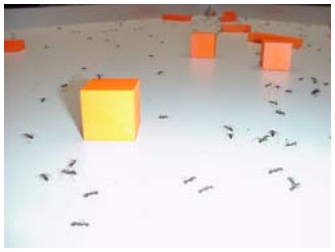
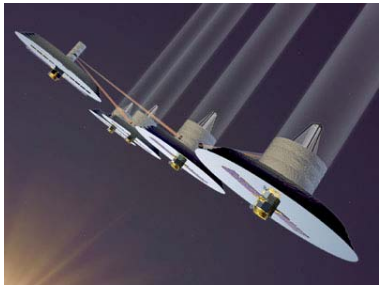
Graph-Based Control of Heterogeneous Robot Networks

Magnus Egerstedt

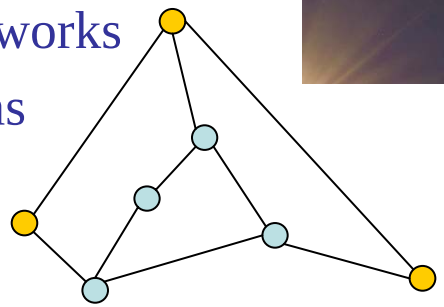
GRITS LAB
Electrical and Computer Engineering
Georgia Institute of Technology
http://www.ece.gatech.edu/~magnus



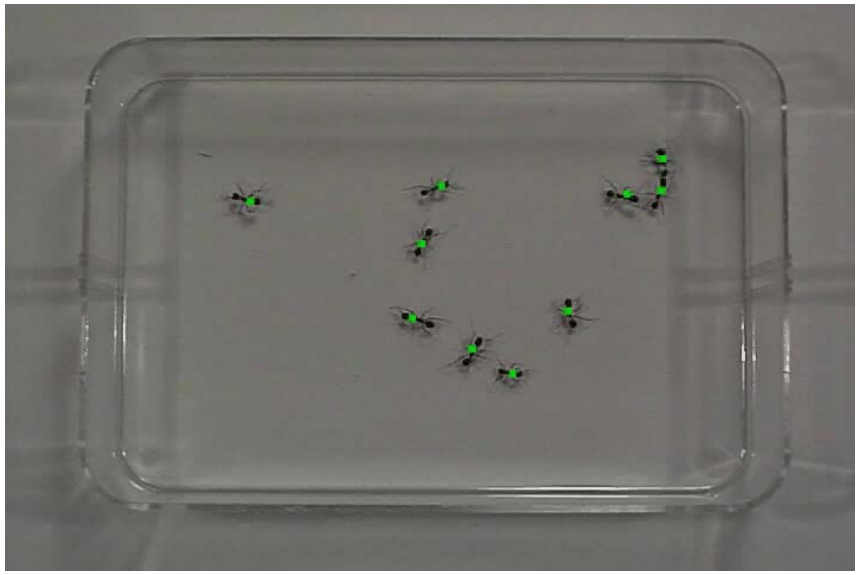


Outline:

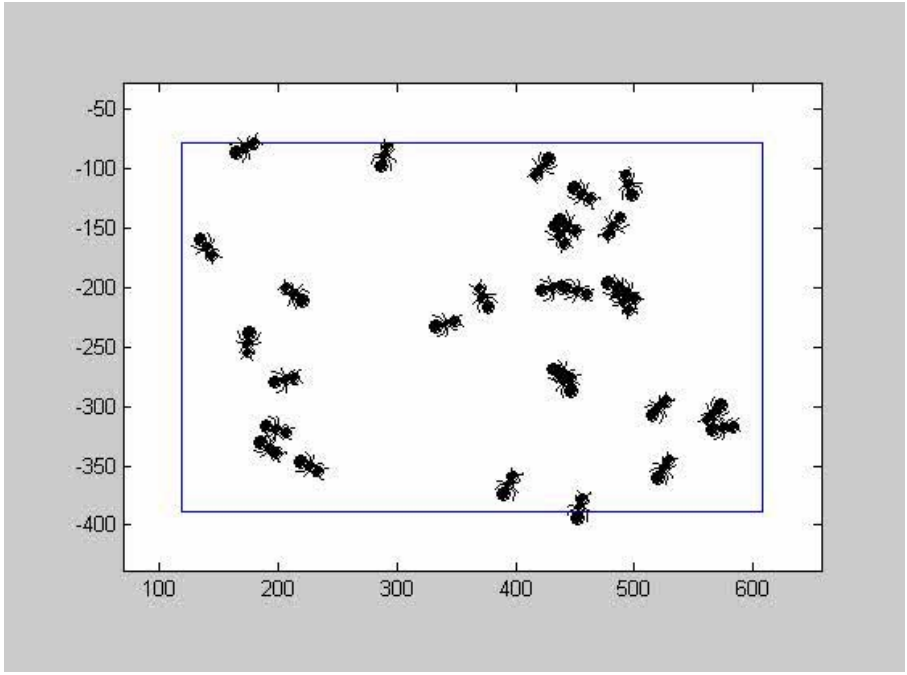
- Graph-Based Control
- Anchor Nodes and Leader-Networks
- Control of Multi-Agent Systems



Why I Started Caring About Multi-Agent Systems



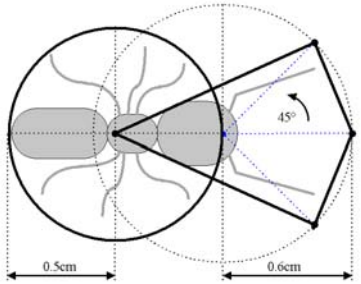
?

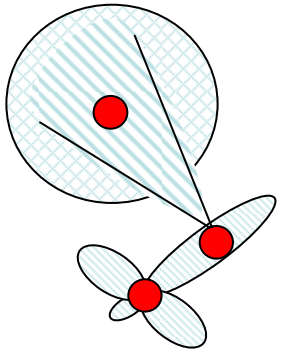


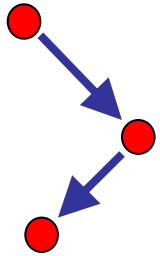
"They look like ants."
– Stephen Pratt, Dept. of Ecology and Evolutionary Biology, Princeton University

Graphs as Network Abstractions

- A networked sensing and actuation system consists of
  - **NODES** - physical entities with limited resources (computation, communication, perception, control)
  - **EDGES** - virtual entities that encode the flow of information between the nodes







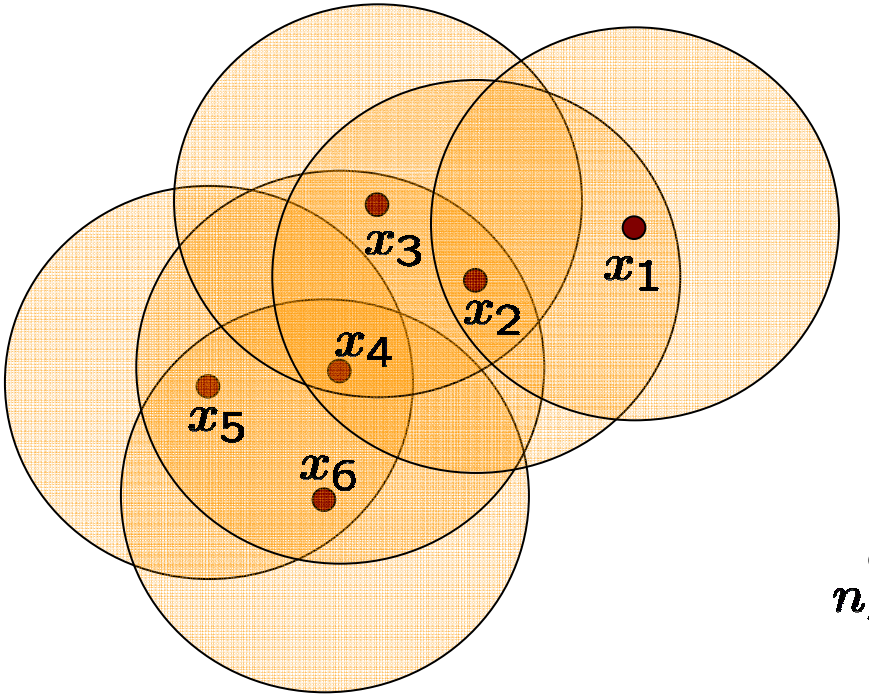
- The "right" mathematical object for characterizing such systems at the network-level is a **GRAPH**
  - Purely ***combinatorial*** object (no ***geometry*** or ***dynamics***)
  - The characteristics of the information flow is abstracted away through the (possibly weighted and directed) edges

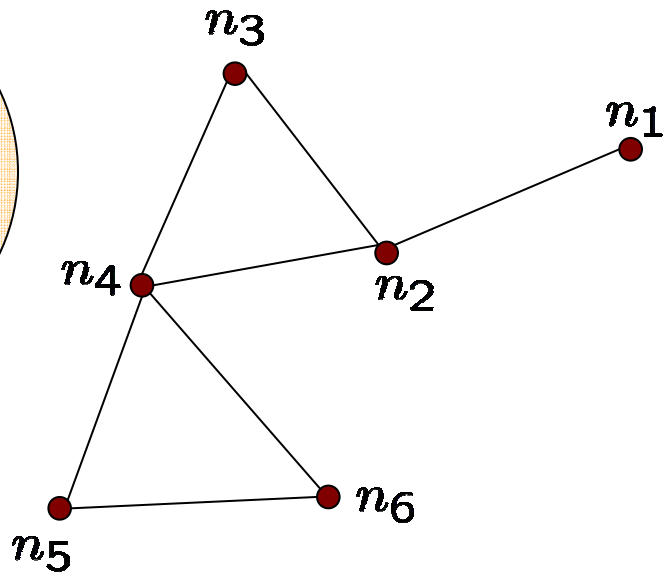
Graphs as Network Abstractions

- The connection between the combinatorial graphs and the geometry of the system can for instance be made through geometrically defined edges.
- Examples of such proximity graphs include **disk-graphs**, **Delaunay graphs, Voronoi graphs**, and **Gabriel graphs**.

- Example: **Disk-graphs** $(G = (\mathcal{N}, \mathcal{E}))$
  - $\delta > 0$: effective sensing/communications range
  - $\mathcal{N} = \{1, \ldots, N\}$: set of nodes corresponding to agents located at $x_i \in \Re^d, \ i = 1, \ldots, N$
  - $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$: set of edges with $(n_i, n_j) \in \mathcal{E} \Leftrightarrow \|x_i - x_j\| < \delta$
- Note: These graphs are undirected, i.e. the edge set contains unordered rather than ordered pairs of nodes

Example: Disk-Graphs





$$\mathcal{N} = \{n_1, n_2, n_3, n_4, n_5, n_6\}$$
$$\mathcal{E} = \{(n_1, n_2), (n_2, n_3), (n_3, n_4), (n_2, n_4), (n_4, n_5), (n_4, n_6), (n_5, n_6)\}$$

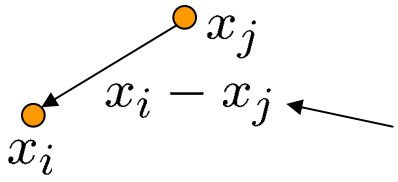
GRAPH-BASED CONTROL

LEADER NETWORKS

CONTROL ISSUES

Rendezvous – A Canonical Problem

- Given a collection of mobile agents who can only measure the relative displacement of their neighbors (no global coordinates)



This is what agent ***i*** can measure

- Problem: Have all the agents meet at the same (unspecified) position

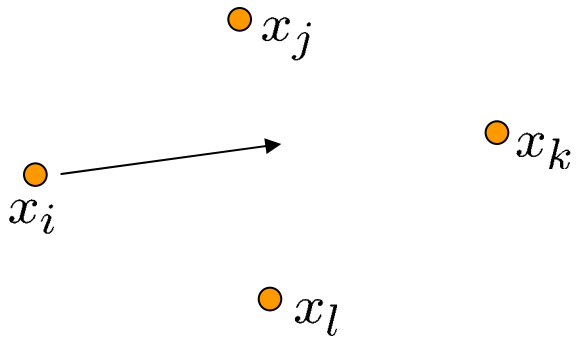
- If there are only two agents, it makes sense to have them drive towards each other, i.e.

$$\dot{x}_1 = -\gamma_1(x_1 - x_2)$$
$$\dot{x}_2 = -\gamma_2(x_2 - x_1)$$

- If $\gamma_1 = \gamma_2$ they should meet halfway

Rendezvous – A Canonical Problem

- If there are more then two agents, they should probably aim towards the centroid of their neighbors (or something similar)



$$\dot{x}_i = -\gamma \sum_{j \in \mathcal{N}_i} (x_i - x_j)$$

- **Does this work? Does the network topology matter? Where, if anywhere, is the rendezvous point?**
- To answer these questions, we need to tap into the wonderful world of algebraic graph theory

Algebraic Graph Theory

- Algebraic graph theory provides a bridge between the combinatorial graph objects and their matrix representations
  - **Degree matrix**:

$$D = \text{diag}(\deg(n_1), \ldots, \deg(n_N))$$

  - **Adjacency matrix**:

$$A = [a_{ij}], \; a_{ij} = \begin{cases} 1 & \text{if } n_i \text{ — } n_j \\ 0 & \text{o.w.} \end{cases}$$

  - **Incidence matrix** (directed graphs):

$$\mathcal{I} = [\iota_{ij}], \; \iota_{ij} = \begin{cases} 1 & \text{if } e_j \rightarrow n_i \\ -1 & \text{if } n_i \rightarrow e_j \\ 0 & \text{o.w.} \end{cases}$$

  - **Graph Laplacian**:

$$\mathcal{L} = D - A = \mathcal{I}\mathcal{I}^T$$

The Consensus Equation

- The reason why the graph Laplacian has become so fashionable is through the already seen "consensus equation"

$$\dot{x}_i = -\sum_{j \in \mathcal{N}_i} (x_i - x_j), \;\; i = 1, \ldots, N$$

or equivalently (W.L.O.G. scalar agents)

$$\left.\begin{array}{l} \dot{x}_i = -\mathsf{deg}(n_i) x_i + \sum_{j=1}^{N} a_{ij} x_j \\ x = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \end{array}\right\} \;\Rightarrow\; \dot{x} = -\mathcal{L} x$$

- This is an autonomous LTI system whose convergence properties depend purely on the spectral properties of the Laplacian.

- Can these be understood directly from the network topology (without having to compute any Laplacians)?

Graph Laplacians: Useful Properties

- It is orientation independent
- It is symmetric and positive semi-definite
- If the graph is ***connected*** then

$$\text{eig}(\mathcal{L}) = \{\lambda_1, \ldots, \lambda_N\}, \text{ with } 0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$$

$$\text{eigv}(\mathcal{L}) = \{\nu_1, \ldots, \nu_N\}, \text{ with } null(\mathcal{L}) = \text{span}\{\nu_1\} = \text{span}\{1\}$$

- Thus, if the graph is connected

$$\lim_{t \to \infty} x(t) \in null(\mathcal{L}) = \text{span}\{1\}$$

with an exponential convergence rate given by λ_2 ← Fiedler number

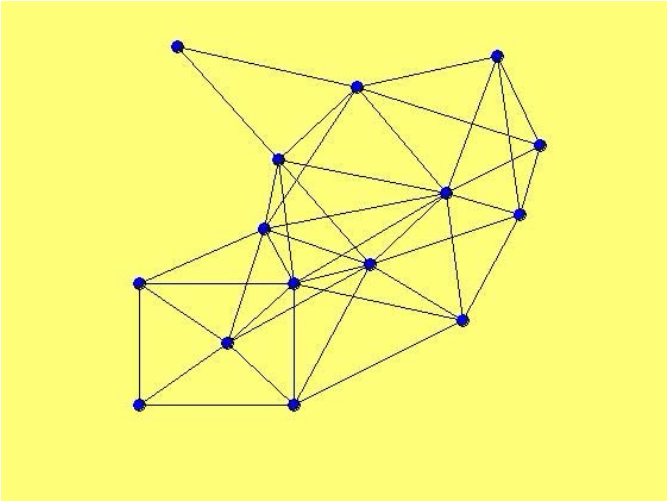
algebraic connectivity

- **Hence the rendezvous problem is solved!!**

Example: Rendezvous

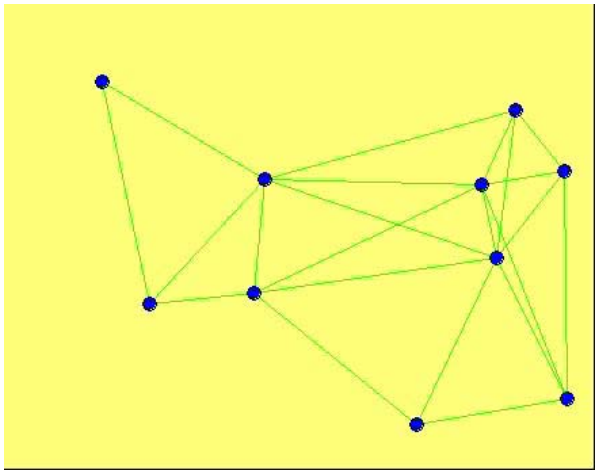
Fact: As long as the graph stays connected, the consensus equation drives all agents to the same state value

$$\lim_{t\to\infty} x_i(t) = \bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j(0)$$

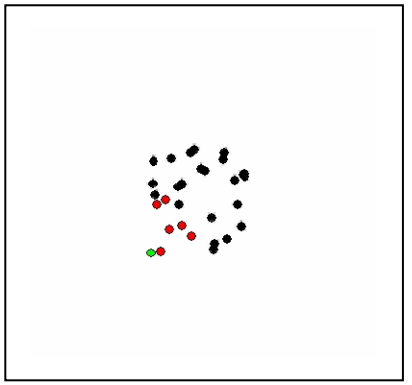


Graph-Based Control

- In fact, based on variations of the consensus equation, a number of different multi-agent problems have been "solved", e.g.
  - **Formation control** (How drive the collection to a predetermined configuration?)
  - **Coverage control** (How produce triangulations or other regular structures?)



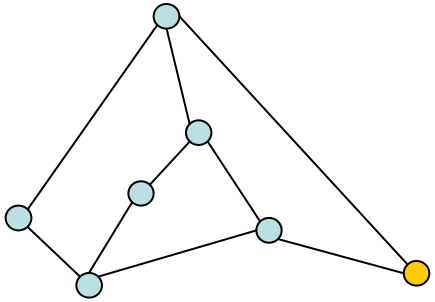
- ***OK – fine. Now what?***
- Need to be able to **reprogram and redeploy** multi-agent systems (**HSI = Human-Swarm Interactions**)
- This can be achieved through active control of so-called leader-nodes

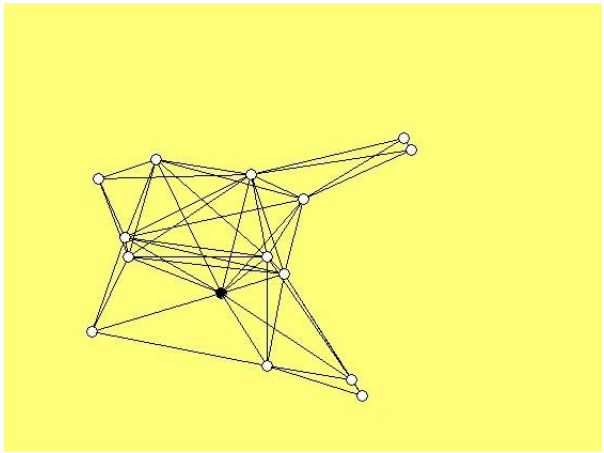


GRAPH-BASED CONTROL
LEADER NETWORKS
CONTROL ISSUES

Leader (Anchor) Nodes

- **Key idea**: Let some subset of the agents act as control inputs and let the rest use the consensus equation (nearest neighbor rule)





Controllability of Networked Systems

- Given a collection of leaders and followers, with

$x \in \Re^{N_f}$ ← followers' positions

$u \in \Re^{N_l}$ ← leaders' positions

$$\dot{x} = -\mathcal{L}_f x - \ell u$$

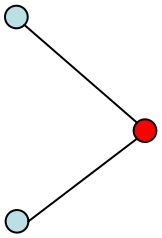
$$\mathcal{L} = \left(\begin{array}{c|c} \mathcal{L}_f & \ell \\ \hline \ell^T & \lambda \end{array}\right)$$ ← static graph Laplacian

- When is the system completely controllable? Well, obviously when

$$\text{rank}\left(\left[\ell \;\; \mathcal{L}_f\ell \;\; \cdots \;\; \mathcal{L}_f^{N_f-1}\ell\right]\right) = N_f$$

- But, from a design point-of-view, it would be useful if the interaction topology could be constructed to ensure **controllability directly based on graph-theoretic properties**.

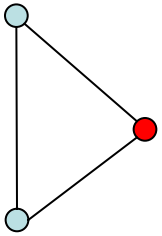
Some Examples



Not controllable!

Why?

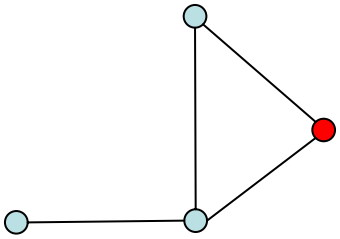
$$\dot{x}_1 = -(x_1 - u),\ \dot{x}_2 = -(x_2 - u)$$
$$x_1(0) = x_2(0) \ \Rightarrow\ x_1(t) = x_2(t)\ \forall u, t \geq 0$$



Not controllable!

Why? - Same reason!

$$\dot{x}_1 = -(x_1 - u) - (x_1 - x_2)$$
$$\dot{x}_2 = -(x_2 - u) - (x_2 - x_1)$$
$$x_1(0) = x_2(0) \ \Rightarrow\ x_1(t) = x_2(t)\ \forall u, t \geq 0$$

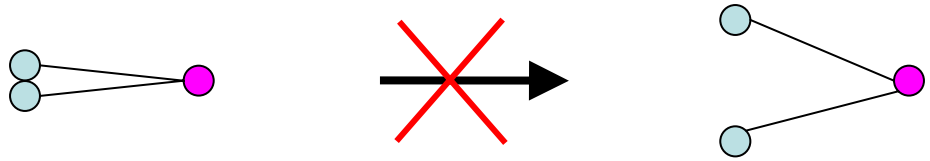


Controllable!

Why? - Somehow it seems like some kind of "symmetry" has been broken.

Graph Symmetries

- If the graph is somehow symmetric with respect to the (single) leader, then there are initial configurations from which not all configurations can be reached.



the followers start from the same position

there is no way they will separate, no matter the movement of the leader

- To make this formal, we need to introduce **graph automorphisms** (more algebraic graph theory)

Graph Automorphisms

- **Def:** An ***automorphism*** of a graph is a permutation of its node set such that

$$(\psi(n_i), \psi(n_j)) \in \mathcal{E} \;\Leftrightarrow\; (n_i, n_j) \in \mathcal{E}$$

- **Def:** The single-leader system is ***leader symmetric*** if there exists a non-identity automorphism that leaves the leaders agent invariant.

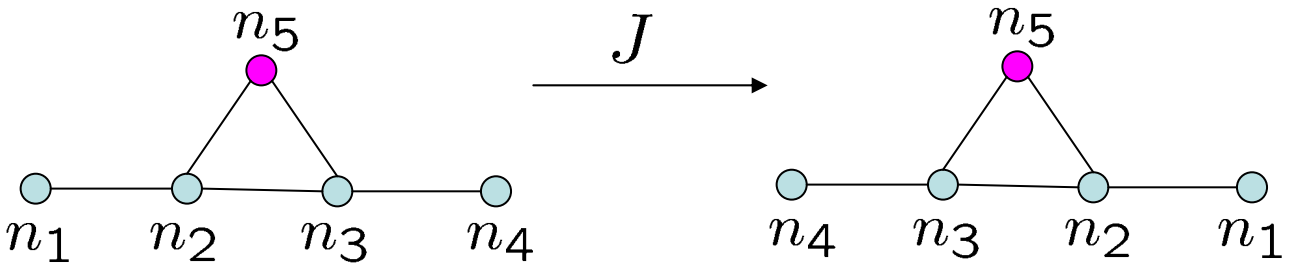
- **Thm**: The single-leader system

$$\dot{x} = -\mathcal{L}_f x - \ell u$$

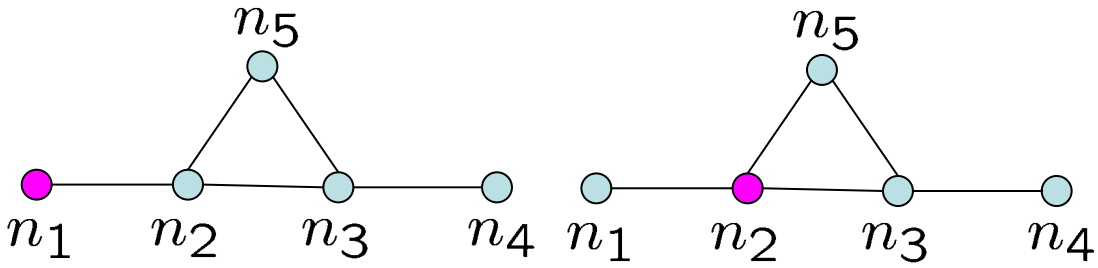
is uncontrollable if it is leader-symmetric

Leader Symmetries

- Intuitively, what this means is that a sufficient condition for the system to not be controllable is that one can relabel the followers without changing the label of the leader, and get an automorphism.



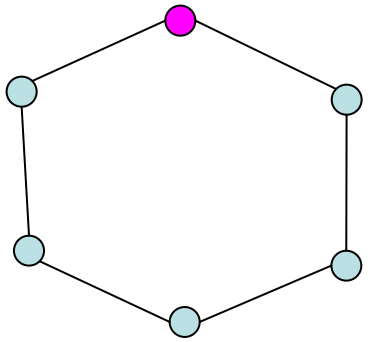
Not controllable!



Not leader-symmetric!
(In fact, these graphs are controllable)

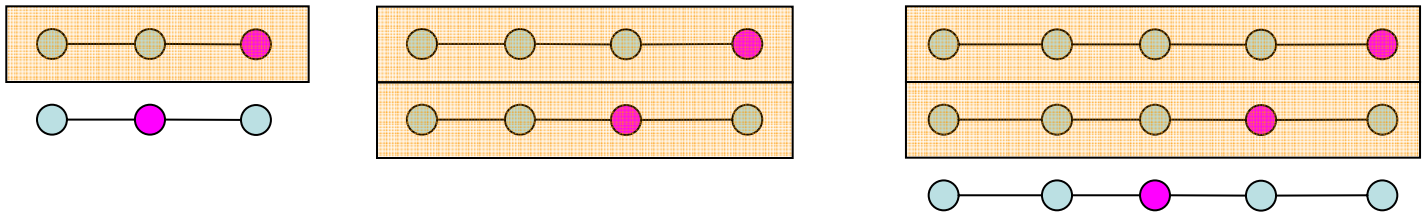
Some Special Graphs

- **Thm:** A ring-graph with a single leader is never controllable.



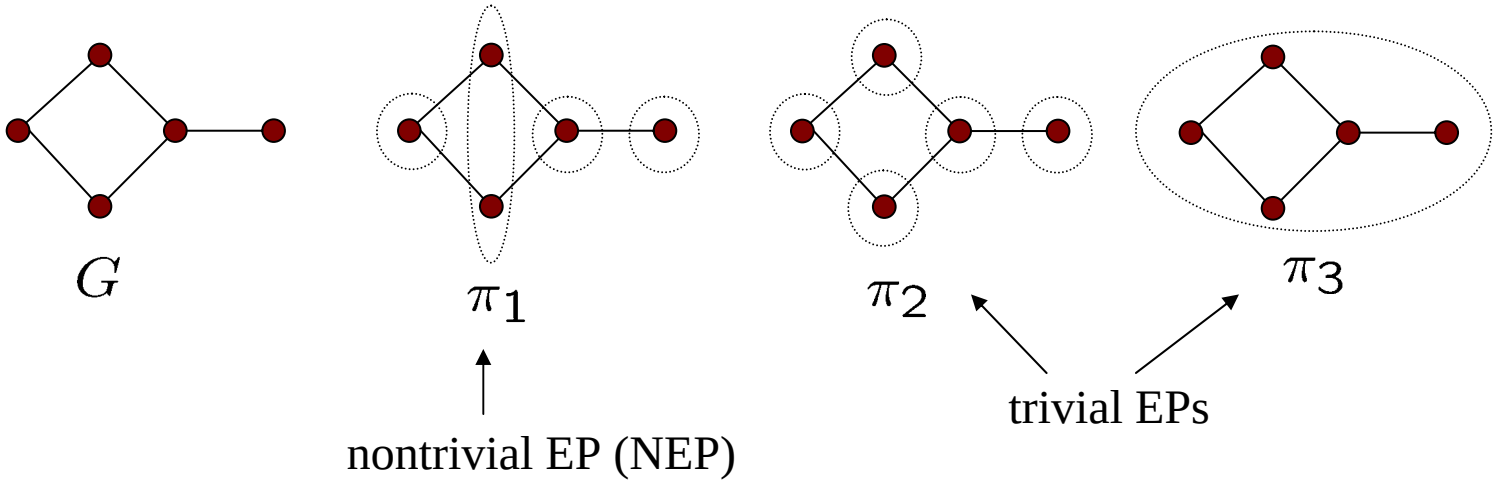
- **Thm:** A line-graph with a single leader is controllable if it has an even number of nodes or the leader is not the middle node

controllable



Multiple Leaders

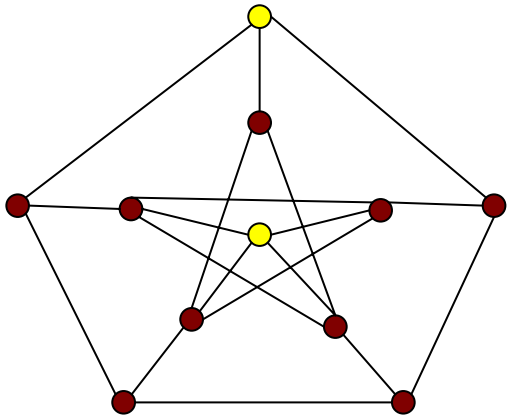
- The symmetry idea breaks down with multiple leaders. For this we need the generalization of ***equitable partitions***
- A *cell* $C \subset \mathcal{N}$ is a subset of the node set. A *partition* of the graph is a grouping of the nodes into different cells.
- A partition π with cells $C_1, \ldots, C_r$ is *equitable* if each node in C_i has the same number of neighbors in C_j



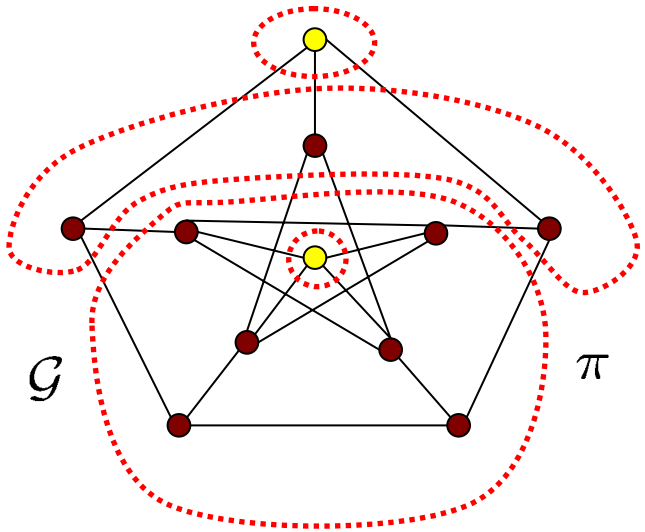
Multiple Leaders

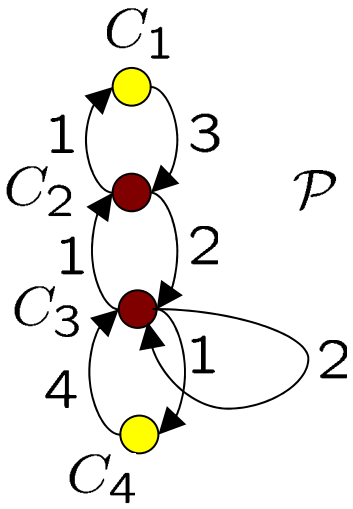
- **Thm**: Given a connected graph $\mathcal{G}$. The system (with nearest-neighbor follower dynamics) is not controllable if there exist NEPs that leave the leader nodes invariant

- Example: Consider the “modified Peterson graph”



Multiple Leaders

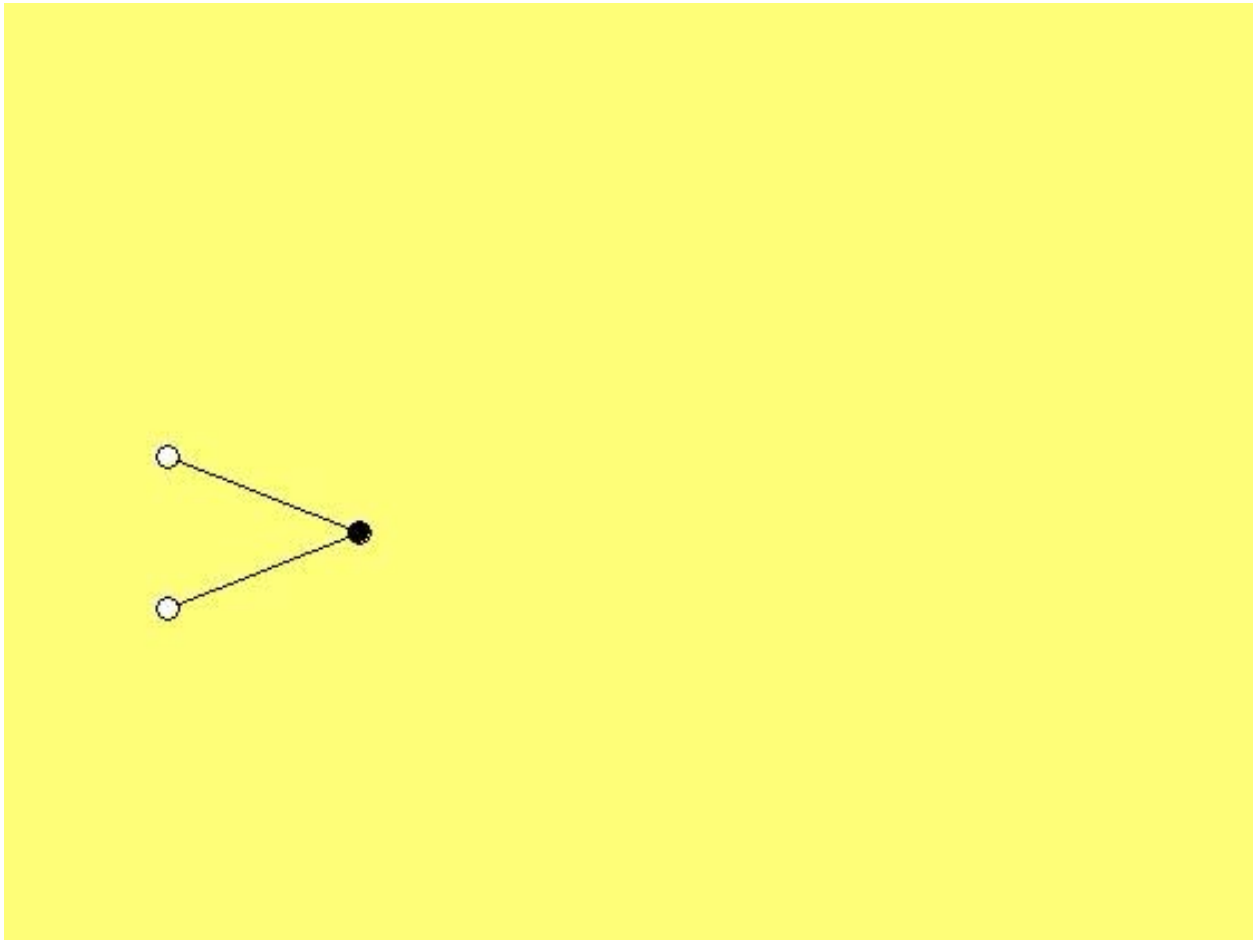




partition graph

Theorem: The controllable subspace is exactly given by the quationt (partition) graph and the uncontrollable part decays exponentially.

Controllable Subspace



GRAPH-BASED CONTROL LEADER NETWORKS

CONTROL ISSUES

Quasi-Static Equilibrium Processes

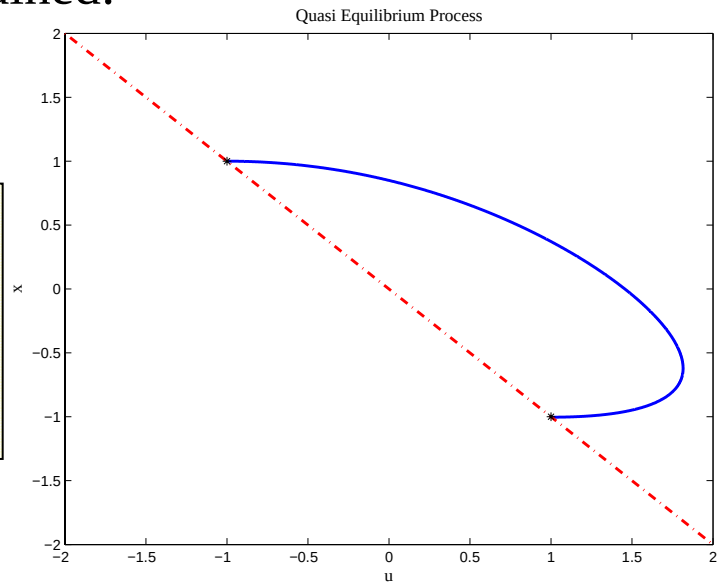
- As we've seen, rather than considering the leader's velocity as the input, we can consider its position (and insist on smooth inputs) in order to get the new system

$$\dot{x} = -\mathcal{L}_f x - \ell u \longleftarrow \left(\mathcal{L}_f \succ 0\right)$$

- For a fixed leader position, the corresponding asymptotically stable (quasi-static) equilibrium point is obtained:

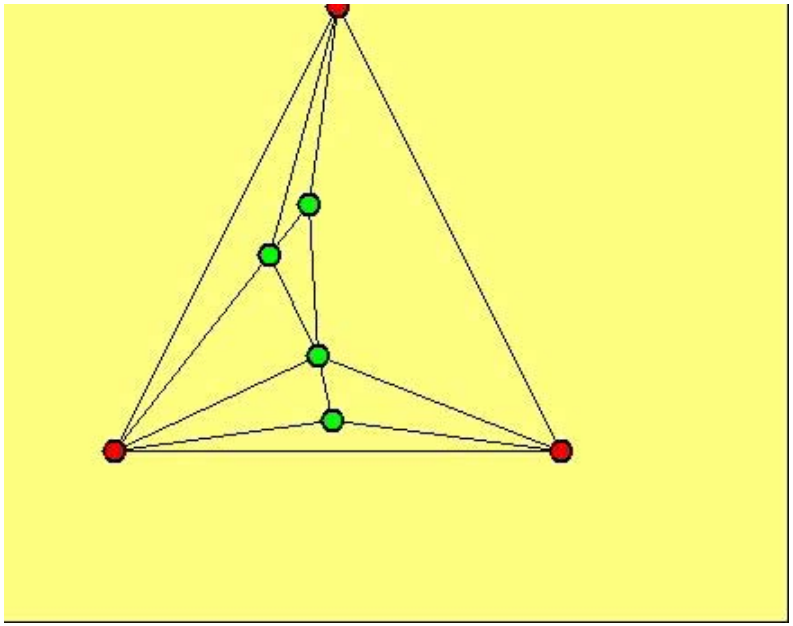
$$x_e = -\mathcal{L}_f^{-1} \ell u_e$$

Problem: Drive from one such equilibrium point to another while minimizing a quadratic performance index



Quasi-Static Equilibrium Processes

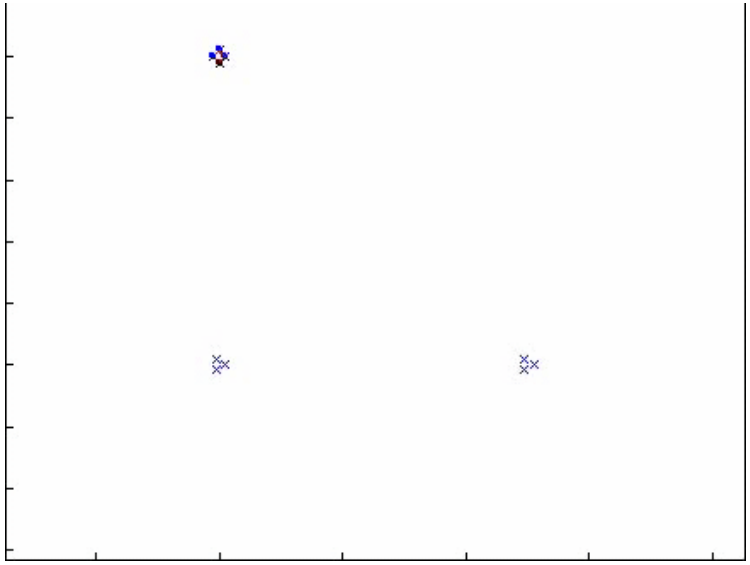
- It is straightforward to solve this problem using standard, optimal control



- What about more elaborate control problems?

General Optimal Control Problems

- We can of course solve general optimal control problems for leader-based networks

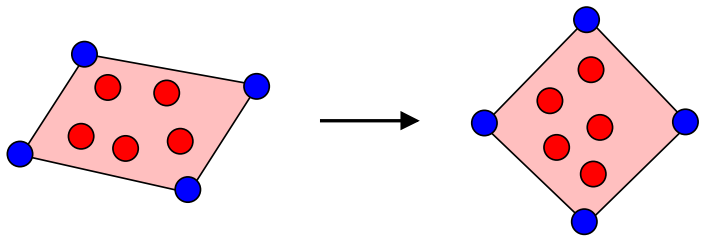


Containment Control

- A special problem for leader-based networks in which controllability does not matter is that of ***containment control***. (No point-to-point transfer)

- **Containment Control:** Ensure that the followers are contained in a given set/area during the maneuver.

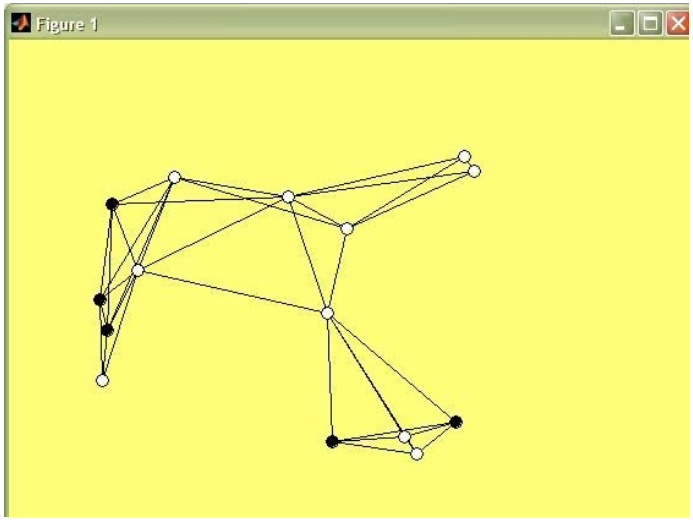
- **Our Problem:** Make them stay in the convex hull spanned by the leaders.



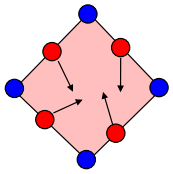
Multiple Stationary Leaders

Theorem: The follower robots will converge to the convex hull spanned by the static leader robots, i.e.

$$x_i^\star \in \Omega_L = \mathrm{Co}(\{x_j,\ x_j \text{ leader}\})$$



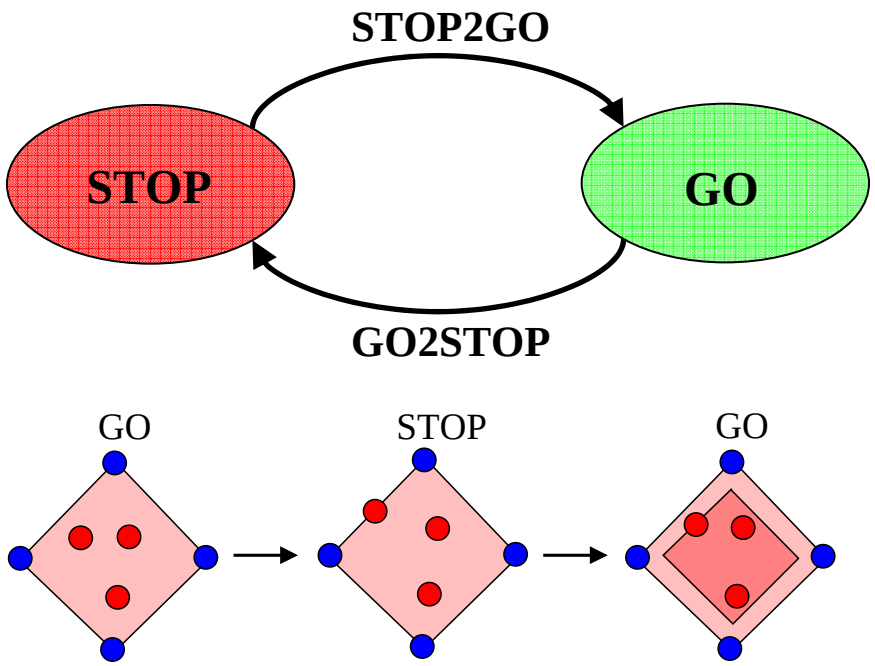
"Corollary"



Throughout the maneuver, the followers' velocities will never point away from Ω_L on the boundary.

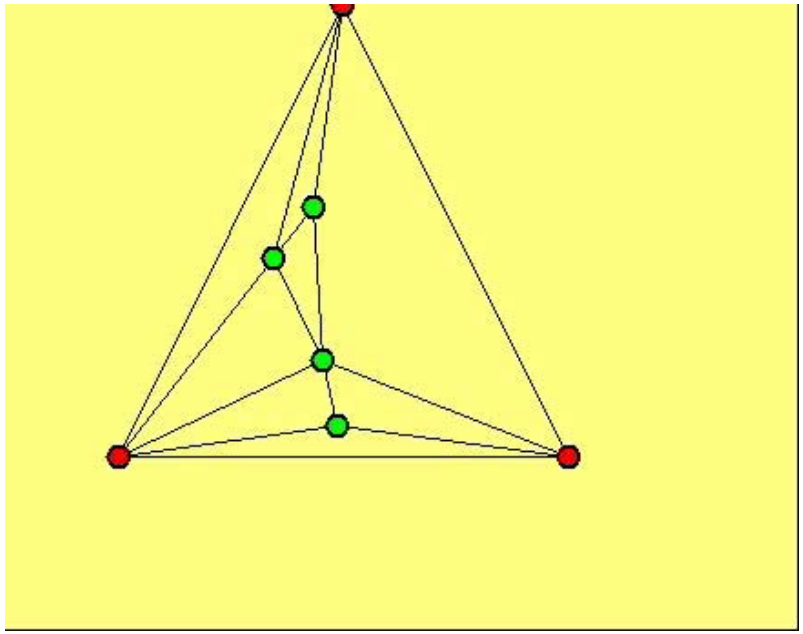
Hybrid Containment Control

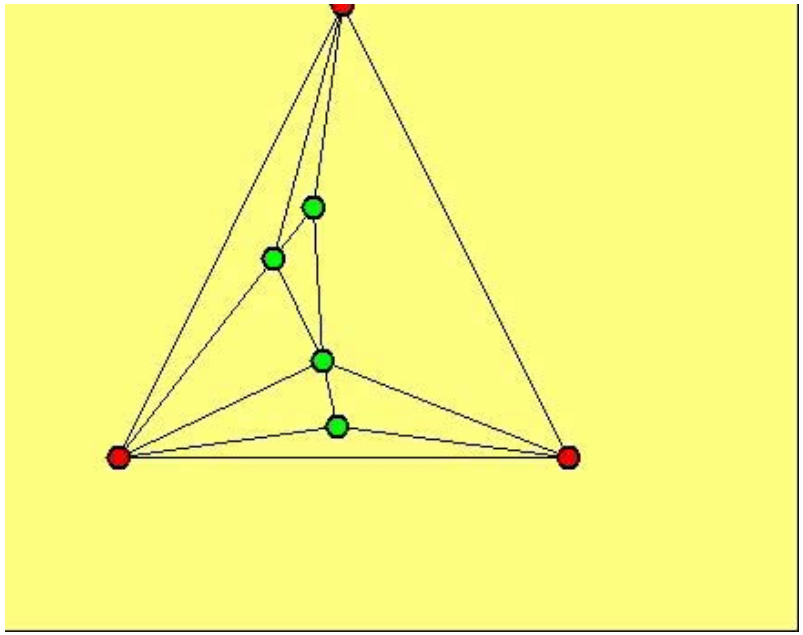
- One can now use a hybrid, STOP-GO control policy for the leaders in order to ensure containment:



- By introducing ***hysteresis***, we can guarantee that the system is non-Zeno:
- **Problem:** We may stay in the STOP-mode forever, i.e. the system is not live.
- **Solution:** If we are willing to accept that the followers can be allowed to be slightly outside of the leader-polytope we can prove liveness as well as convergence!

Example

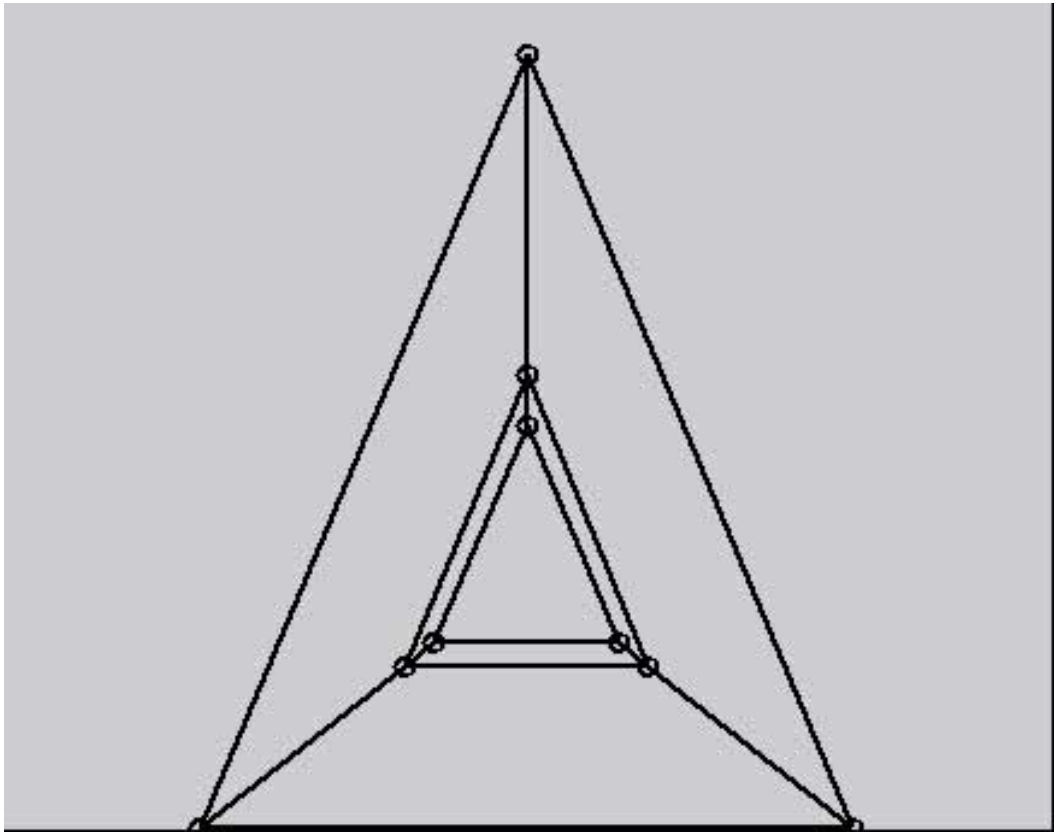




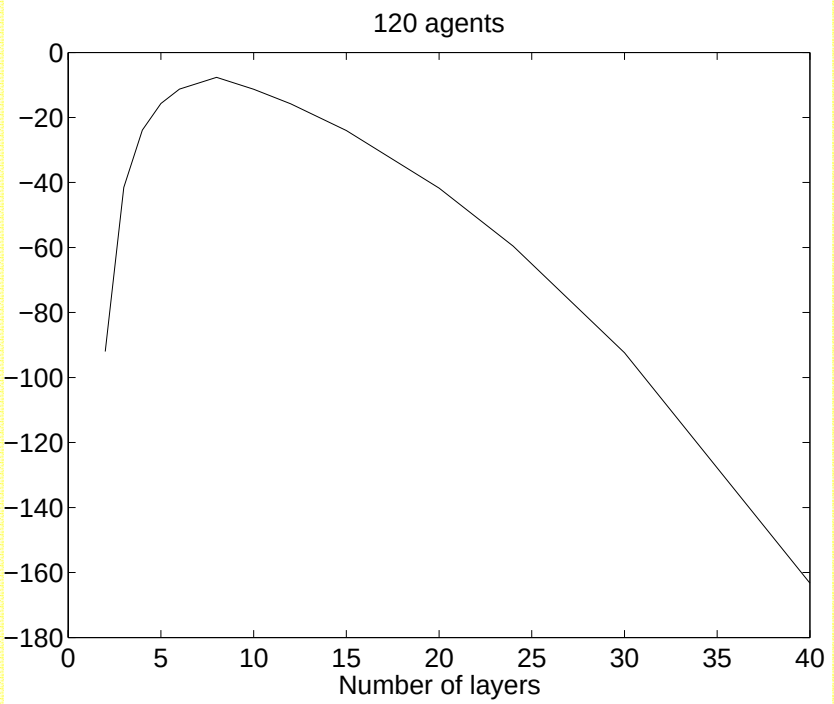
No Containment
(GO-mode only)

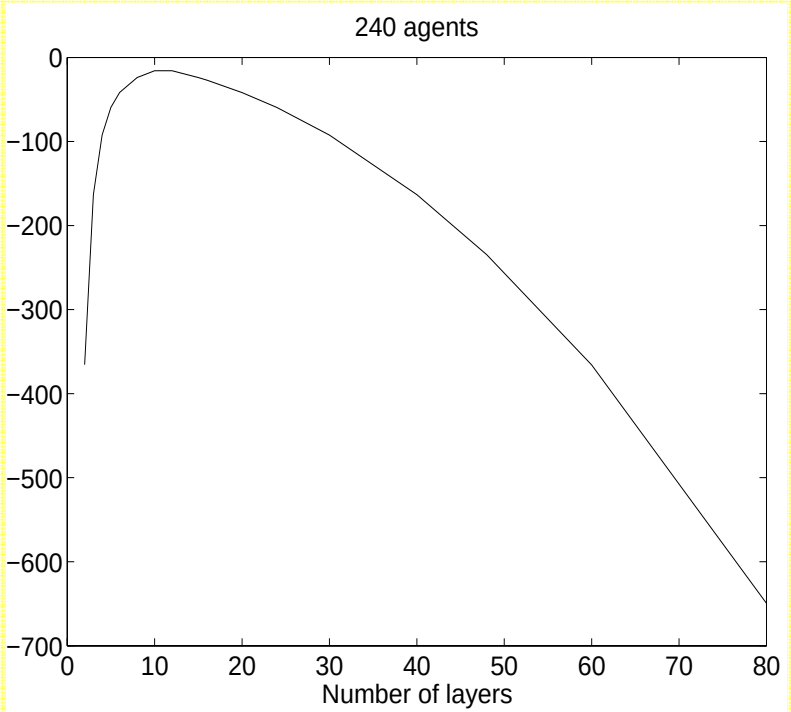
Containment
(STOP-GO)

Hierarchical Version



Complexity Considerations





$$J_1 = -\frac{1}{\lambda_2} - C_K$$

Conclusions

- The graph is a useful and natural abstraction of the interactions in networked control systems
- Tools and objects from algebraic graph-theory (and even spectral graph-theory) such as the graph Laplacian can be used to solve consensus, formation control, and coverage control problems
- By introducing leader-nodes, the network can be "reprogrammed" to perform multiple tasks such as move between different spatial domains
- Example problems include
  - Quasi-static equilibrium processes
  - Containment control problems
- Controllability based on graph-theoretic properties were introduced through symmetry groups

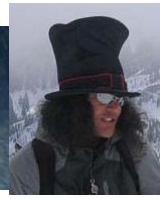
Acknowledgements

Joint work with

- **Meng Ji**, **Simone Martini**, Musad Haque, Abubakr Muhammad, Staffan Bjorkenstam, Brian Smith (current and previous
- **Giancarlo Ferrari-Trecate, Annalisa Buffa** (containment control)
- **J.M. McNew, Eric Klavins** (coverage control)
- **Mehran Mesbahi, Amirreza Rahmani** (controllability)





This work was sponsored by the **US National Science Foundation (NSF)** [ants], the **US Army Research Office** [formations], the **National Aeronautics and Space Administration (NASA)** [coverage], and **Rockwell Collins** [HSI]

THANK YOU!