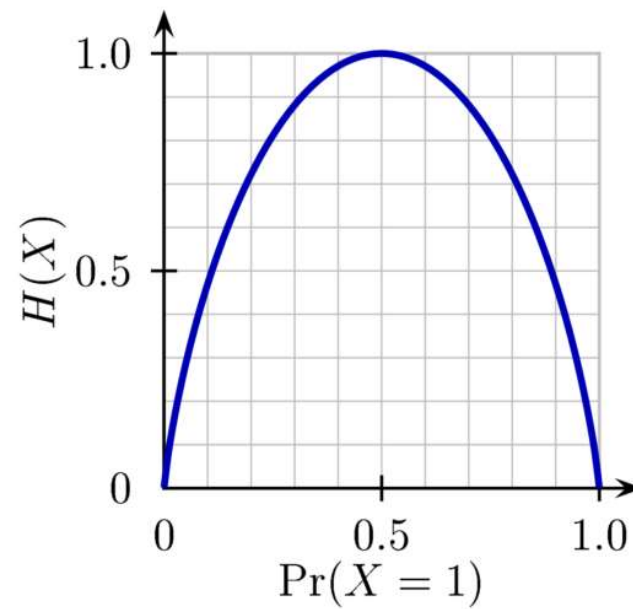
Entropy

- For a discrete R.V. $X \sim p_X(x)$ **entropy** is defined as:

$$H(X) = E[-\log_2(p_{X(x)})] = -\sum_{x \in X} p(x) \log_2 p(x)$$

- Quantifies “randomness” or “uncertainty” or “information content” of PMF
- Measured in bits
- Can also be measured in nats for \log_e or other bases
- N.B. entropy is a function of a *probability distribution*
- N.B.2 this and other information measures can be extended to continuous RVs

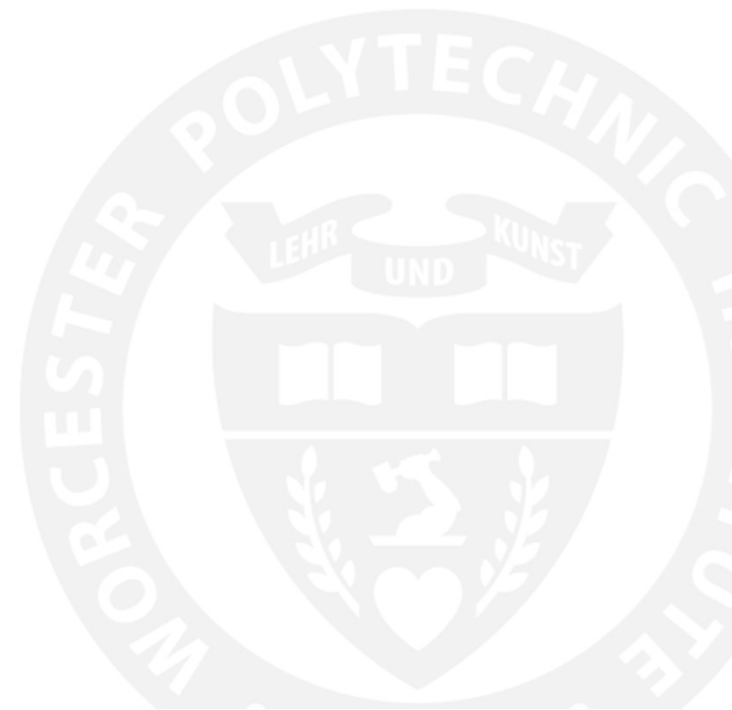
Binary entropy



Some interesting properties

1. $H(X) \geq 0$
2. Maximized for uniform distribution: $H(X) = \log_2 |X|$
3. For uniform distribution, $H(X)$ increases monotonically in $|X|$
4. Any move toward uniformity increases $H(X)$

An additional aside about entropy



Some (hopefully) Helpful Intuition

- 1. Zero-order approximation (symbols independent and equiprobable).
 - XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD
QPAAMKBZAACIBZLHJQD.
- 2. First-order approximation (symbols independent but with frequencies of English text).
 - OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA
NAH BRL.
- 3. Second-order approximation (digram structure as in English).
 - ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE
TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.

Some (hopefully) Helpful Intuition

- 4. Third-order approximation (trigram structure as in English).
 - IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.
- 5. First-order word approximation. Rather than continue with tetragram, ... , n-gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.
 - REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.
- 6. Second-order word approximation. The word transition probabilities are correct but no further structure is included.
 - THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

Some (hopefully) Helpful Intuition

- Zero-order approximation is uniformly random – highest entropy
 - XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD.
- Exactly deterministic is very boring – lowest entropy
 - AA
- “Interesting” entropy is usually in the middle
 - THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

Some (hopefully) Helpful Intuition

Increasing Entropy



The Big Bang
Deterministic
All matter condensed to a
single point
Lowest Entropy
"Boring"



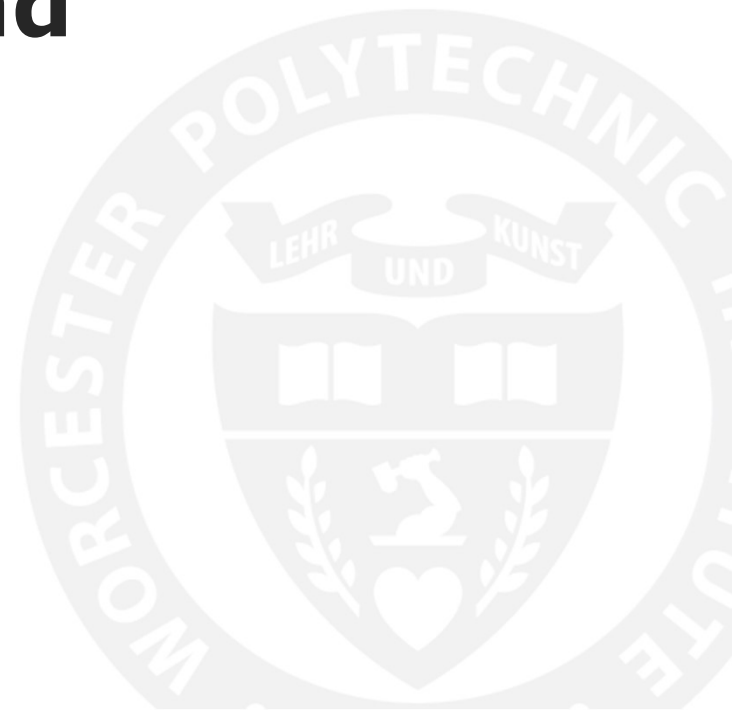
In Between
Interestingly random
Pretty good, all things
considered
Medium Entropy
"Interesting"



The Heat Death of the
Universe
Uniform
Highest Entropy
"Boring"

Worcester Polytechnic Institute

Back to the matter at hand



Joint and Conditional Entropy

- For two RVs, X and Y
- $H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y)$
- $H(Y|X) = \sum_{x \in X} p(x) H(Y|X = x)$
- $H(Y|X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log p(y|x)$
- $H(Y|X) \leq H(Y)$ – Conditioning reduces entropy! (on average)

Mutual Information

- Mutual Information is the relative entropy between joint and marginal PMFs of two RVs, X and Y
- $I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$
- This measure always exists and is non-negative

$$I(X; Y) = H(X) - H(X|Y)$$

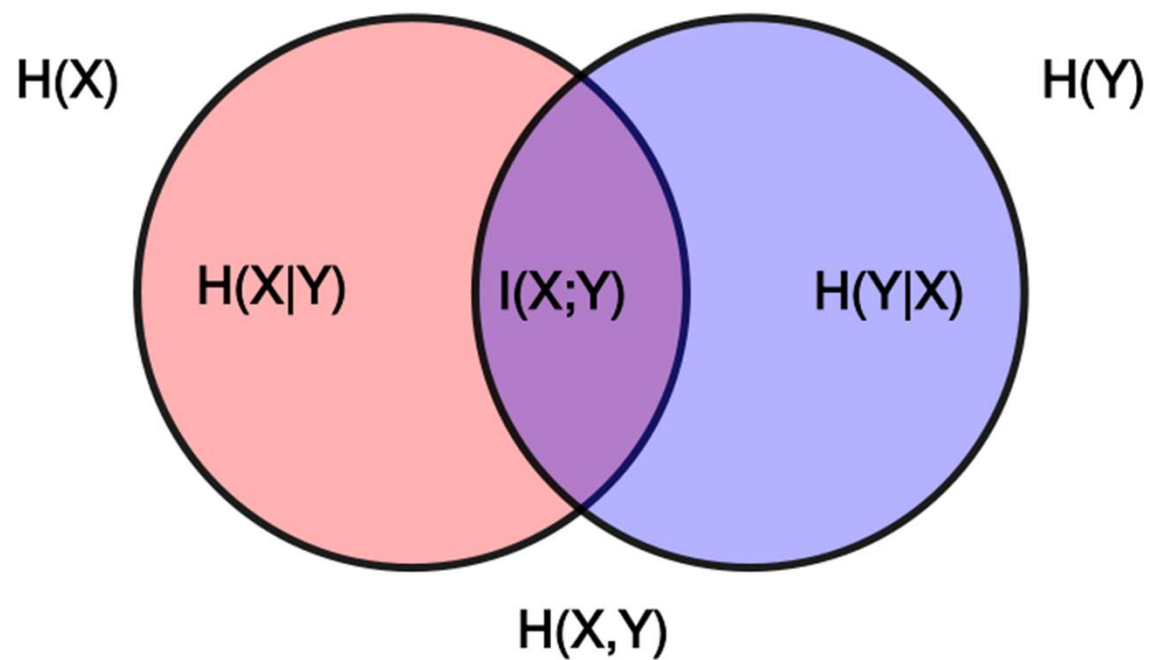
$$I(X; Y) = H(Y) - H(Y|X)$$

$$I(X; Y) = H(X) + H(Y) - H(X, Y)$$

$$I(X; Y) = I(Y; X)$$

$$I(X; X) = H(X).$$

Information Diagram



Numerical Example

$H(X)$

$Y \backslash X$	1	2
	0	$\frac{3}{4}$
1	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

Numerical Example

$Y \backslash X$	1	2
	0	$\frac{3}{4}$
1	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

$H(Y)$

Numerical Example

$Y \backslash X$	1	2
	1	2
1	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

$H(X,Y)$

Numerical Example

$Y \backslash X$	1	2
	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

$$H(X | Y = 1)$$

Numerical Example

$Y \backslash X$	1	2
	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

$$H(X | Y = 2)$$

Numerical Example

$Y \backslash X$	1	2
	0	$\frac{3}{4}$
1	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

$$H(X | Y)$$

Numerical Example

$Y \backslash X$	1	2
	0	$\frac{3}{4}$
1	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

$I(X;Y)$

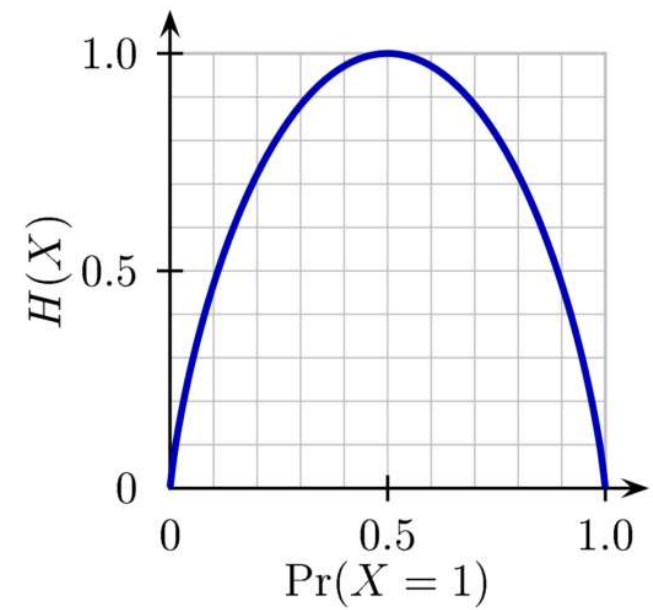
Why do we care?

- Sensor placement can be viewed as a problem of mutual information
- Given a random variable we want to measure, how much we can reduce the uncertainty
- Let's return to sensor placement

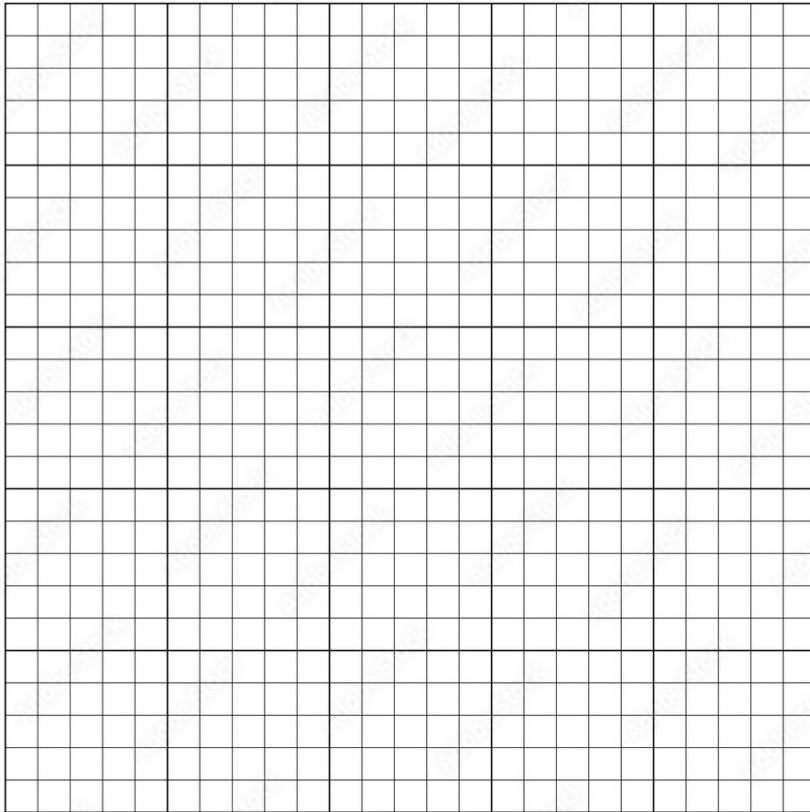
Recall

- Belief forms a simplex
- PMF of a random variable *also* forms a simplex
- Let's use our belief like a probability distribution

Entropy and Belief



Recasting Our Problem



- Environment with finite set of possible locations
- Random variables Ω associated with each location
- Set of k sensors to place in the environment

$$X_k^* = \arg \max_{X \subseteq \Omega, |X|=k} I(X; \Omega \setminus X)$$

- Selecting the “most informative” locations

Entropy Reduction

- A good goal – reduce entropy via our measurements

$$H(X | \lambda) - H(X | Y, \lambda)$$

- Similarly

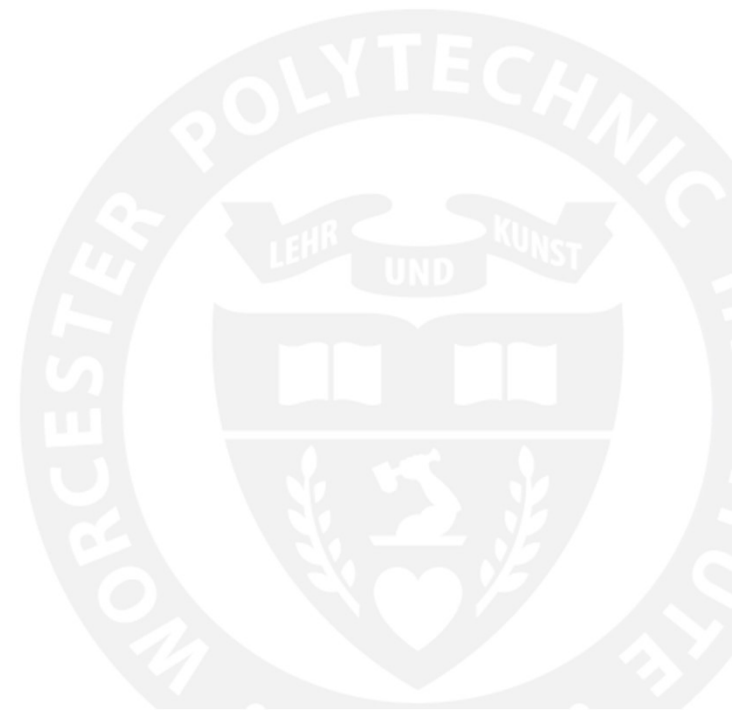
$$I(X; Y | \lambda)$$

- If we have the right probabilities, can compute *expected* entropy reduction given the sensor placement

Computing Expectations

- Compute expectation of belief and entropy – it is complex, esp. if multiple agents/outcomes/locations to choose from!

Submodularity





Submodularity



- Popularized by Krause and Guestrin; Also Stefanie Jegelka, among others

Submodularity in ML



PLATINUM: Semi-Supervised Model Agnostic Meta-Learning using Submodular Mutual Information

Changbin Li*, Suraj Kothawade*, Feng Chen, Rishabh Iyer

ICML 2022
July 2022, Baltimore MD

Deep Submodular Peripteral Networks

Gantavya Bhatt* Arnab M. Das* Jeffrey A. Bilmes
University of Washington, Seattle, WA 98195
{gbhatt2, arnavmd2, bilmes}@uw.edu

Abstract

Submodular functions, crucial for various applications, often lack practical learning methods for their acquisition. Seemingly unrelated, learning a scaling from oracles offering graded pairwise preferences (GPC) is underexplored, despite a rich history in psychometrics. In this paper, we introduce deep submodular peripteral networks (DSPNs), a novel parametric family of submodular functions, and methods for their training using a GPC-based strategy to connect and then tackle both of the above challenges. We introduce newly devised GPC-style “peripteral” loss which leverages numerically graded relationships between pairs of objects (sets in our case). Unlike traditional contrastive learning, or RHLF preference ranking, our method utilizes graded comparisons, extracting more nuanced information than just binary-outcome comparisons, and contrasts sets of any size (not just two). We also define a novel suite of automatic sampling strategies for training, including active-learning inspired submodular feedback. We demonstrate DSPNs’ efficacy in learning submodularity from a costly target submodular function and demonstrate its superiority both for experimental design and online streaming applications.

1 Introduction and Background

This paper jointly addresses two seemingly disparate problems presently open in the machine learning community.

The first is that of identifying a useful practical scalable submodular function that can be used for real-world data science tasks. Submodular functions, set functions that exhibit a diminishing returns property (see Appendix B), have received considerable attention in the field of machine learning. This has fostered new algorithms that offer near-optimal solutions for various applications. These applications include detecting disease outbreaks [56], modeling fine structure in computer vision [41], summarizing images [102, 72, 26, 37], active learning [34, 32, 25], compressed sensing, struc-

Submodularity

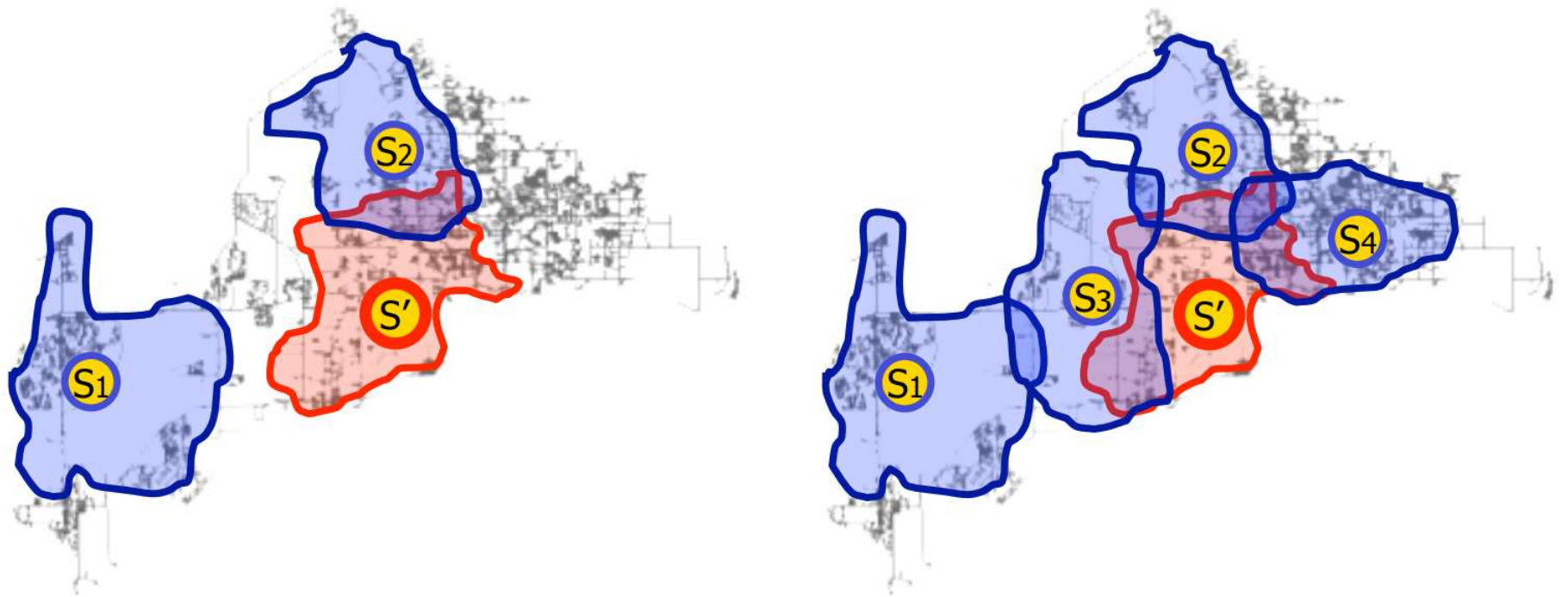
- For a finite set Ω , a set function $f: 2^\Omega \rightarrow \mathbb{R}$ is a **submodular set function** if
 - $\forall X, Y \subseteq \Omega$,
 - $X \subseteq Y$,
 - $\forall x \in \Omega \setminus Y$,

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$$

Submodularity (Equivalent Def.)

- For a finite set Ω , a set function $f: 2^\Omega \rightarrow \mathbb{R}$ is a **submodular set function** if
 - $\forall X \subseteq \Omega$,
 - $\forall x_1, x_2 \in \Omega \setminus X$,
$$f(X \cup \{x_1\}) + f(X \cup \{x_2\}) \geq f(X \cup \{x_1, x_2\}) + f(X)$$

Submodularity Examples



Greedy Maximization

- Start with the empty set $S_0 = \emptyset$
- At i^{th} iteration, select
$$S_i = S_{i-1} \cup \{\operatorname{argmax}_e \Delta(e \mid S_{i-1})\}$$

Submodularity and the Greedy Approach

- Let f be:
 - Submodular over Ω
 - Monotone ($\forall X \subseteq Y \subseteq \Omega, \quad f(Y) \geq f(X)$)
 - $f(\emptyset) = 0$
- Then, choosing A_k (set of k elements chosen greedily) results in
$$f(A_k) \geq \left(1 - \frac{1}{e}\right) f(A_{opt})$$
- Where A_{opt} is the globally optimal selection
- Credited to Nemhauser et al. 1978

Submodular Maximization Intuition

- Proof sketch of greedy suboptimality – details in Nemhauser and Krause and Guestrin

Our objective function

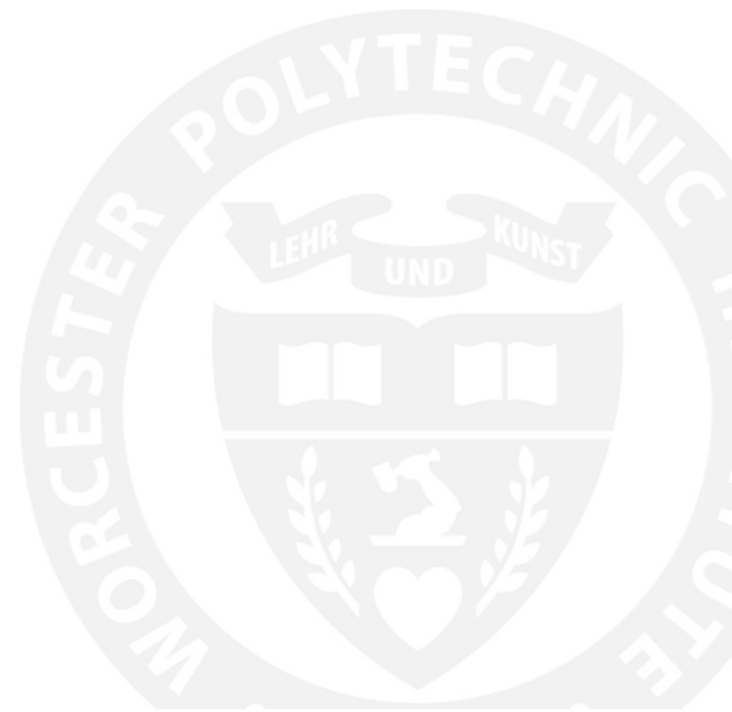
- Is our objective function submodular?
 - Yes!

A tiny little asterisk

- As shown by Krause and Guestrin, entropy, mutual information, etc. are not monotone
- Compromise with a tiny fudge factor

$$f(A_k) \geq \left(1 - \frac{1}{e}\right) (f(A_{opt}) - k\epsilon)$$

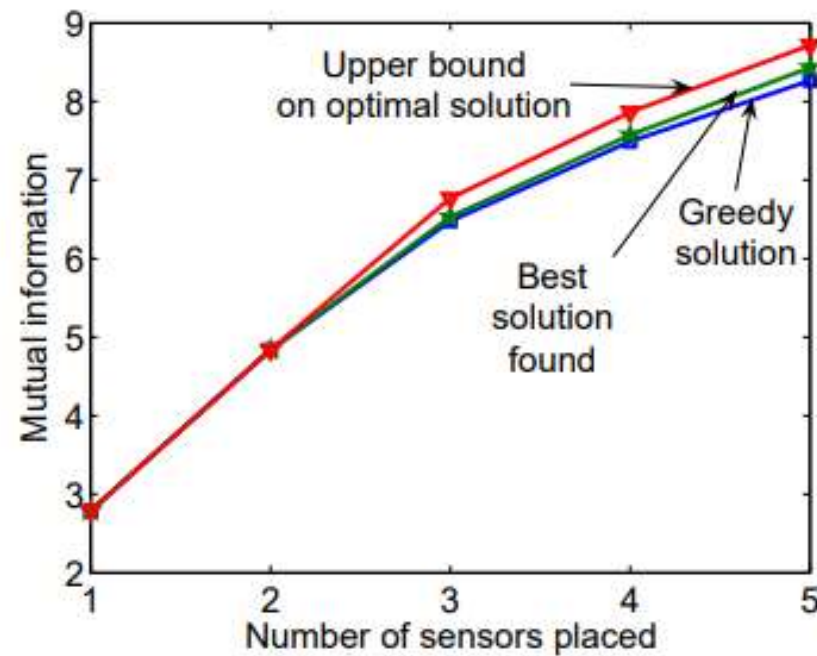
Analysis



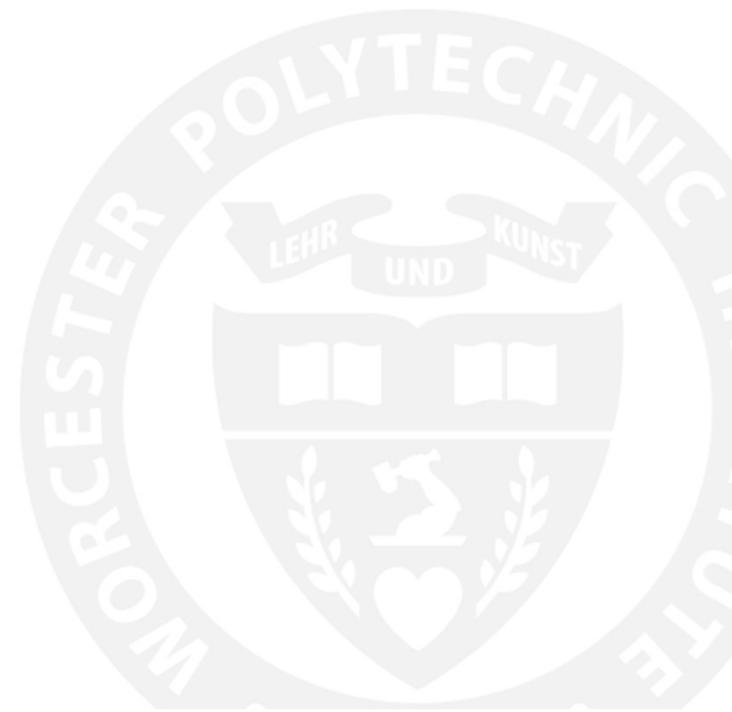
Complexity vs Optimality

- Brute force approach
 - $\binom{\Omega}{k}$ combinations to try
 - But, guaranteed optimal
 - Only feasible for small combinations
- Heuristic approach
 - k locations selected greedily ($O(\Omega k)$, depending on greedy mechanism)
 - Suboptimal within factor of $\left(1 - \frac{1}{e}\right) \approx 63\%$ of optimal
 - Very scalable
- Both approaches are *centralized*

Optimality

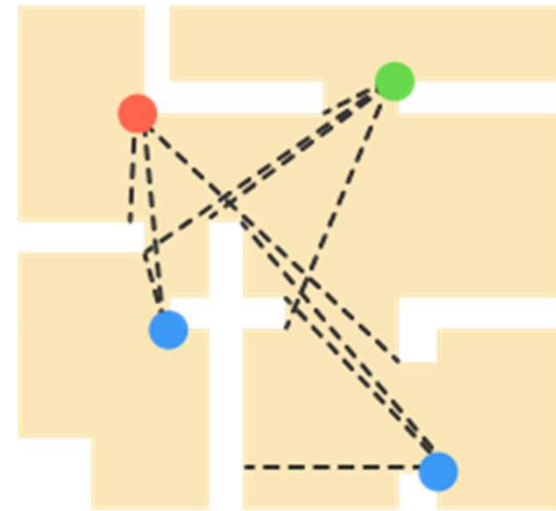


Similar Problems



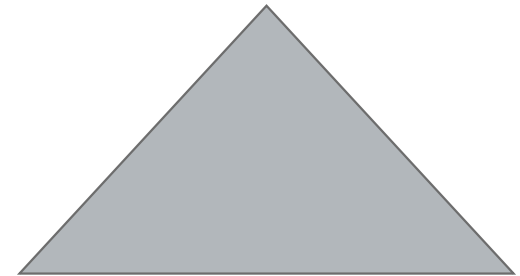
Art Gallery Problem

- Given a floor plan of an “art gallery”
- Agent model with line-of-sight visibility
- What is the minimum number of “guards” you can place that covers the whole area?

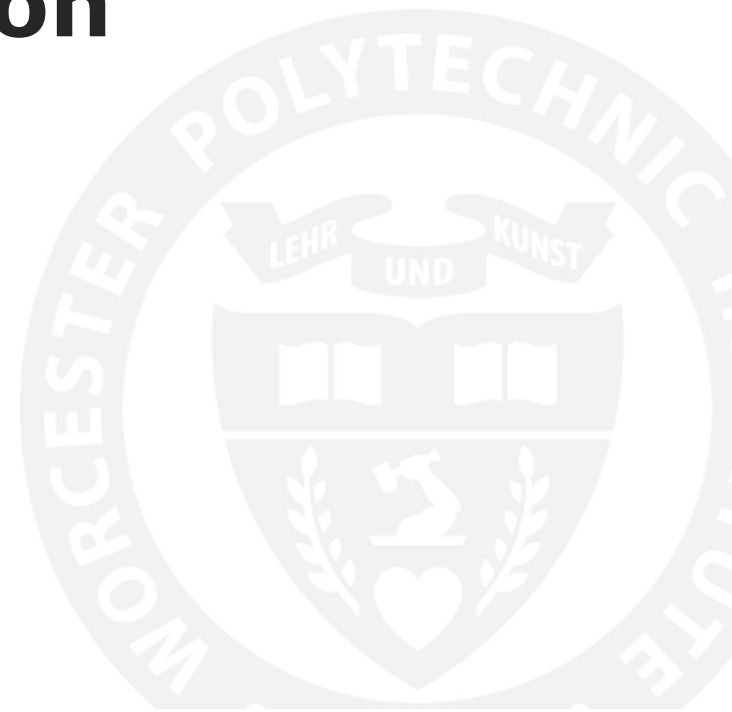


Art Gallery Problem

- Any floor plan with n walls
- At most $\lfloor \frac{n}{3} \rfloor$ guards
- Proved by Chvatal in 1975
- Useful for maintaining connectivity in non-convex environments
- Many variants – range-limited, moving guards (“patrolling”), etc.



Coverage and Triangulation



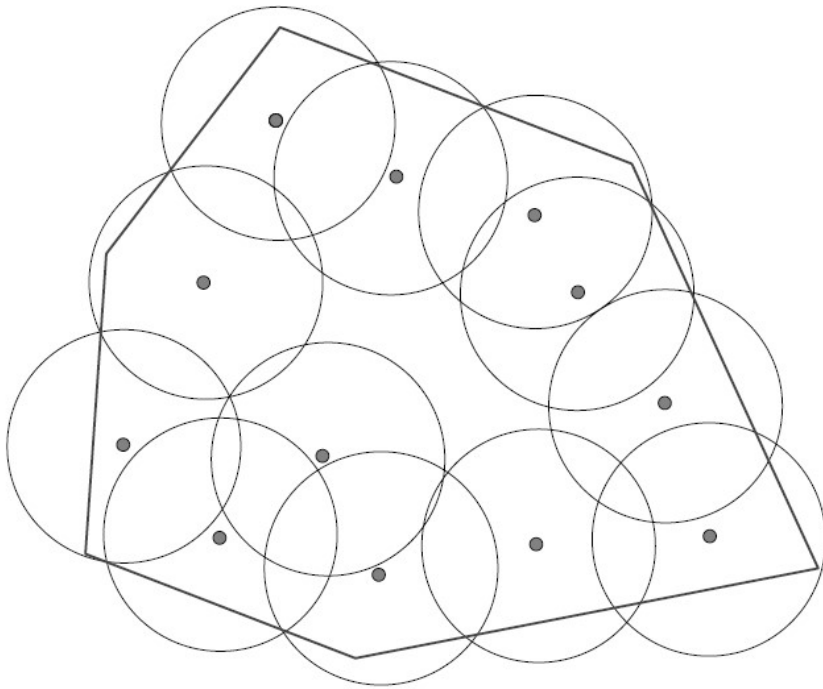
Sources

- Mesbahi and Egerstedt, Ch 7
- Bullo, Cortes, and Martinez, Ch 5
- Lots of great examples/tutorials available
 - S. Martinez, J. Cortes, and F. Bullo. “Motion Coordination with Distributed Information”. *IEEE Control Systems*, 27(4):75-88, 2007.

Previously

- Coverage metrics
 - Area-based
 - Information based
- Solutions
 - Brute-force (combinatorial!)
 - Optimization (still complex)
 - Greedy (suboptimal but scalable)
- All of these approaches are **centralized**
- Can we do anything without centralization?

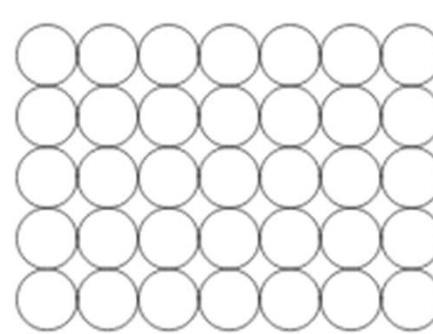
Coverage



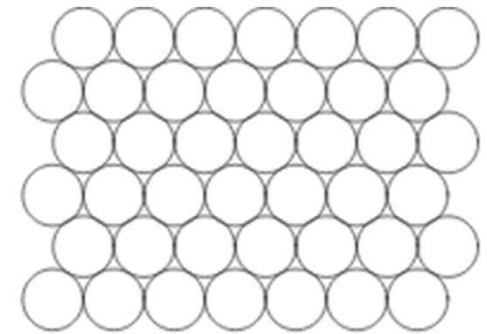
- If we can't cover the whole environment
 - Where is the best set of places to put our limited number of sensors?
 - How can we get them there?

Thinking Geometrically

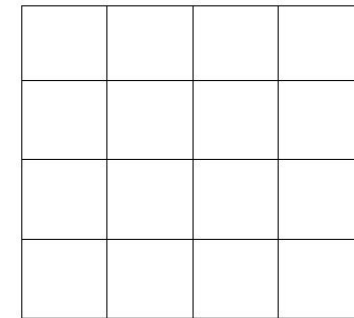
- At the limit, this is a purely geometrical argument
 - What is the geometry of the area we need to cover?
 - Of the sensors we have?
- Given an area and a sensing radius, can we pack them in tightly enough?
- Are there enough agents to cover?
- What if some areas are more important than others?
- Still, graphs can help us out



square packing

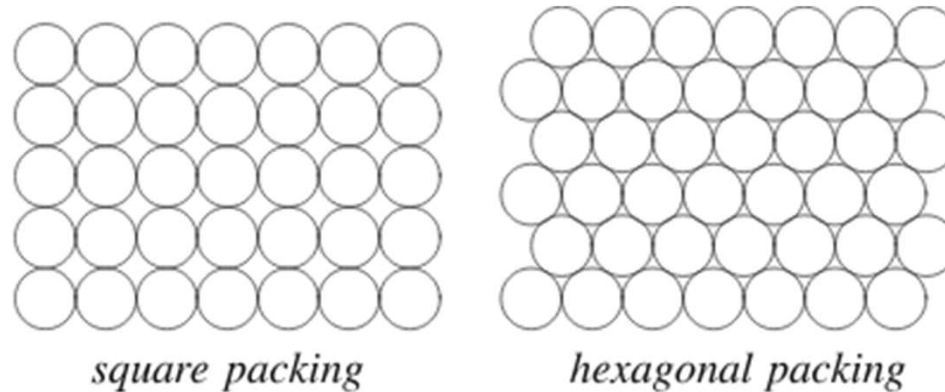


hexagonal packing



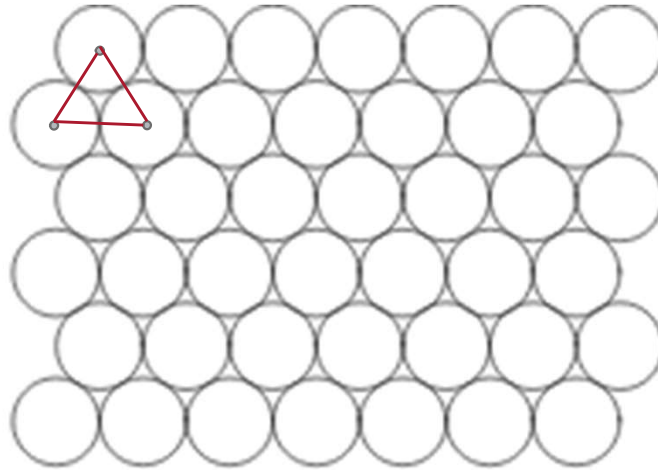
Sphere Packing

- Let's assume arbitrarily large area with finite number of agents
- Agents have circular sensor footprint
- What is the most efficient way to cover the area around them while minimizing overlap?



From Sphere Packing to Graphs

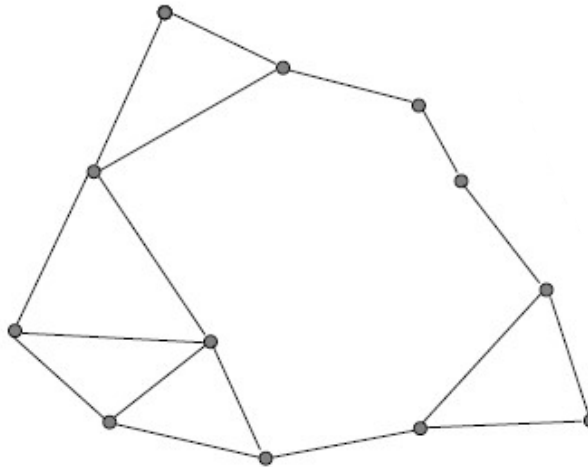
- This suggests some properties of graphs corresponding to good coverage



- **Triangulation** may be a good target for a coordination graph

Triangular Subgraph

- A **triangular subgraph** is any subgraph with three fully connected vertices

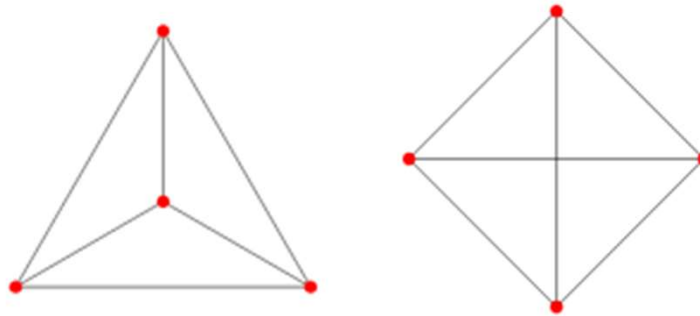


Planar Embedding

- A graph G is **planar** if there is an **embedding**
 $\zeta: V \rightarrow \mathbb{R}^2$

Such that when the edges of G are drawn as straight lines in the plane, none of them intersect

- The pair G, ζ is a **plane graph**

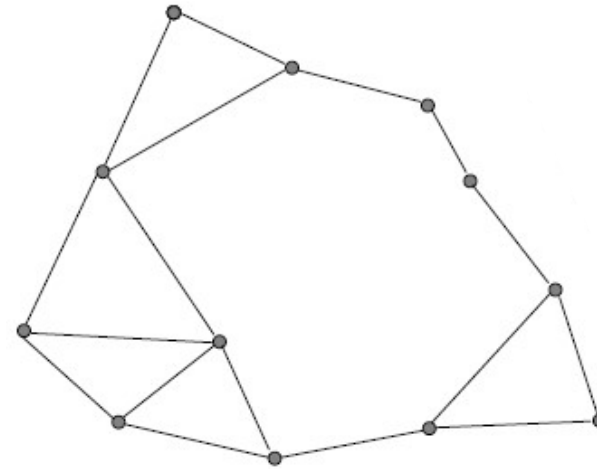
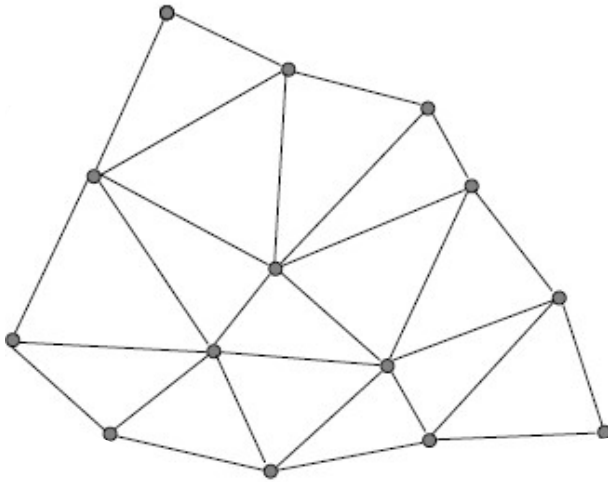


Planar Embedding Example

- $G = (V, E)$
- $V = \{1, 2, 3, 4\}$
- $E = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$
- $\zeta_1: V_1, (1, 1); V_2, (0, 0); V_3, (0, 2); V_4, (1, 0.5)$
- $\zeta_2: V_1, (1, 1); V_2, (0, 0); V_3, (0, 2); V_4, (1, -1)$

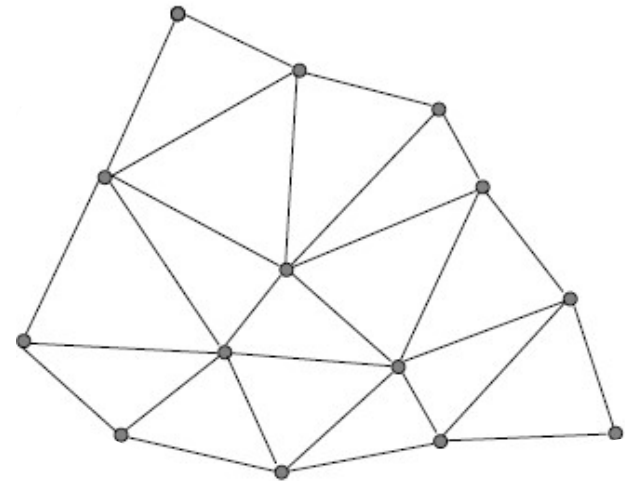
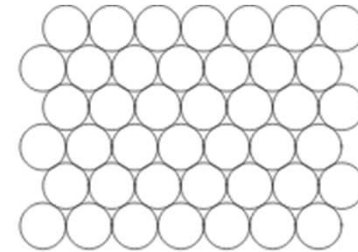
Perfect Plane Triangulation

- A **perfect planar triangulation** is a plane graph
 - Whose outer face is a cycle
 - Inner faces are all triangles



Coverage and Triangulation

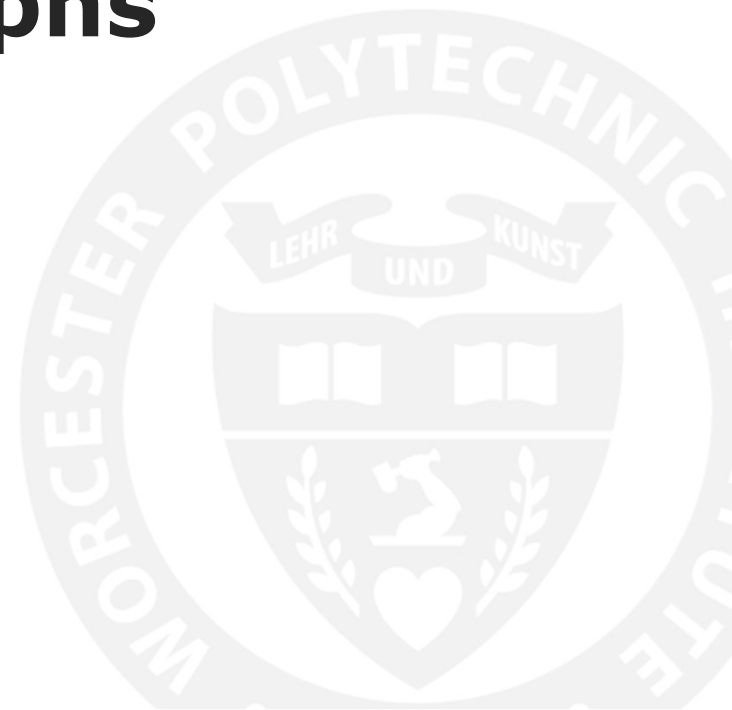
- Seems like a good option for topology and coverage
 - Efficient
 - Connected
- Can we get our agents to create a perfect plane triangulation?



Coverage (Graph-based)

- Given a set of agents tasked with densely covering an area
 - Find a distributed controller such that the agents remain connected
 - The resulting graph is a perfect planar triangulation
- Note: we are looking at this from a graph perspective
- Assumption: we can't cover the whole area so we want dense coverage of what we can sense
- We will solve this problem with **Gabriel graphs**

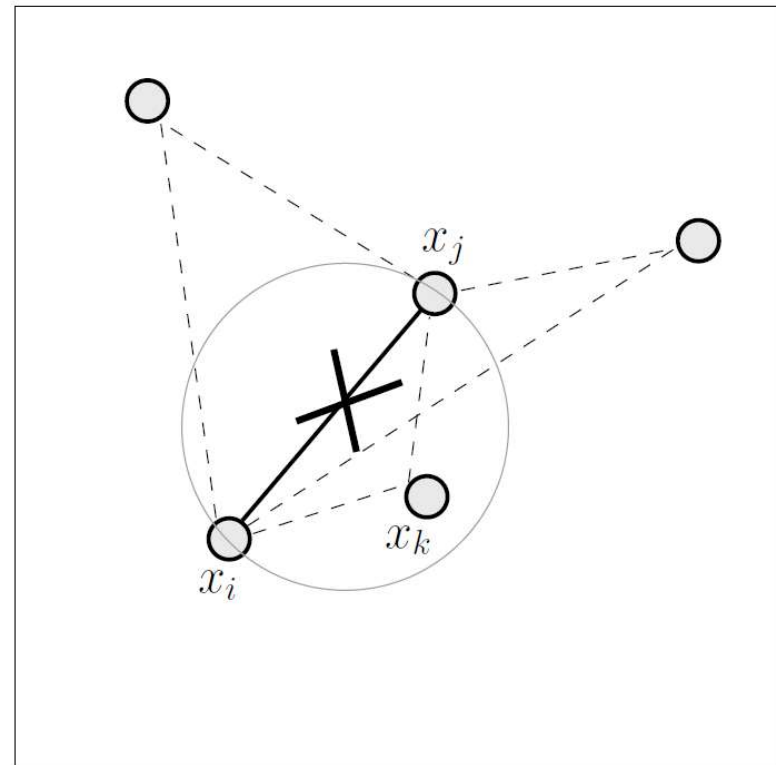
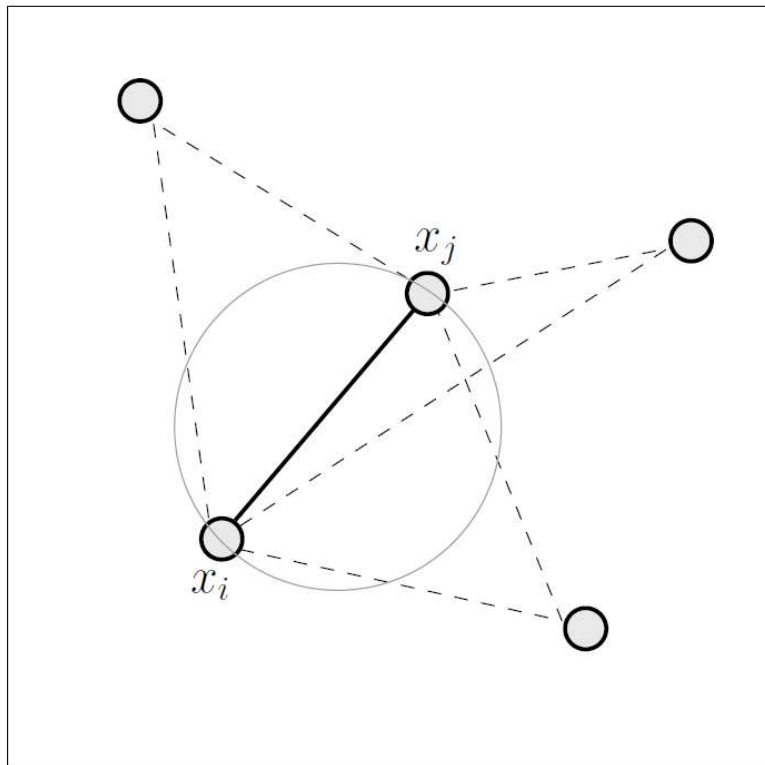
Coverage Via Gabriel Graphs



Solving Graph-based Coverage

- Consider n agents in the plane
- Location of v_i is given by x_i
- Let's *choose* edges between agents such that the resulting graph has the properties we want
- A **Gabriel graph** is a graph where $(v_i, v_j) \in E$ iff $\angle(x_i, x_k, x_j)$ is acute for all other x_k for $k \in \{1, \dots, n\} \setminus \{i, j\}$
- Equivalently, $(v_i, v_j) \in E$ iff the circle of diameter $\|x_i - x_j\|$ that contains x_i and x_j does not contain any other x_k

Gabriel Graphs



Gabriel Graph Properties

- If $\|x_i - x_j\| < \|x_i - x_k\| \forall k \neq i, j$ then $(v_i, v_j) \in E$

Gabriel Graph Properties

- If $(v_i, v_j) \notin E$, then $\exists v_k$ such that $\|x_i - x_k\| < \|x_i - x_j\|$ and $\|x_j - x_k\| < \|x_i - x_j\|$

Gabriel Graph Properties

- Gabriel graphs are always planar

Gabriel Graph Properties

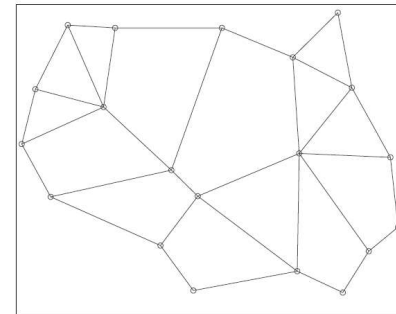
- Gabriel graphs are always connected

Gabriel Graphs, Robots, and Control

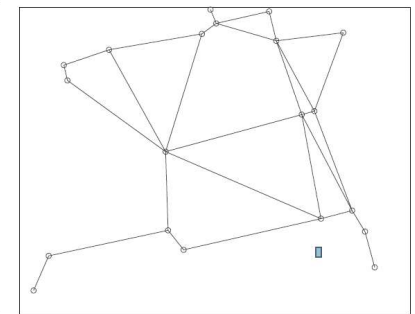
- How to achieve a Gabriel graph?
- Assume agents can determine the relative positions of their neighbors
- Then they have sufficient information to determine a Gabriel graph

Resulting graphs

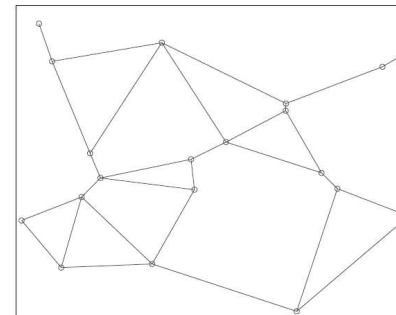
- Gabriel graphs
 - Planar
 - Connected
 - Nearest-neighbor
 - Not good coverage



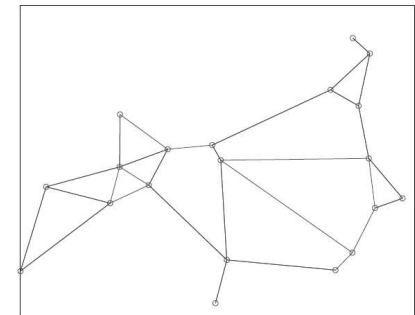
(a)



(b)



(c)



(d)

- How about we move some vertices around?

Formation Control Revisited

- Let's return to the potential-based view of formation control
- We want edges to achieve a *common*, fixed inter-agent distance Δ
- Define potential as

$$\Phi_{ij} = \frac{1}{2} (\|x_i - x_j\| - \Delta)^2 \text{ for all } (i, j) \in E$$

- Corresponding control law is

$$\dot{x}_i(t) = - \sum_{j \in \mathcal{N}_i} \nabla_{x_i} U_{ij} = - \sum_{j \in \mathcal{N}_i} \frac{(\|x_i(t) - x_j(t)\| - \Delta)}{\|x_i(t) - x_j(t)\|} (x_i(t) - x_j(t))$$

Formation Control Revisited

$$\dot{x}_i(t) = - \sum_{j \in \mathcal{N}_i} \nabla_{x_i} U_{ij} = - \sum_{j \in \mathcal{N}_i} \frac{(\|x_i(t) - x_j(t)\| - \Delta)}{\|x_i(t) - x_j(t)\|} (x_i(t) - x_j(t))$$

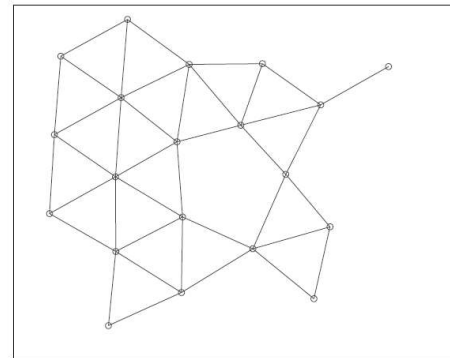
- $SE(N)$ -invariant
- Distributed
- Topology will change over time

Process Overall

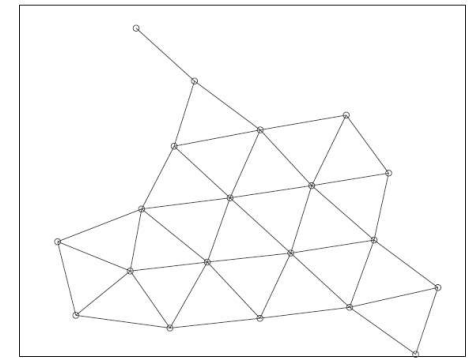
- Create initial Gabriel graph
- While not converged, do:
 - Execute formation controller on current edge set
 - Update edges of Gabriel graph

Results

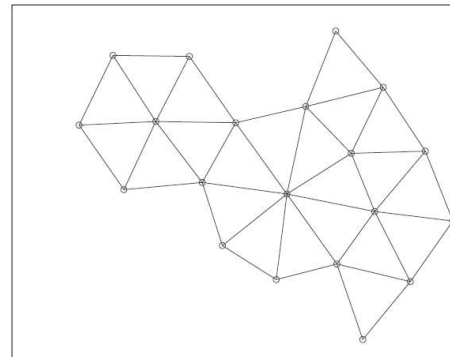
- Almost achieve perfect planar triangulations
- Move beyond nearest-neighbor approach to include requisite long-range connections



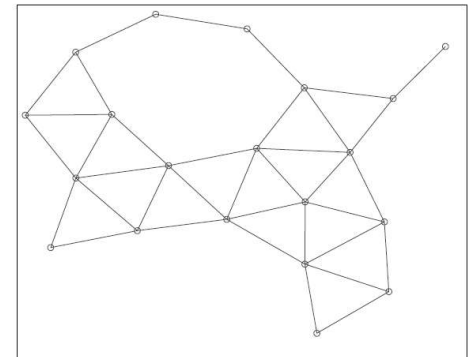
(a)



(b)



(c)

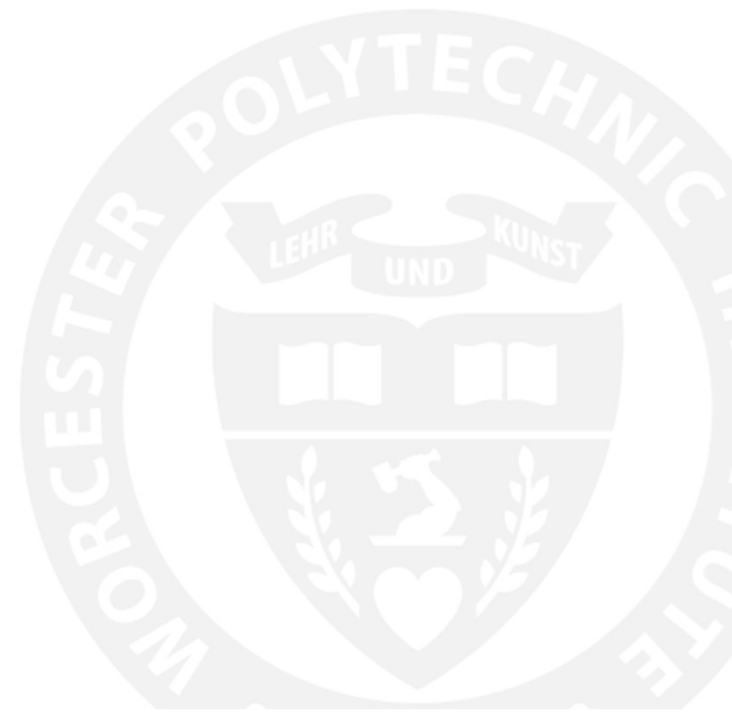


(d)

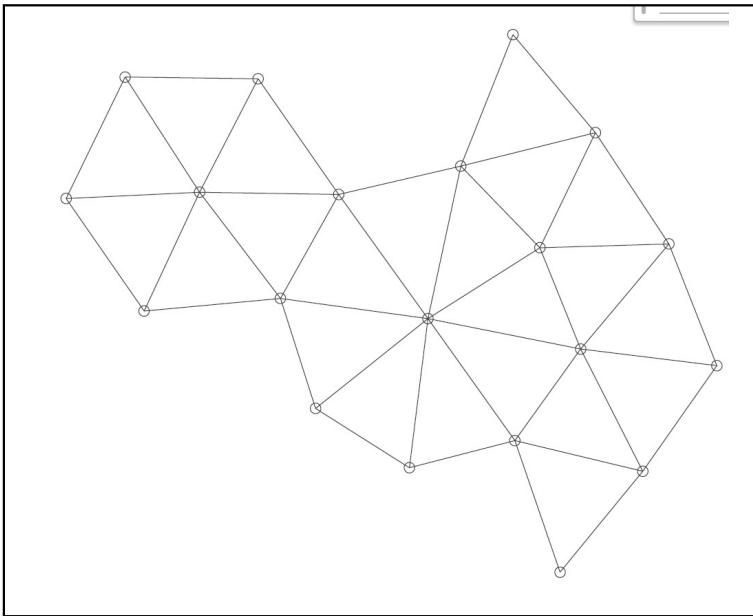
Comparison to static sensor placement

- Static case
 - Works on sets arbitrarily
 - Does not consider dynamics
 - Optimality bound
 - Centralized
- Gabriel coverage
 - Works on planar agents
 - Considers dynamics
 - No notion of optimality or sensing quality
 - Distributed
 - No global reference frame

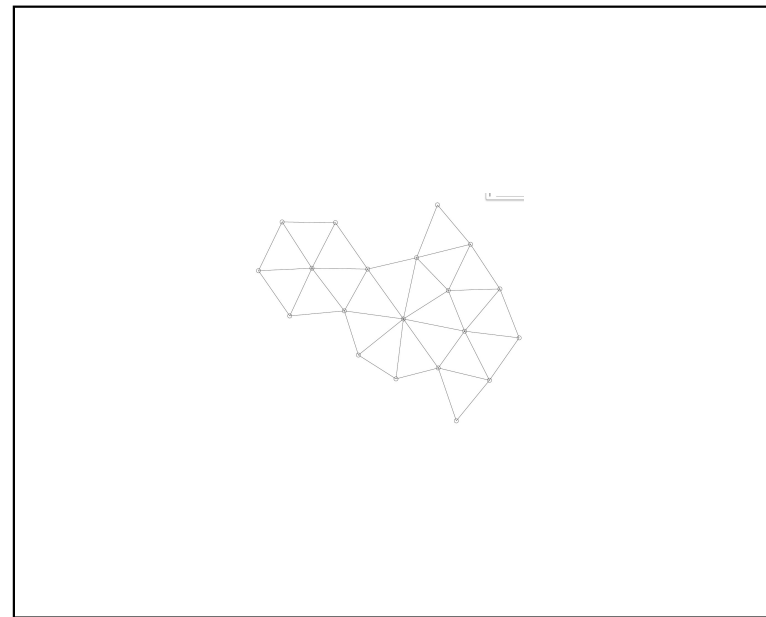
Voronoi Coverage



Covering Large Areas



Dense coverage, based on distance to neighbors



What if we have a much larger area to cover?

Partitioning the Area

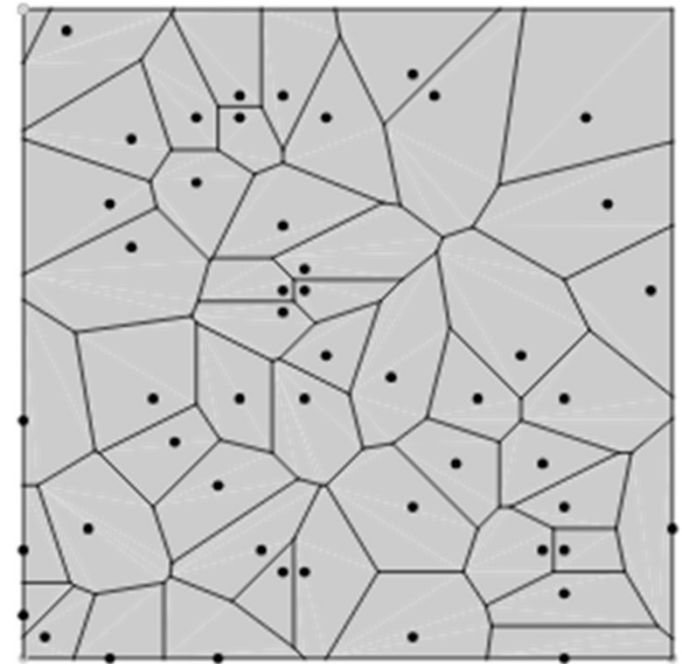


- Area of interest $\Omega \in \mathbb{R}^2$
- Assign each agent i a region W_i such that
- $\bigcup_{i=1}^n W_i = \Omega$
- $W_i \cap W_j = \emptyset, i \neq j$
- This is called a **partition** of Ω

Voronoi Partition/Tessellation

- For agent i , it's Voronoi cell V_i are the points in Ω that are closer to agent i than to any other agent

$$V_i = \{ q \in \Omega \mid \|q - x_i\| \leq \|q - x_j\|, \forall j \neq i \}$$



Voronoi Partition Example

- Given a set of points, let's compute a Voronoi partition
- But our points can move! Where to put them?

Deriving a Controller

- Define our (global) cost as

$$H_V(x) = \min \sum_{i=1}^n \int_{V_i} \|q - x_i\|^2 dq$$

- Gradient with respect to x_i

$$\dot{x}_i(t) = -\frac{\partial H_V(x)}{\partial x_i}$$

Deriving Controller

- Gradient with respect to x_i

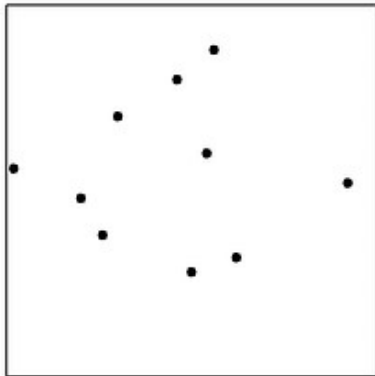
$$\dot{x}_i(t) = -\frac{\partial H_V(x)}{\partial x_i}$$

Deriving a Controller

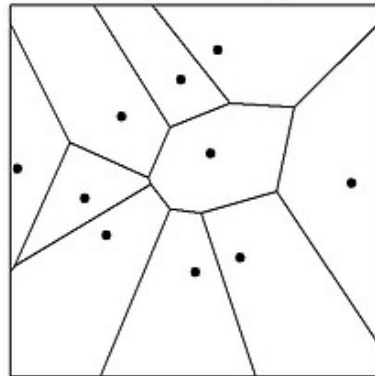
- Note: full proof involves Lyapunov and LaSalle

Example

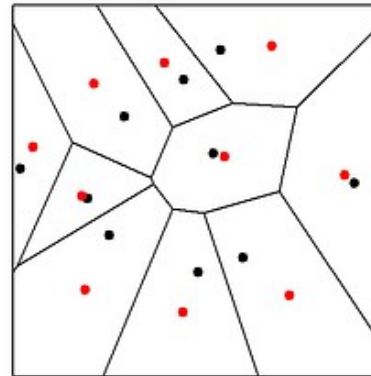
Initiate points



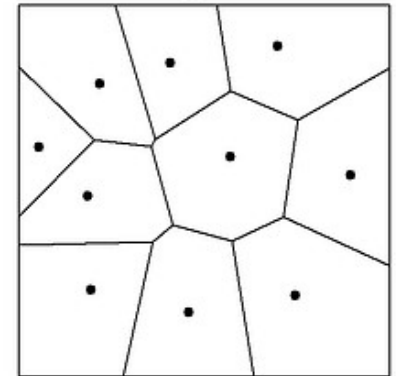
Iteration 0:
Build Voronoi diagram



Find cells' centroids



Iteration 1:
Build Voronoi diagram from centroids



Relation to Other Approaches

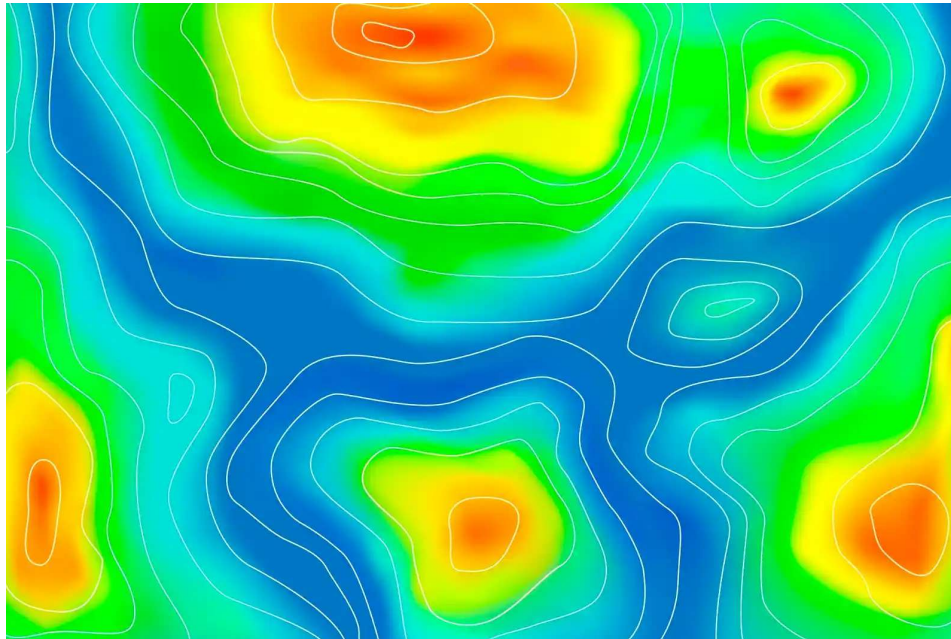
- Closely related to Lloyd's algorithm
- S. Lloyd, "Least squares quantization in PCM," in *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129-137, March 1982, doi: 10.1109/TIT.1982.1056489.

Distributed Voronoi Calculation Example

What Does an Agent Need to Know?

Weighted Voronoi

- What if some areas are more important than others?

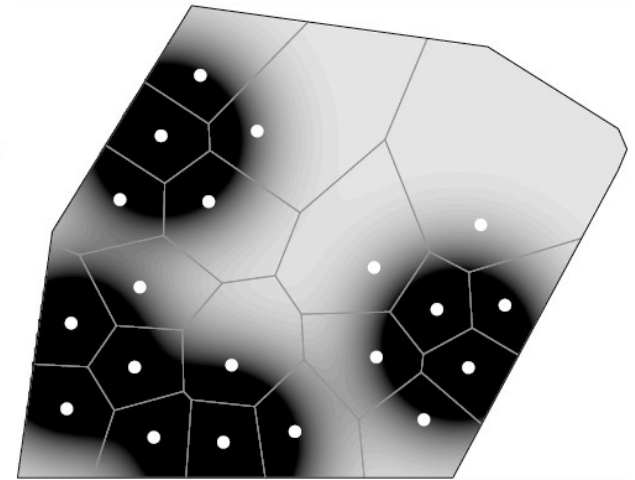
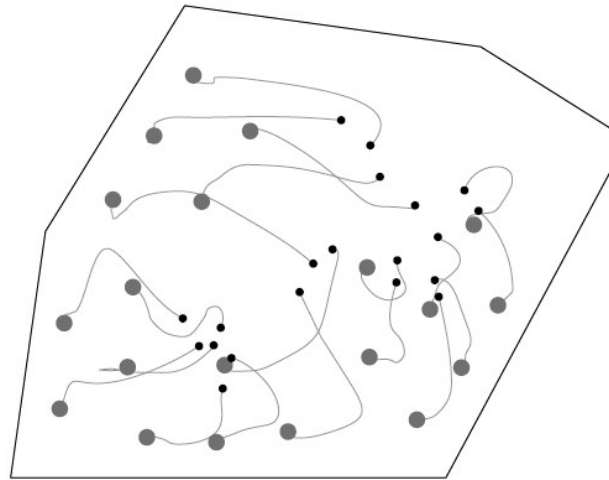
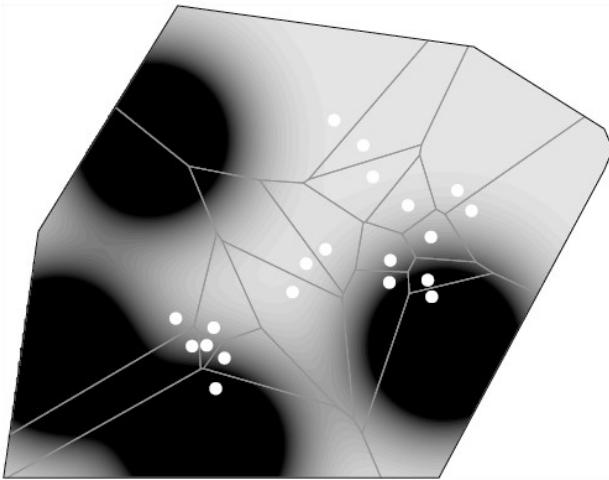


Weighted Voronoi

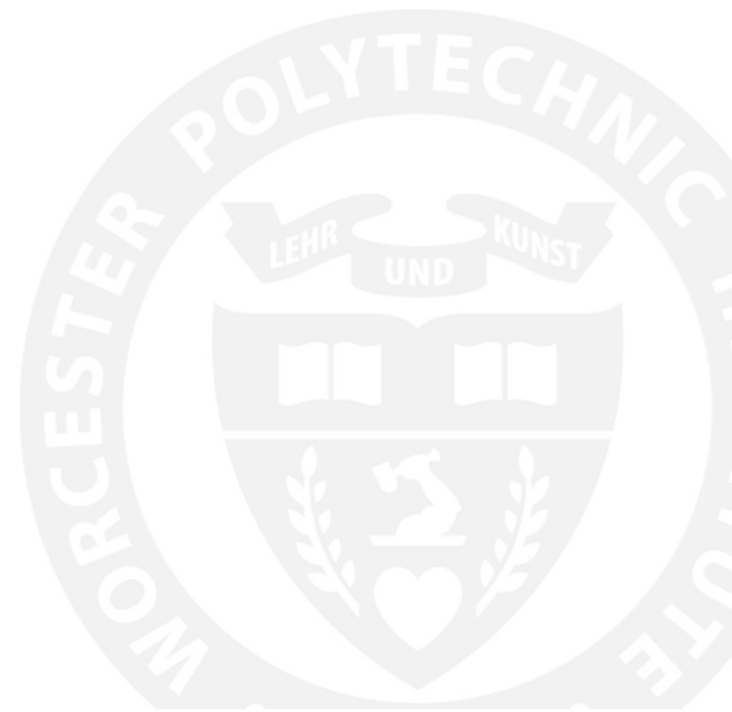
- Change our cost function

$$H(x, V) = \sum_{i=1}^n \int_{V_i} f(\|q - x_i\|) \phi(q) dq$$

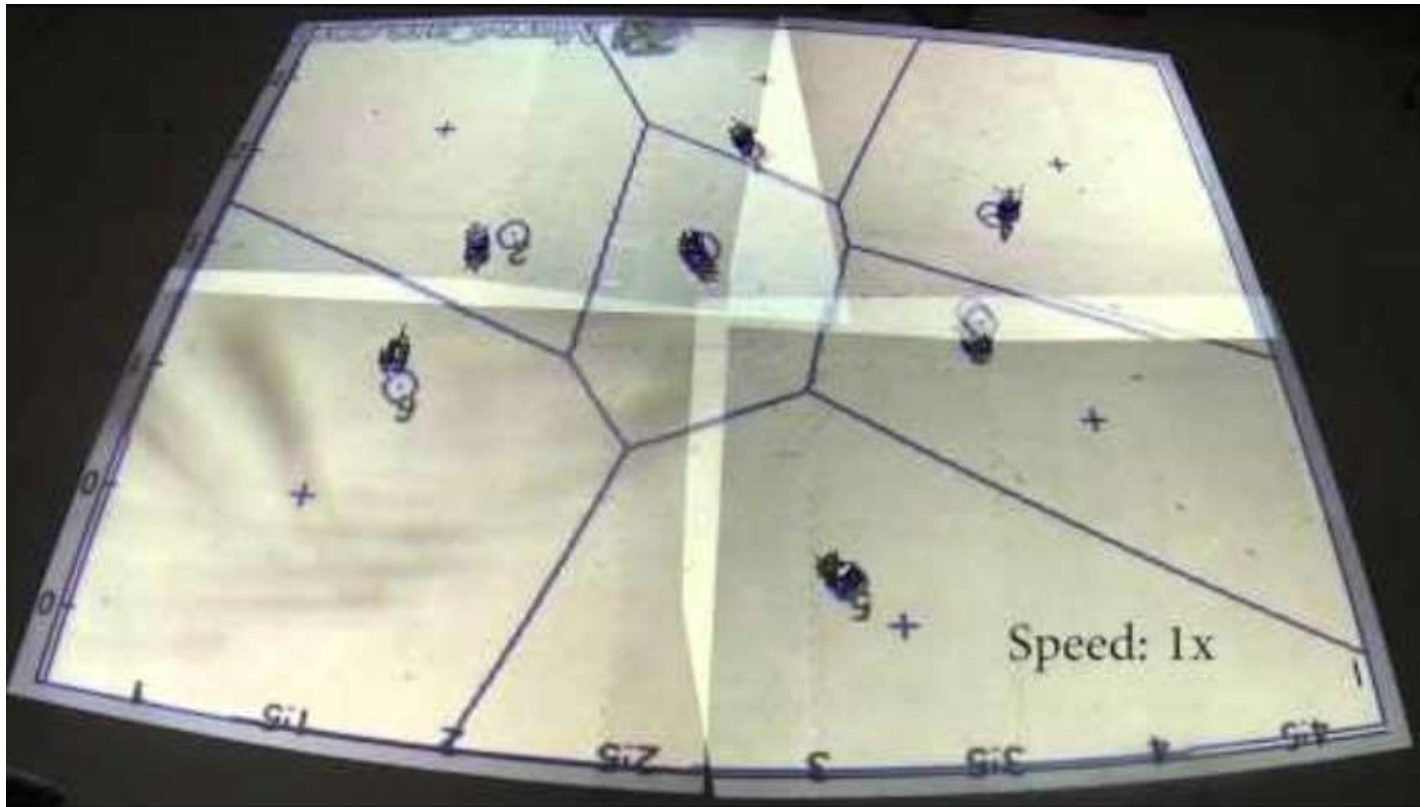
Results



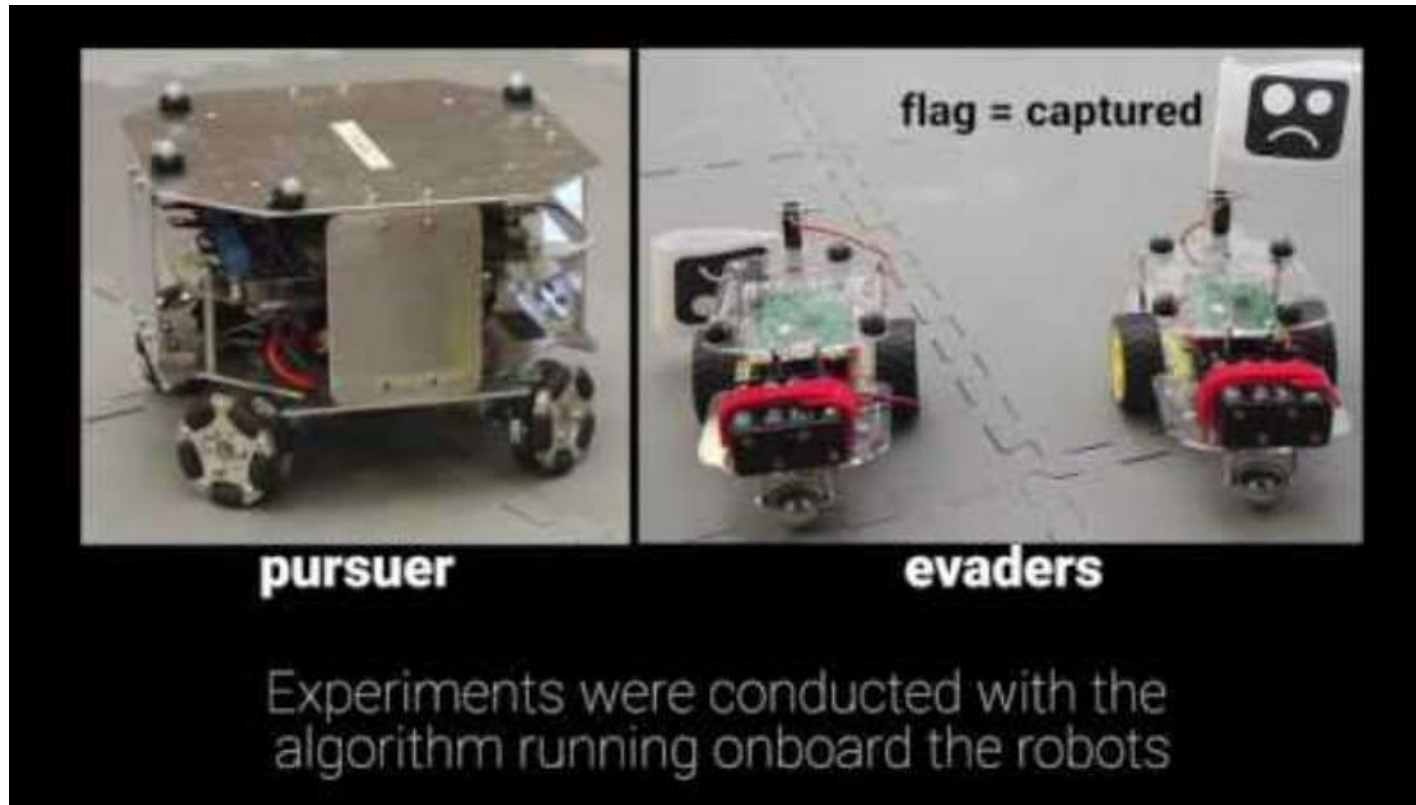
Voronoi Extensions and Applications



Adapting to Robot Performance

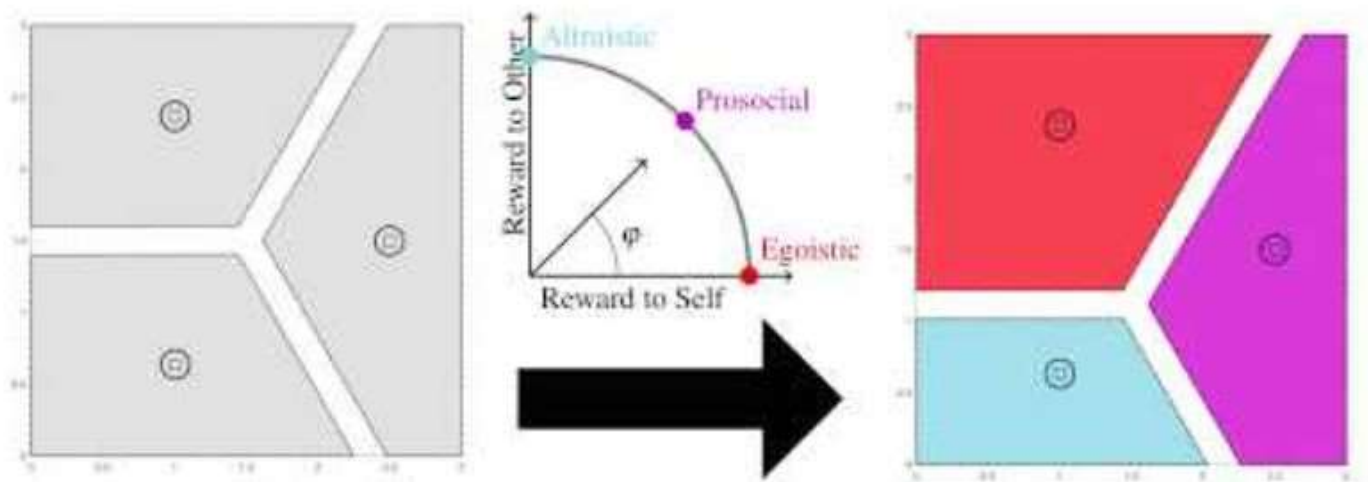


Pursuit Evasion



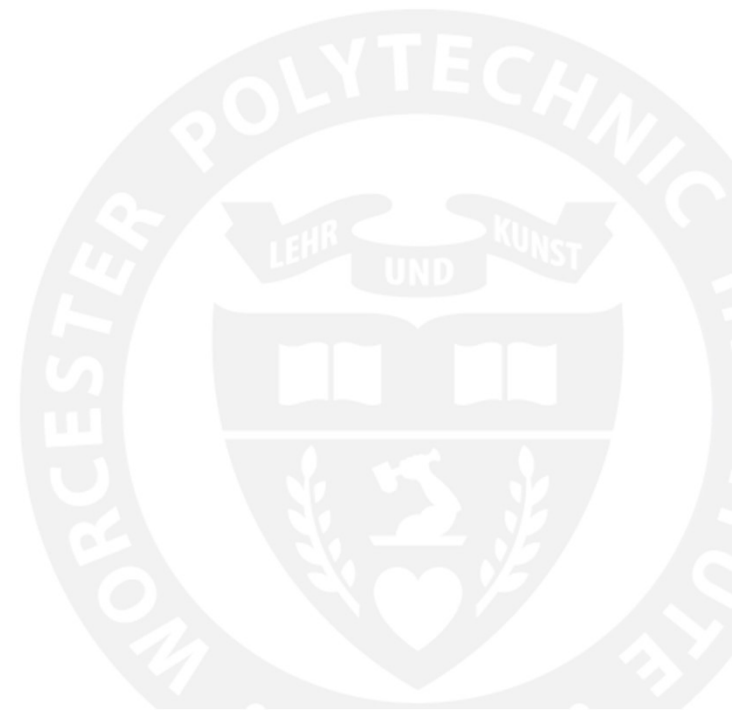
Cooperation and Collision Avoidance

Figure 1



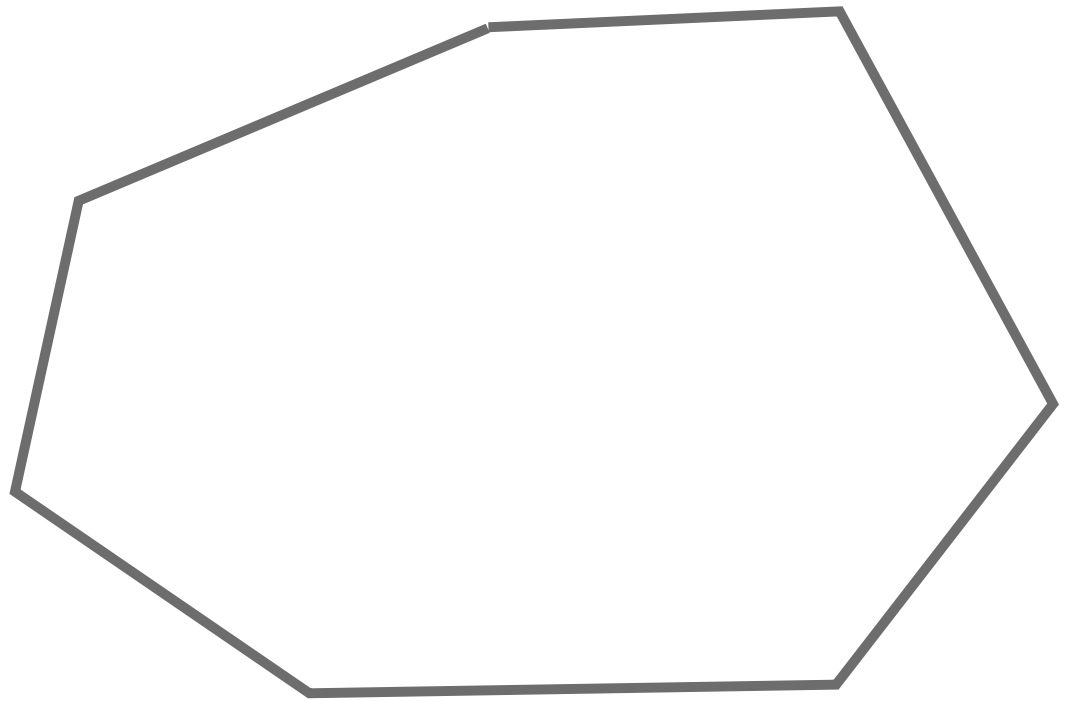
In this paper, we encode Social Value Orientation (SVO) preferences into Weighted Buffered Voronoi Cells (WBVC).

Delaunay Triangulation

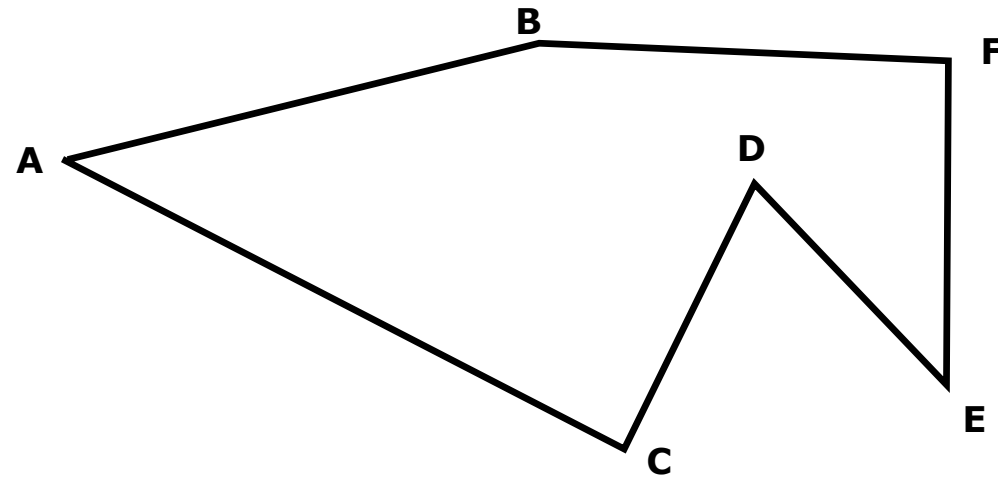


Triangulation

- Partition a polygon P into a set of triangles
- For a convex polygon, this is possible in linear time
- Why might this be unsuitable for robot planning?

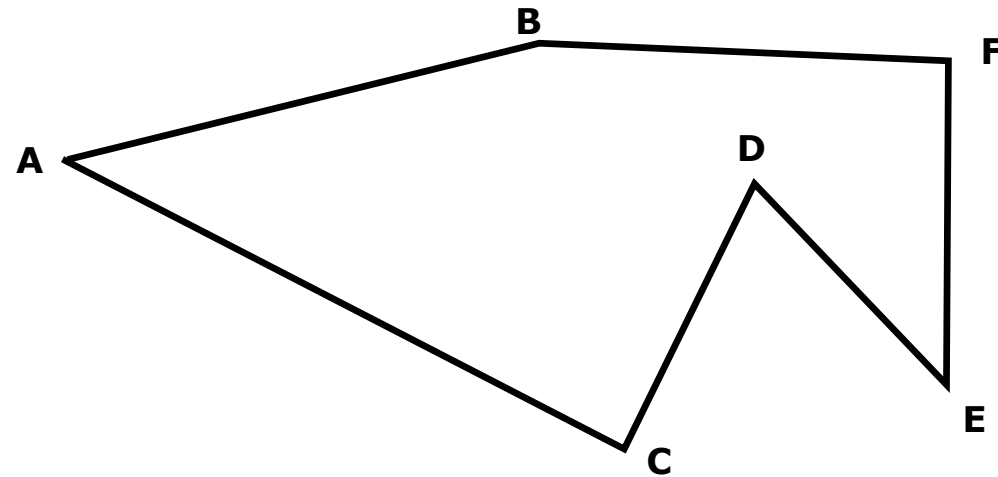


Two Ears Theorem



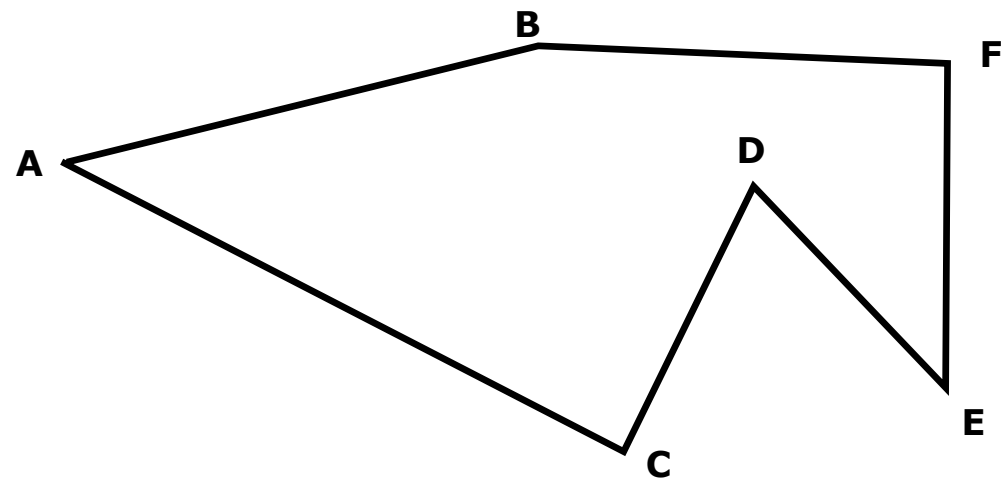
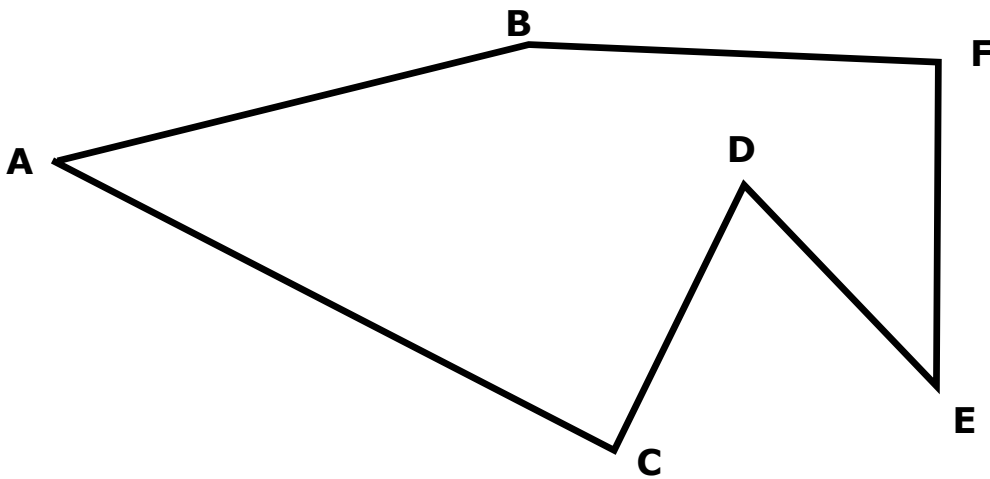
- A vertex is an *ear* if the adjacent vertices can be joined by a line that lies entirely in the polygon
- Two ears theorem states that every simple polygon with more than three vertices has at least two ears
- How can we use this theorem for triangulation?

Triangulation via Ear Clipping



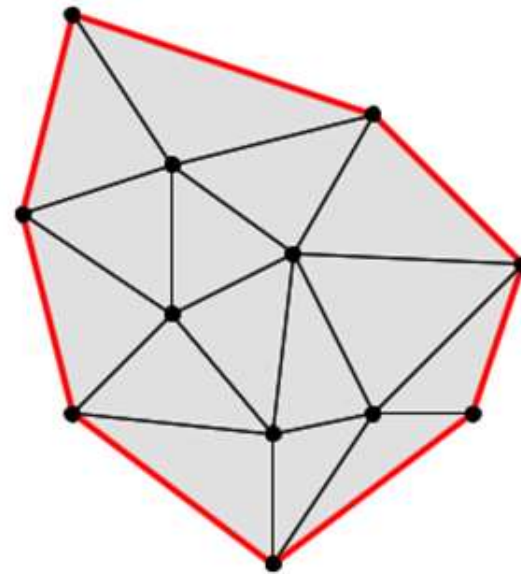
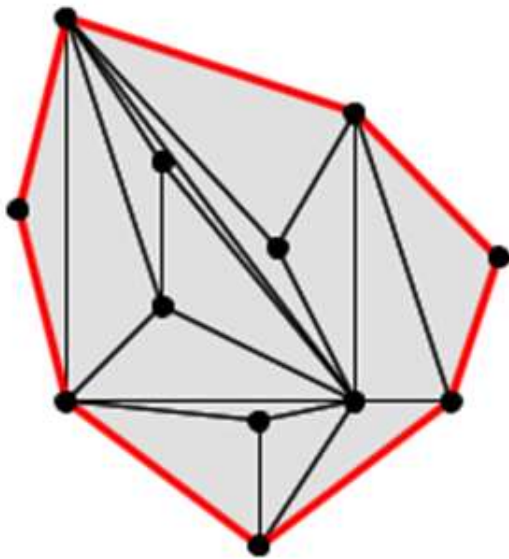
- Find an ear (always a triangle)
- Clip it
- Repeat until polygon is reduced to a triangle
- Runs in $O(n^2)$ time

Triangulation via Ear Clipping



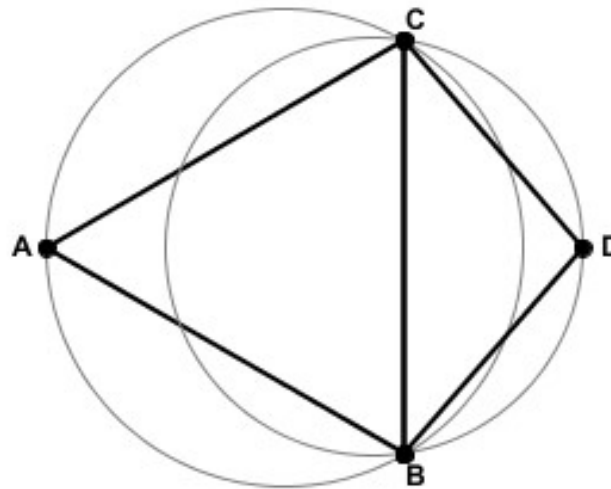
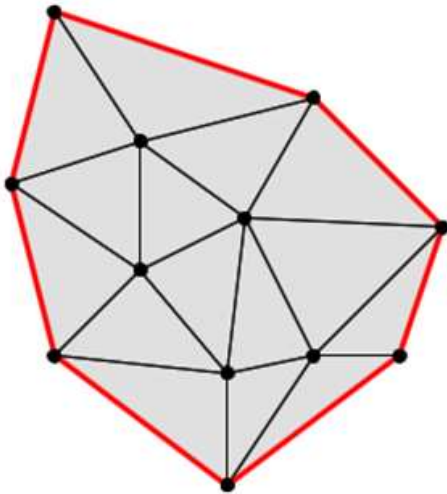
- Triangulation is not unique

Triangulation Quality



- Which one looks more “optimal”?

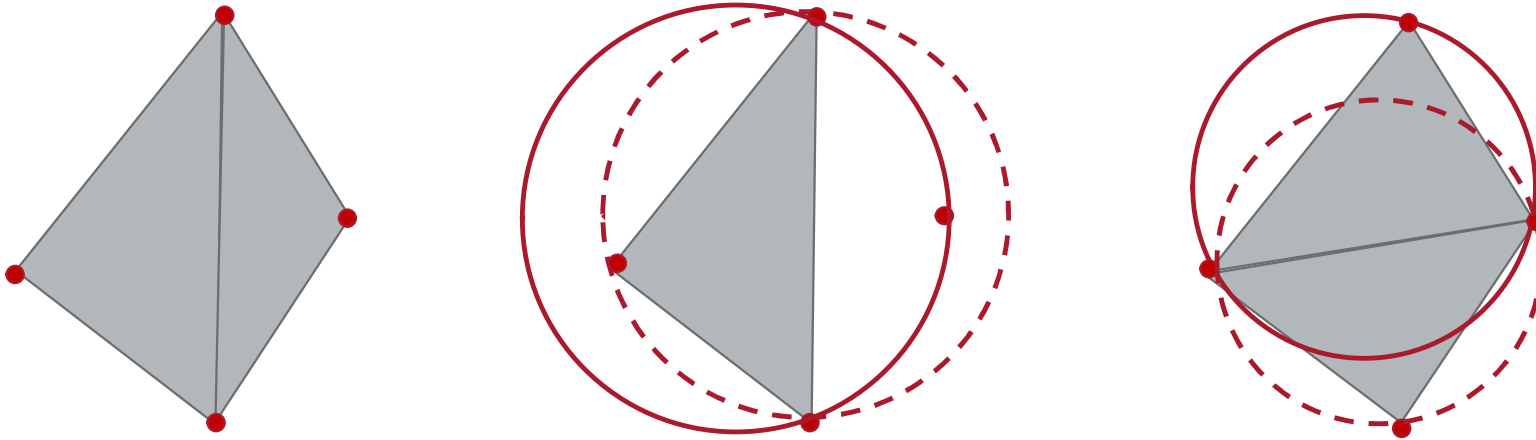
Delaunay Triangulation



**Boris Delaunay
(Delone)**

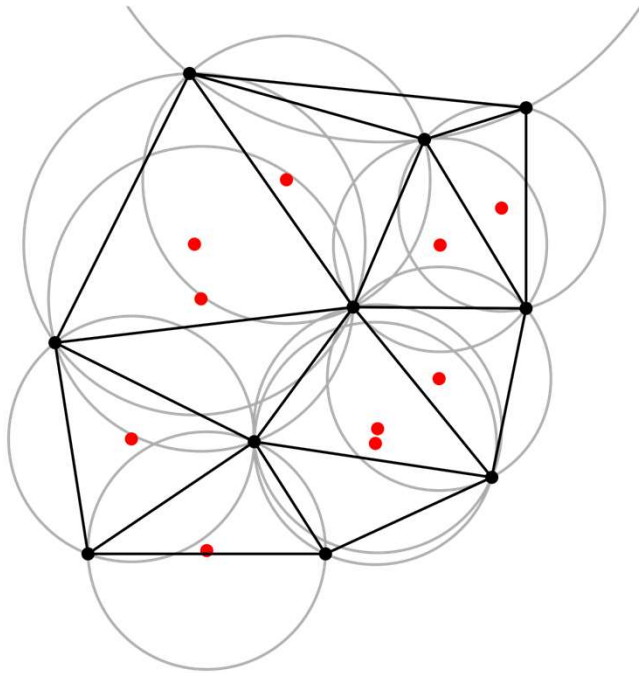
- No point is in the circumcircle of any other triangle
- Maximizes minimum angle of triangles (triangles tend to be closer to equilateral)
- Results in a “nice” partition

How to Compute Delaunay Triangulation

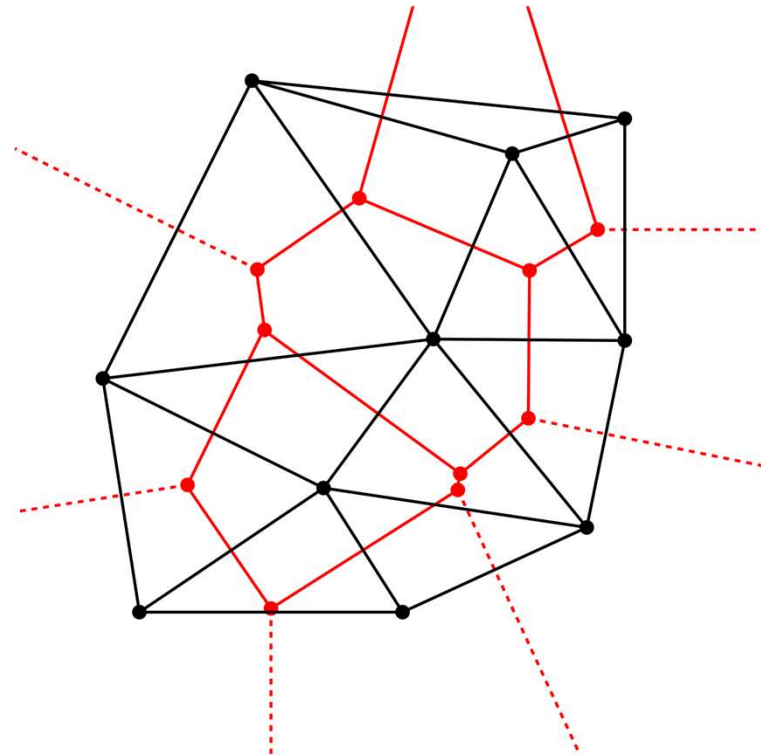


- Triangulate the set of points
- For each pair of triangles, check if they are locally Delaunay
 - If not, swap the common edge
- Repeat until convergence

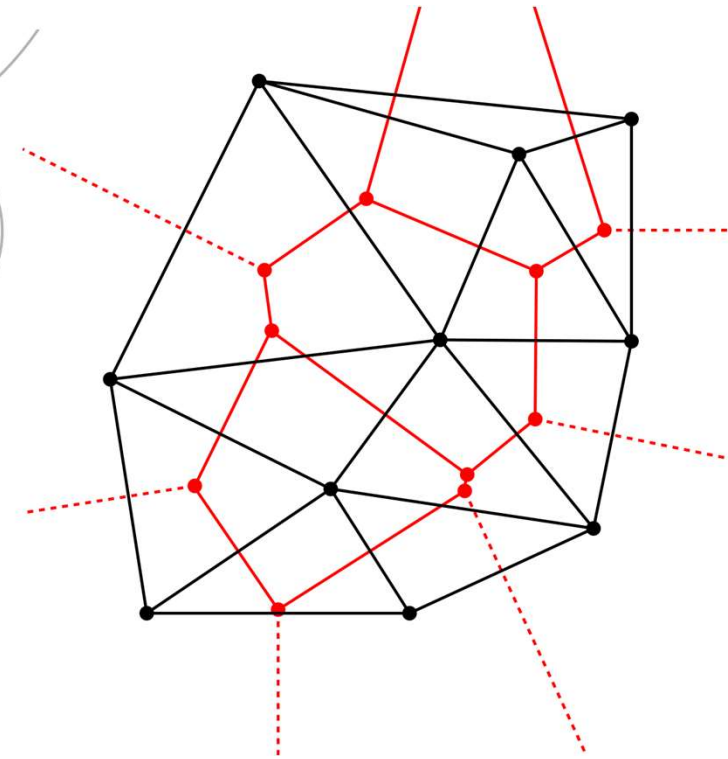
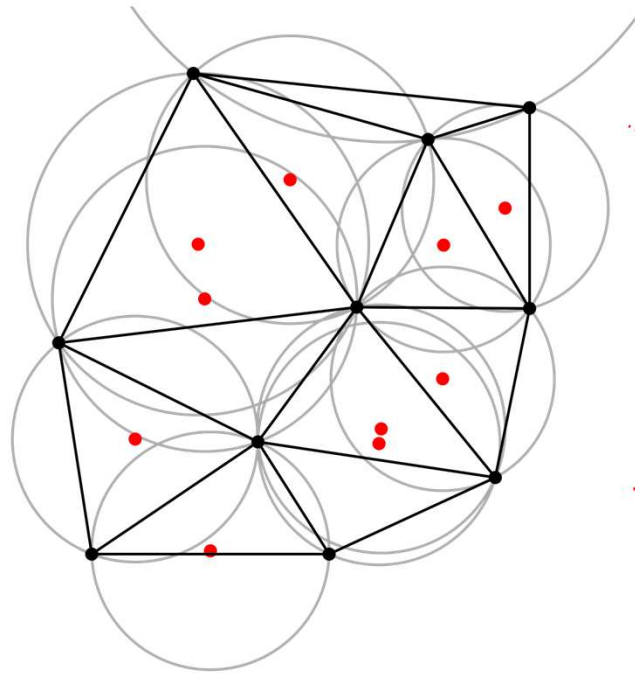
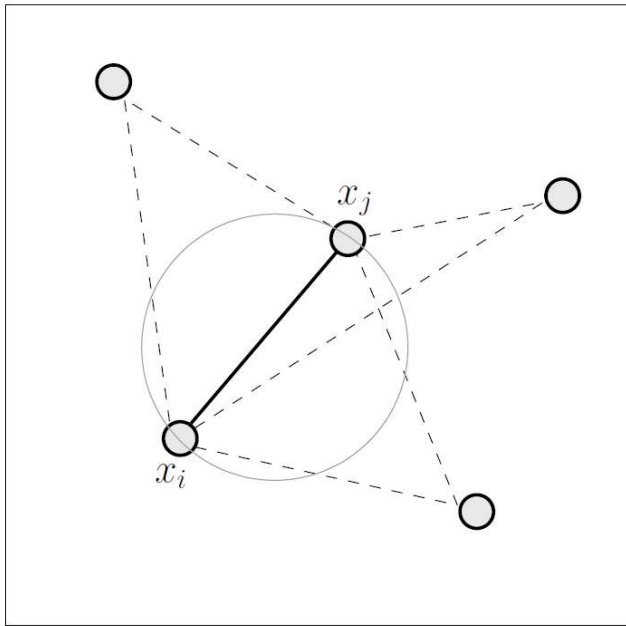
Delaunay Triangulation



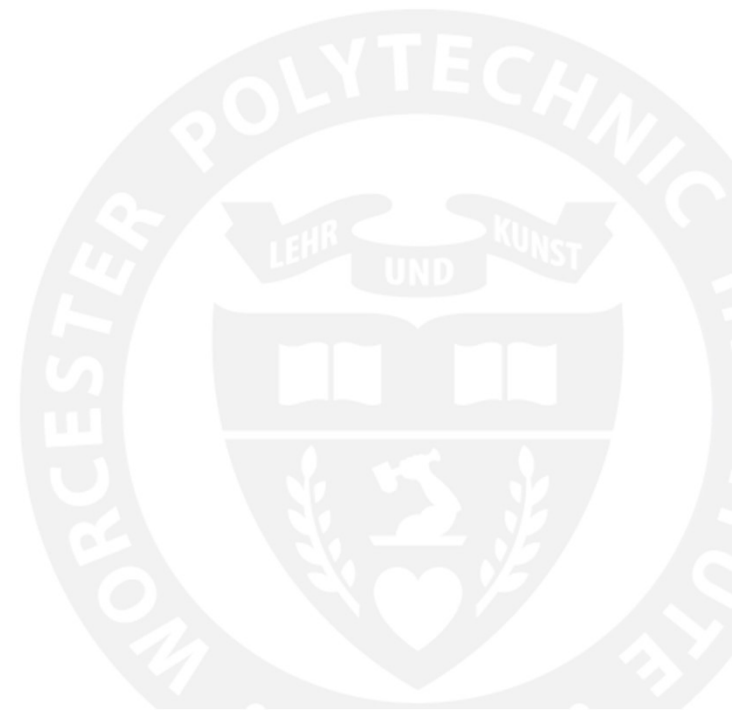
- Dual of Voronoi tessellation



Delaunay, Voronoi, and Gabriel



Wrap Up



Recap

- Coverage Control
 - Geometric notions
 - Gabriel graphs
 - Voronoi tessellations

Next Time

- Search
 - Single agent and multi-agent
 - Static and moving targets