A Projected Lloyd's Algorithm for Coverage Control Problems

Yoav Palti

Faculty of Aerospace Engineering, Technion - Israel Institute Of Technology

M.Sc. Seminar

Supervisor: Associate Professor Daniel Zelazo

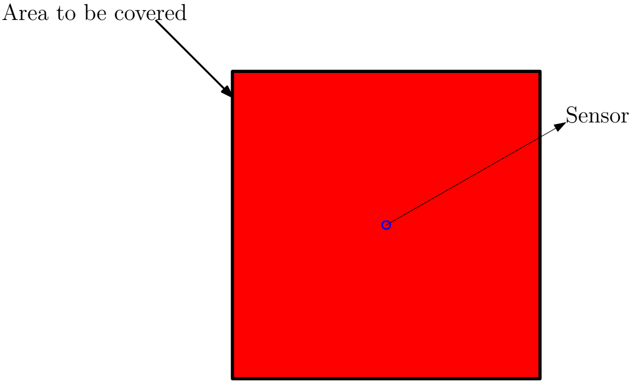
December 10, 2018

Table of Contents

Table of Contents

What Is Sensor Coverage?

Given an area, we want to sense what's happening inside



In red - the sensing coverage.

What Is Sensor Coverage?

Why would we like to do that?

- Surveillance[1]
- Photographing[1]
- Exploring[2]
- iRobot! [3]

Those are all sensing coverage problem[1,3].

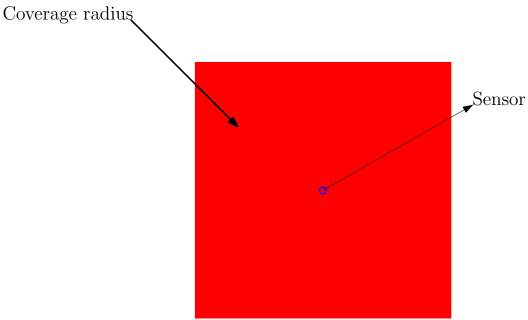
[1] Nigam, N., Bieniawski, S., Kroo, I., & Vian, J. (2012). Control of multiple UAVs for persistent surveillance: Algorithm and flight test results. IEEE Transactions on Control Systems Technology, 20(5), 1236–1251.

[2] Loizou, S. G., & Constantinou, C. C. (2016). Multi-Robot Coverage on Dendritic Topologies Under Communication Constraints, (Cdc).

[3] Montijano, E., Sagues, C., & Llorente, S. (2016). Multi-Robot Persistent Coverage with Optimal Times, (Cdc), 3511–3517.

What Is Sensor Coverage?

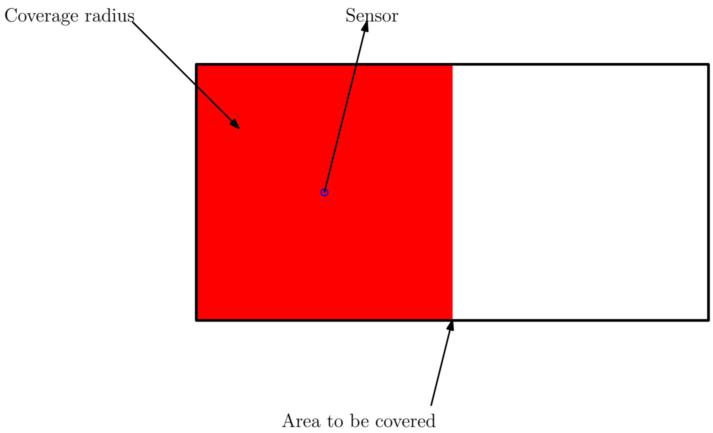
Coverage "Radius" - how much a sensor can sense



In red - the sensor coverage *radius*

Partial Coverage

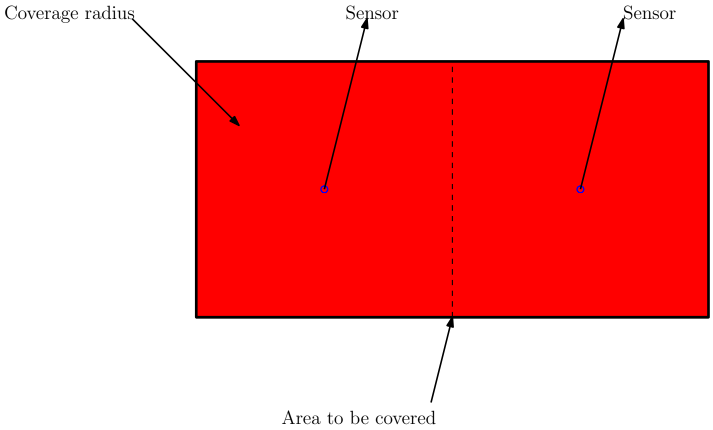
Single sensor - double the area size



In red - the sensor coverage. No *full coverage*!

Full Coverage (again)

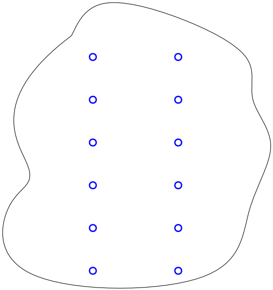
Let's add another sensor!



New sensor added - now we have *full coverage* once again!

Deployment

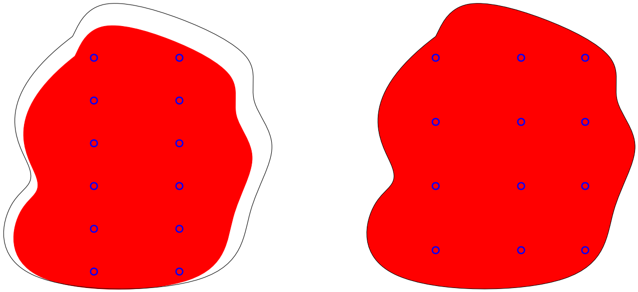
Now we're dealing with multiple sensors. How should we configure them?



We have 12 sensors which we can *deploy* in various configurations.

Deployment and full coverage

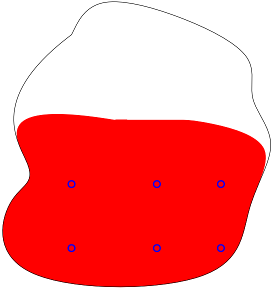
Is it that simple?

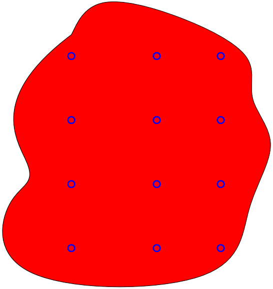


One configuration results with full coverage, while the other one doesn't.

Partial Coverage

There exists a deployment with 12 sensors which can cover the area. What if we only have 6 sensors?



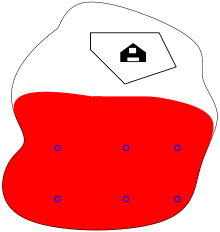


There doesn't exists a configuration that can supply full coverage!

Making it a bit more interesting...

Let's say that we want to maintain coverage on a specific area, due to:

- Connection to home base
- Maintain surveillance on a target
- ...



We have to take this into account when we build our coverage *strategy*.

Partial Coverage Strategy

Dealing with partial coverage - many possible behaviours:

- Set of trajectories [1,2]
- *Tiling the area*[3]
- ...

By choosing any strategy, a *coverage controller*[3] must be provided.

[1] Atinç, G. M., Stipanović, D. M., Voulgaris, P. G., & Karkoub, M. (2013). Supervised coverage control with guaranteed collision avoidance and proximity maintenance. Proceedings of the IEEE Conference on Decision and Control, 3463–3468.

[2] Hussein, I. I., & Stipanovic, D. M. (2007). Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance. IEEE Transactions on Control Systems Technology, 15(4), 642–657.

[3] Cortes, J., & Martinez, S. (2004). Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation, 20(2), 243-255.

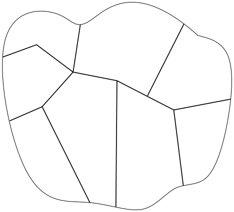
[4] Cassandras, C. G., & Li, W. (2005). Sensor Networks and Cooperative Control. Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference On, 4237–4238.

Partitioning as a strategy

Partitioning (or tiling) an area - cover a small part of the area each time.

- Main benefit - provide coverage of a subset of an area constantly.

Cortes et al. provided a controller that know how to partition an area and provide coverage, using Centroidal Voronoi Tessellations[1].



[1] Du, Q., Faber, V., & Gunzburger, M., "Centroidal Voronoi Tessellations: Applications and Algorithms," SIAM Review, Vol. 41, No. 4, 1999, pp. 637–676.

Problem Formulation

- There is some area $A \in \mathbb{R}^2$ That we aim to cover.
- A sub-area $A_m \subset A$ must be covered always (e.g. ground station).
- There exist a set of **mobile** sensors $S = \{s_1 \ldots s_n\}$ located in positions $p_i(t) \in \mathbb{R}^2$ (for $i = 1, \ldots, n$) at time t.
    - The sensors can be controller with integrator dynamics $\dot{p_i}(t) = u$.
    - Each sensor can cover an area defined as $Cover\,(p_i, t)$.

Problem Formulation

- A configuration c at time t is the stack of the sensor positions at time t,

$$c\left(t\right) = \begin{bmatrix} p_1^T\left(t\right) & \cdots & p_n^T\left(t\right) \end{bmatrix}^T \in \mathbb{R}^{2n}$$

  - The coverage of a configuration $D\left(c\left(t\right)\right) = \cap Cover\left(p_i, t\right)$.

Assumption

$D\left(c\left(t\right)\right) \subset A$ - a single configuration *can't* provide full coverage!

Problem Formulation

- A partition j of the area A is $pr_j \subset A$
- The partitioning of $A, PR(A)$, built from n partitions:

$$PR(A) = \{pr_j \mid \forall i \neq j,\ pr_i \cap pr_j = \emptyset \textbf{ and } \cup pr_j = A\}$$

Problem Formulation

Problem

1. *Find partitioning such that for each partition* $j, pr_j \cap A_m \neq \emptyset$
2. *Find a deployment controller such that for each partition* j*, and some given time* t*,* $A_{pr_j} \subseteq D\left(c\left(t\right)\right)$*, assuming that the controller is in its steady state.*

Table of Contents

Voronoi Partitioning

A little story about a town, a city planner and post offices...

Voronoi Partitioning

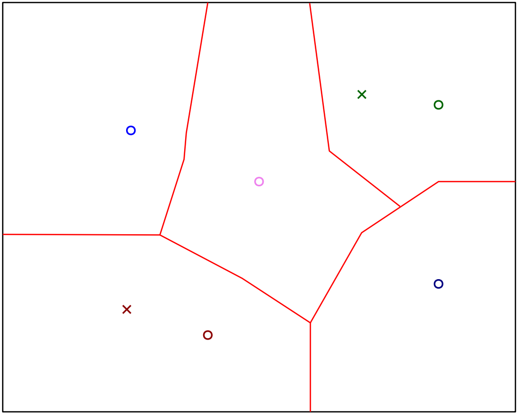
The Voronoi Diagram of a region $\Omega \subset \mathbb{R}^2$ is the set of partitions $\mathcal{V} = \{V_i \mid \cup V_i = \Omega\}$, generated by the generators $\mathcal{Z} = \{z_1, \ldots, z_n \mid z_i \in \Omega\}$, such that

$$V_i = \{q \in \Omega \mid \|q - z_i\| \leq \|q - z_j\| \forall z_i, z_j \in \mathcal{Z}\},$$

where V_i corresponds to the i-th element of $\mathcal{Z}$, and $\|\cdot\|$ denotes the Euclidean distance.

Voronoi Partitioning

A rough example:



Circles - generators, crosses - some point inside the appropriate partition.

Central Voronoi Tessellations

Let's get back to our city planner.

Central Voronoi Tessellations

Let us define a density function, ρ_i, for each Voronoi partition V_i. Then, we can define the center of mass for each partition as

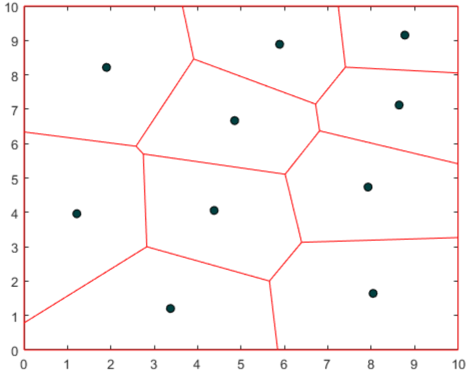
$$z_i^* = \frac{\int_{V_i} y\rho(y)dy}{\int_{V_i} \rho(y)dy}.$$

If a generator $z_i = z_i^* \,\forall\, V_i$, we call this partitioning a *centroidal Voronoi tessellation* (CVT). Common examples for density function:

- $\rho(y) = \mathcal{N}\left(\mu, \sigma^2\right)$ (Gaussian distribution)
- $\rho(y) = 1$

Central Voronoi Tessellations

How it looks like?



Lloyd's Algorithm

How do we calculate CVT?

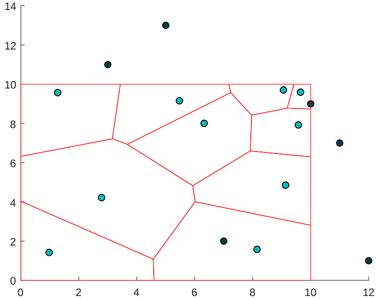
Algorithm 1 Lloyd's Algorithm

1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Move the agents to the center of mass.
4: Repeat until convergence.

Lloyd's Algorithm

So how do we calculate it?

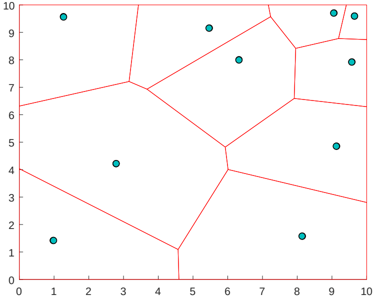
Step 1:



Black - initial guess, turquoise - after first iteration

Lloyd's Algorithm

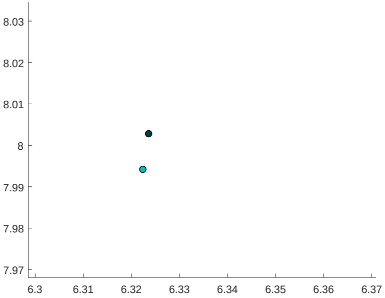
Step 2:



Black - first iteration solution, turquoise - after second iteration

Lloyd's Algorithm

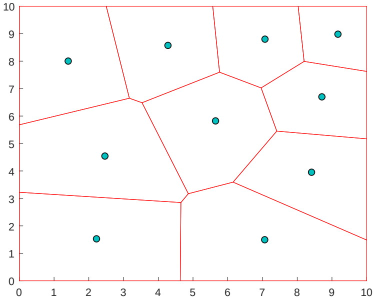
Step 2 - zoom in:



Almost converged...

Lloyd's Algorithm

After n iterations:



Lloyd's Algorithm

[Cortes2004] proposed a continuous time controller for Lloyd's algorithm. If we define agent i position as p_i and the i's partition centroid as C_{V_i}, then for some proportional constant k_p, the controller can be defined as:

$$u_i = -k_p \left(p_i - C_{V_i}\right)$$

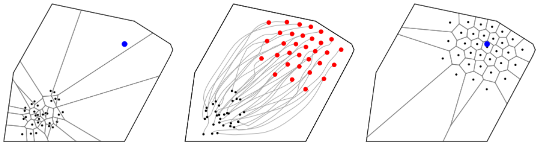


Figure: A simulation from [Cortes2004] with 32 agents and Gaussian density function

Lloyd's Algorithm

Another important result in [Cortes2004] is that this controller is locally asymptotically stable. Proof is given in the paper using the direct Lypunov methos, using the following potential function:

$$\mathcal{H}_{\mathcal{V}}\left(P\right)=\sum_{i=1}^{n}J_{V_i,C_{V_i}}+\sum_{i=1}^{n}M_{V_i}\|p_i-C_{V_i}\|^2$$

Where:

- P - the set of the agents positions
- V_i - the i'th Voronoi partition
- $J_{V_i,C_{V_i}}$ - the polar moment of inertia of V_i about its centroid C_{V_i}
- M_{V_i} - the i'th partition "mass"

Lloyd's Algorithm

- This form of Lloyd's algorithm is centralized!
- The calculation of the Voronoi partitions converges into a local minima

An important thing we have to remember -

Table of Contents

Problem reminder

So what were we trying to do (in simple words)?

Reminder

1. Partition the area A, such that any partition will intersect with some sub-area A_m.
2. For each partition, find some deployment strategy.

Problem reminder

So what were we trying to do (in simple words)?

Reminder

1. Partition the area A, such that any partition will intersect with some sub-area A_m.
2. For each partition, find some deployment strategy.

[Cortes2004] came up with a solution for the second issue. But what about the first one?

Projection

A possible solution - After calculating the center of mass of each cell, *project* the results into A_m boundaries.

Projection

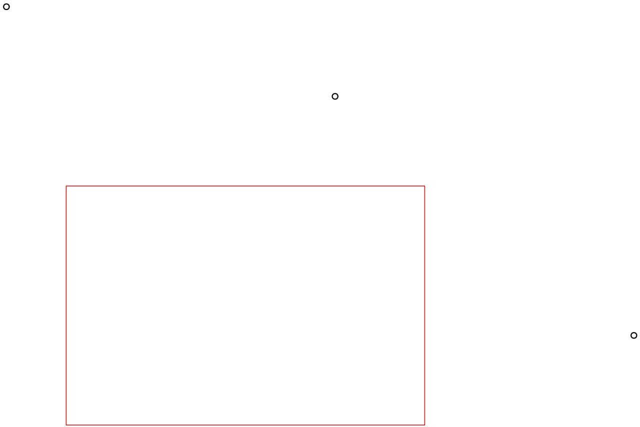
A possible solution - After calculating the center of mass of each cell, *project* the results into A_m boundaries.

Projection

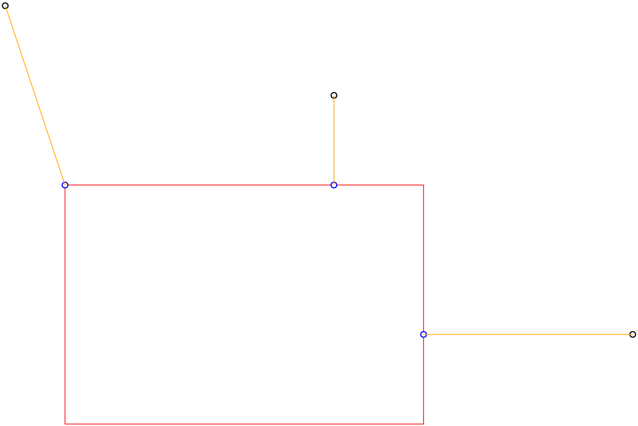
A linear transformation P from a vector space to itself such as $P^2 = P$. In other words, the transformation P is idempotent.

We will project using Euclidean distance.

Projection example



Projection example



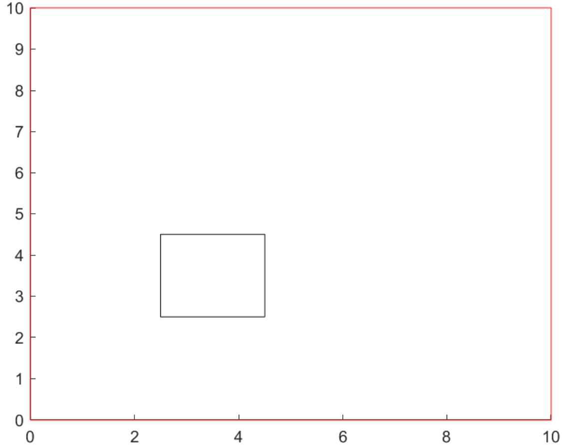
Projected Lloyd's Algorithm

Algorithm 2 Projected Lloyd's Algorithm (PLA)

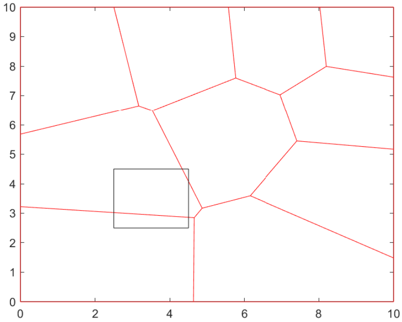
1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Project the center of mass of every cell to the area constraint limiting polygon.
4: Move the agents the projected center of mass.
5: Repeat until converge.

Projected Lloyd's Algorithm Example

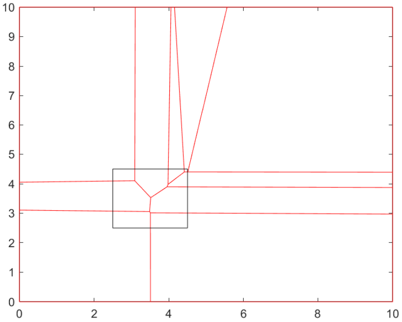
Assume that we have an area A and some sub-area A_m:



Projected Lloyd's Algorithm



CVT using Lloyd's Algorithm



Partitioning using PLA

Projected Lloyd's Algorithm

PLA in continuous time controller form:

$$u_i = -k_p \left(p_i - \mathit{proj}\left(C_{V_i}\right)\right) \tag{1}$$

Theorem

The projected Lloyd's Algorithm is locally asymptotically stable

Proof of stability is using the direct Lyapunov method, with a potential function being very similar to the one proposed in [Cortes2004]:

$$\mathcal{H}_{\mathcal{V}}\left(P\right) = \sum_{i=1}^{n} J_{V_i,C_{V_i}} + \sum_{i=1}^{n} M_{V_i} \left\| p_i - \text{PROJ}\left(C_{V_i}\right)\right\|^2$$

Problem Solution Algorithm

So far:

- Covering a given area using Voronoi partitioning - **Solved** ([Cortes2004]).
- Partition and area such that the coverage constraint is fulfilled - **Solved** (PLA).

Therefore, we are ready for the problem solution algorithm...

Problem Solution Algorithm

Algorithm 3 Problem Solution Algorithm

1: Using some random initial guess, partition the whole area using PLA.

2: For each partition (assuming that the agents can actually cover each partition with their coverage radius), calculate the CVT. The initial positions for the CVT calculation is the previous partition CVT.

Some Simulation

List of simulations to create:

- 3 agents, 5 big partitions, no formation, no PLA
- 3 agents, 5 big partitions, no formation, PLA
- 10 agents, 5 big partitions, no formation, PLA

One more thing...

- Problem solved.

One more thing...

- Problem solved.
- What if maintaining spatial properties is also needed?
    - Geolocation
    - communications
    - ...

One more thing...

- Problem solved.
- What if maintaining spatial properties is also needed?
  - Geolocation
  - communications
  - ...
- Incorporate formation control into existing algorithms!

Formation Control

The concept of distance-based formation control is well researched[1,2]. For complete background, some prior knowledge on algebraic graph theory is needed. Let's simplify:

- We have agents $1 \dots n$ on positions p_i

---

[1] aura Krick, Mireille E. Broucke & Bruce A. Francis (2009) Stabilisation of infinitesimally rigid formations of multi-robot networks, International Journal of Control, 82:3, 423-439.

[2] K. Oh and H. Ahn, "Distance-based formation control using euclidean distance dynamics matrix: Three-agent case," Proceedings of the 2011 American Control Conference, San Francisco, CA, 2011, pp. 4810-4815.

Formation Control

The concept of distance-based formation control is well researched[1,2]. For complete background, some prior knowledge on algebraic graph theory is needed. Let's simplify:

- We have agents $1 \ldots n$ on positions p_i
- Goal - agents $i, j (i \neq j)$ will be at distance d_{ij}

[1] aura Krick, Mireille E. Broucke & Bruce A. Francis (2009) Stabilisation of infinitesimally rigid formations of multi-robot networks, International Journal of Control, 82:3, 423-439.

[2] K. Oh and H. Ahn, "Distance-based formation control using euclidean distance dynamics matrix: Three-agent case," Proceedings of the 2011 American Control Conference, San Francisco, CA, 2011, pp. 4810-4815.

Formation Control

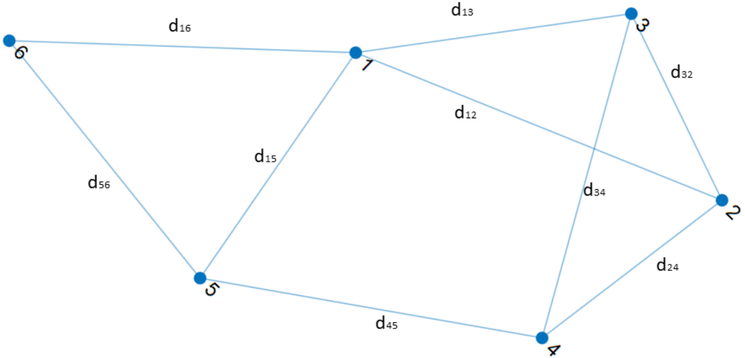
The concept of distance-based formation control is well researched[1,2]. For complete background, some prior knowledge on algebraic graph theory is needed. Let's simplify:

- We have agents $1 \dots n$ on positions p_i
- Goal - agents $i, j (i \neq j)$ will be at distance d_{ij}
- Only ε agents can share information.

[1] aura Krick, Mireille E. Broucke & Bruce A. Francis (2009) Stabilisation of infinitesimally rigid formations of multi-robot networks, International Journal of Control, 82:3, 423-439.

[2] K. Oh and H. Ahn, "Distance-based formation control using euclidean distance dynamics matrix: Three-agent case," Proceedings of the 2011 American Control Conference, San Francisco, CA, 2011, pp. 4810-4815.

Formation Control



Formation Control

For a single agent p_i, the controller will have the following form:

$$\dot{p}_i = -\sum_{i \sim j} \left(\|p_i - p_j\|^2 - d_{ij}^2 \right) (p_i - p_j) \tag{2}$$

This controller is locally asymptotically stable.

Lloyd's Algorithm and Formation Control

As both of the controllers are convex, we propose to simply combine them with some coefficient:

$$u_i = \alpha\left(-k_p\left(p_i - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(\|p_i - p_j\|^2 - d_{ij}^2\right)\left(p_i - p_j\right)\right] \tag{3}$$

Lloyd's Algorithm and Formation Control

As both of the controllers are convex, we propose to simply combine them with some coefficient:

$$u_i = \alpha\left(-k_p\left(p_i - C_{V_i}\right)\right) + (1-\alpha)\left[-\sum_{i\sim j}\left(\|p_i - p_j\|^2 - d_{ij}^2\right)\left(p_i - p_j\right)\right] \tag{3}$$

Theorem

The combined controller is Locally Asymptotically Stable

Lloyd's Algorithm and Formation Control

How to prove:

- Pretty long and technical, based on Lyapunov function.
- We know the Lyapunov function of each controller - Let's combine!
- After long calculations, we can show using the Lyapunov direct method that this controller is locally asymptotically stable.
- In the same way, we can show it works with the PLA.