Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks

Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, Alejandro Ribeiro

Motivation

- Enable distributed controllers for large networks of mobile robots with interacting dynamics and sparsely available communications

- Learn local controllers that require only local information and local communications at test time by imitating centralized controllers that use global information at training time
- **Learning as the tool of choice** for designing approximately optimal behaviors
- Testing in AirSim simulates quadrotors with latency and complex dynamics.
- We propose a solution leveraging:
  - $\Rightarrow$ Imitation learning
  - $\Rightarrow$ Offline training
  - $\Rightarrow$ Graph Neural Networks

Imitation Learning

- Given the centralized expert policy $\pi^*(\mathbf{x}_n)$
- Collect local information history from a k-hop neighborhood of node i :

$$\mathcal{H}_{in} = \bigcup_{k=0}^{K-1} \left\{ \mathbf{x}_{j(n-k)} : j \in \mathcal{N}_i^k \right\}$$

- Learn a decentralized policy by imitating the centralized controller

$$\mathbf{H}^* = \underset{\mathbf{H}}{\operatorname{argmin}}\, \mathbb{E}^{\pi^*}\left[\mathcal{L}\left(\pi\left(\mathcal{H}_{in}, \mathbf{H}\right), \pi^*(\mathbf{x}_n) \right) \right]$$

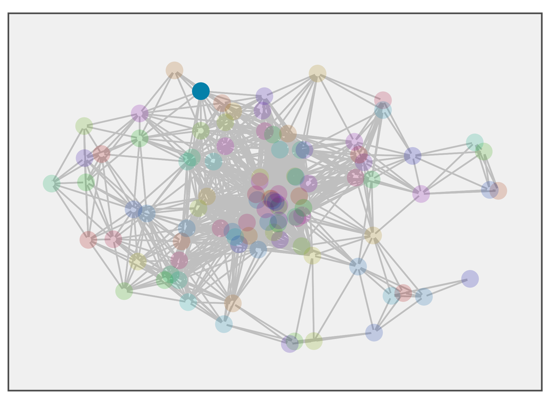
- In practice, the DAgger algorithm was used for imitation learning
- Aggregate information at nodes through successive averaging using the graph adjacency matrix $\mathbf{S}$, where $\mathbf{y}_{0n} = \mathbf{x}_n$:

$$\left[\mathbf{y}_{kn}\right]_i = \left[\mathbf{S}\mathbf{y}_{k-1(n-1)}\right]_i = \sum_{j=1, j\in\mathcal{N}_{in}} \left[\mathbf{S}\right]_{ij} \left[\mathbf{y}_{k-1 n-1}\right]_j$$

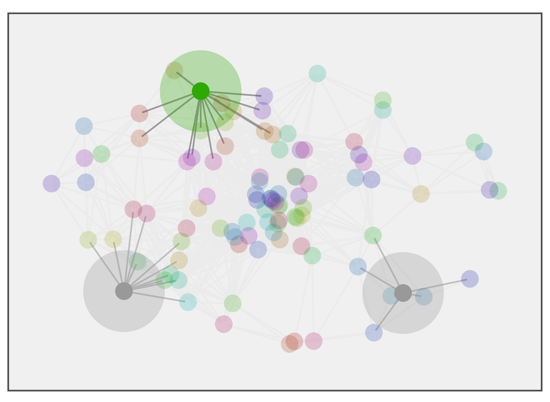
- Computed using local operations that respect the information structure of the distributed system

Delayed Aggregation Graph Neural Network

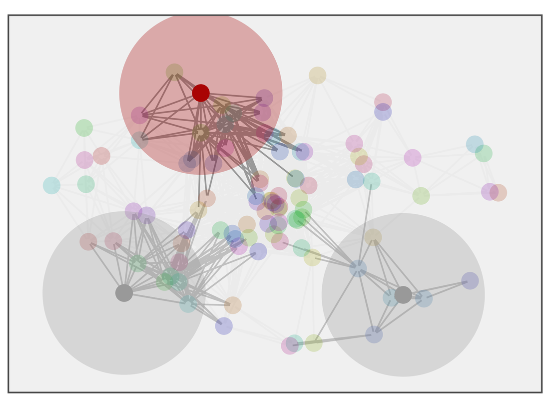
- We extend aggregation graph neural networks [GGMRL19] to time varying signals and time varying network support.



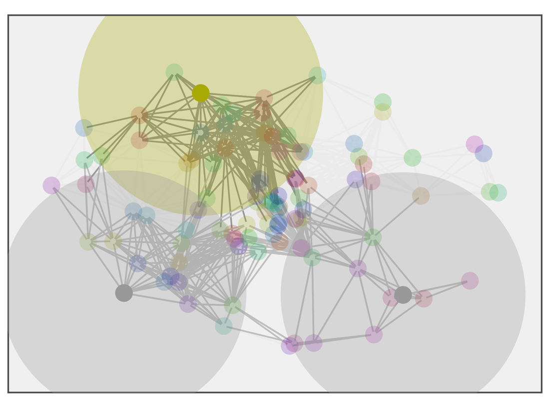
(a) Local state (K=1)



(b) 1-hop neighborhood (K=2) delayed by T_s



(c) 2-hop neighborhood (K=3) delayed by $2T_s$



(d) 3-hop neighborhood (K=4) delayed by $3T_s$

- We learn a single common local controller which exploits information from distant teammates using only local communication interchanges.

Flocking Formulation

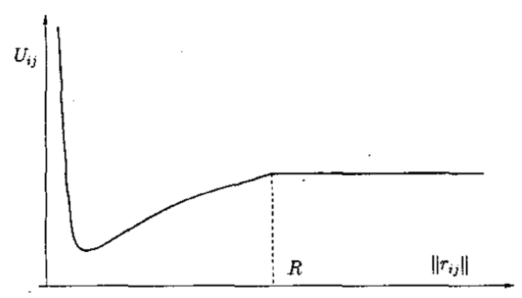
- Flocking in a team of robots:
  - $\Rightarrow$ Aligned robot velocities $\mathbf{v}_i$,
  - $\Rightarrow$ Regular inter-robot spacing $\mathbf{r}_{ij}$
- Acceleration-controlled discrete-time dynamics in 2D with $T_s = 0.01$ s
- Positions, $\mathbf{r}_{i,n+1} = \mathbf{r}_{i,n} + T_s\mathbf{v}_{i,n} + 0.5T_s^2\mathbf{u}_{i,n}$
- Velocities, $\mathbf{v}_{i,n+1} = \mathbf{v}_{i,n} + T_s\mathbf{u}_{i,n}$
- The cost is the variance in agent velocities

$$C = \frac{1}{N}\sum_{n=1}^{T}\sum_{j=1}^{N}\left\| \mathbf{v}_{j,n} - \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{v}_{i,n}\right]\right\|_2^2 .$$

Flocking Controllers

- Communication range R
- Regulate spacing among agents, $\|\mathbf{r}_{ij}\|_2^2$, using pairwise potentials U_{ij} [TJP03]

$$U_{ij} = \begin{cases} \frac{1}{\|\mathbf{r}_{ij}\|^2} + \log\|\mathbf{r}_{ij}\|^2, & \|\mathbf{r}_{ij}\| < R, \\ V_R, & \|\mathbf{r}_{ij}\| \geq R \end{cases}$$



- **Expert**: Global controller that relies on communication among all agents:

$$\mathbf{u}_i^* = -\sum_{j=1}^{J}(\mathbf{v}_i - \mathbf{v}_j) - \sum_{j=1}^{J}\nabla_{\mathbf{r}_i}U_{ij}$$

- **Baseline**: Local controller uses only information from neighbors $\mathcal{N}_i$ of node i:

$$\mathbf{u}_i^\dagger = -\sum_{j\in\mathcal{N}_i}(\mathbf{v}_i - \mathbf{v}_j) - \sum_{j\in\mathcal{N}_i}\nabla_{\mathbf{r}_i}U_{ij}$$

- Agents observe neighbors' positions and velocities with no time delay

$$\mathbf{x}_{i,n} = \left[\sum_{j\in\mathcal{N}_i}(\mathbf{v}_{i,n} - \mathbf{v}_{j,n}), \sum_{j\in\mathcal{N}_i}\frac{\mathbf{r}_{ij,n}}{\|\mathbf{r}_{ij,n}\|_2^4}, \sum_{j\in\mathcal{N}_i}\frac{\mathbf{r}_{ij,n}}{\|\mathbf{r}_{ij,n}\|_2^2}\right]$$

Algorithm 1: Delayed Aggregation GNN at Agent i

1: **for** n=0,1,..., **do**

2: Receive aggregation sequences from neighbors

$$\mathbf{z}_{j(n-1)} = \left[\left[\mathbf{y}_{0(n-1)}\right]_j; \left[\mathbf{y}_{1(n-1)}\right]_j; \dots; \left[\mathbf{y}_{(K-1)(n-1)}\right]_j\right]$$

3: Update aggregation sequence components

$$\left[\mathbf{y}_{kn}\right]_i = \left[\mathbf{S}_n\mathbf{y}_{k(n-1)}\right]_i = \sum_{j=1, j\in\mathcal{N}_{in}}\left[\mathbf{S}_n\right]_{ij}\left[\mathbf{y}_{k(n-1)}\right]_j$$

4: Observe system state $\mathbf{x}_{in}$

5: Update local aggregation sequence

$$\mathbf{z}_{in} = \left[\mathbf{x}_{in}^T; \left[\mathbf{y}_{1n}\right]_i; \dots; \left[\mathbf{y}_{(K-1)n}\right]_i\right]$$

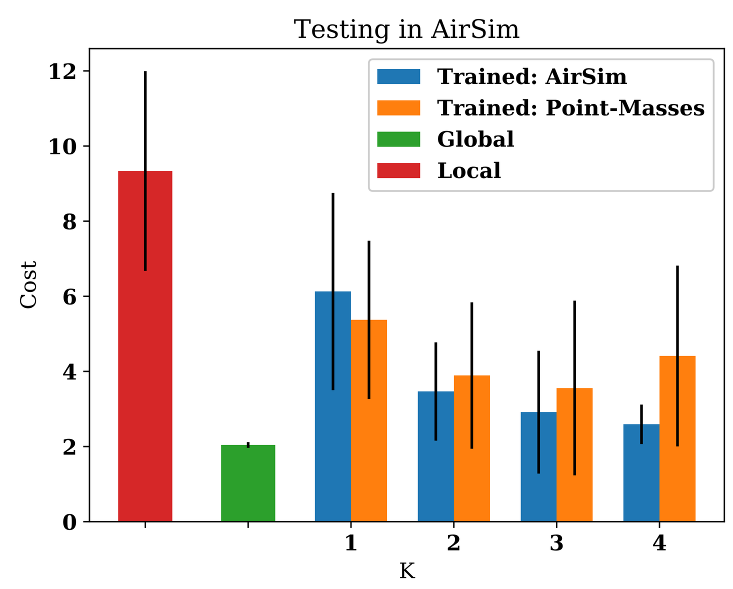
6: Compute local action using the learned controller

$$\mathbf{u}_{in} = \pi(\mathbf{z}_{in}, \mathbf{H})$$

7: Transmit local aggregation sequence $\mathbf{z}_{in}$ to neighbors $j \in \mathcal{N}_{in}$

8: **end for**

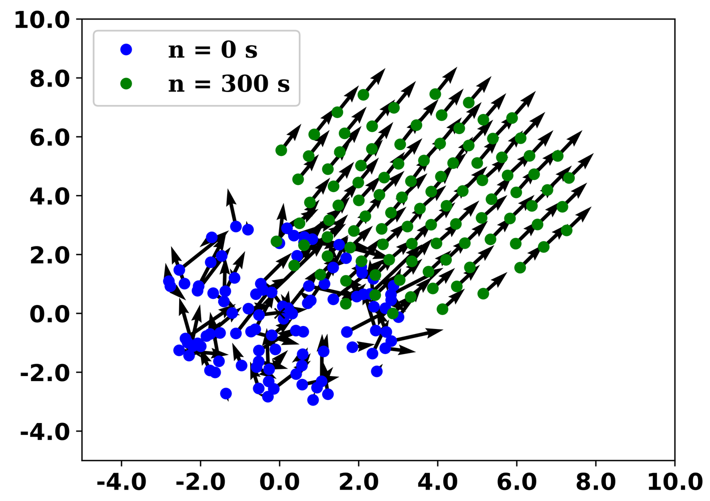
AirSim Results



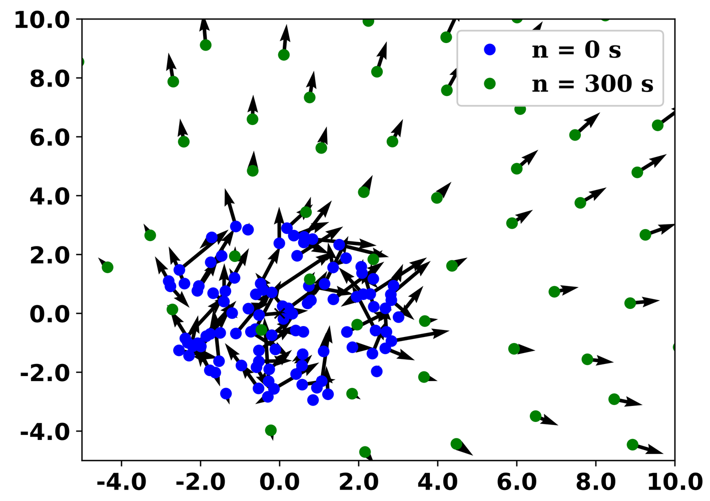
- Four models were trained in AirSim and four on stochastic point masses tuned to the parameters of the simulation, and then all were tested in AirSim.

Flocking Trajectories

- The GNN ($K = 3$) maintains a cohesive flock, while the local controller allows the flock to scatter.



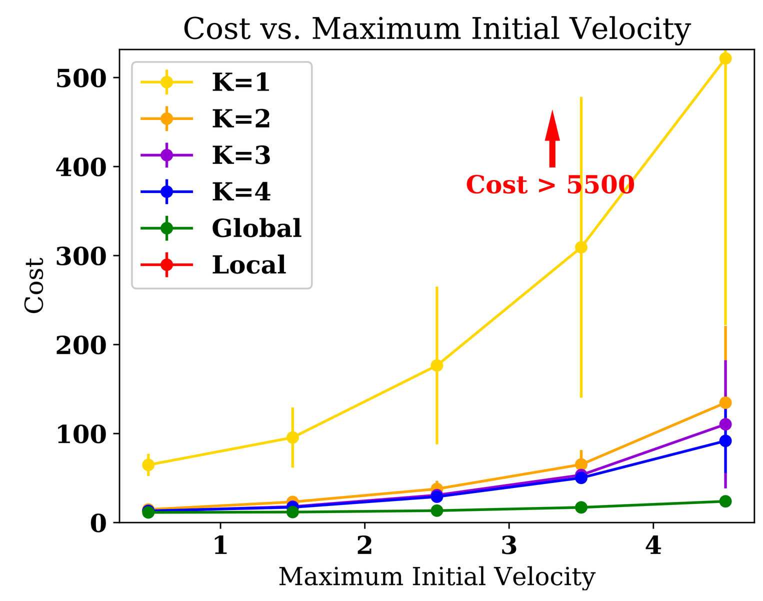
(a) Flock positions using the GNN

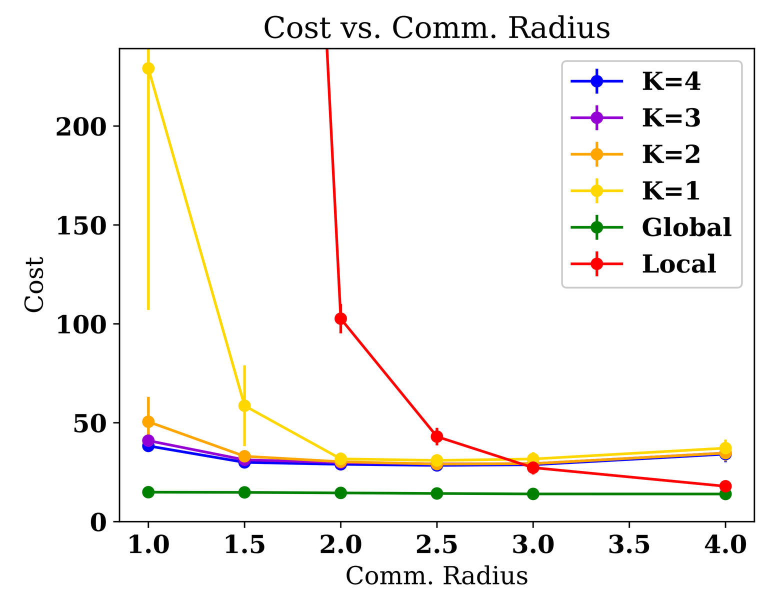


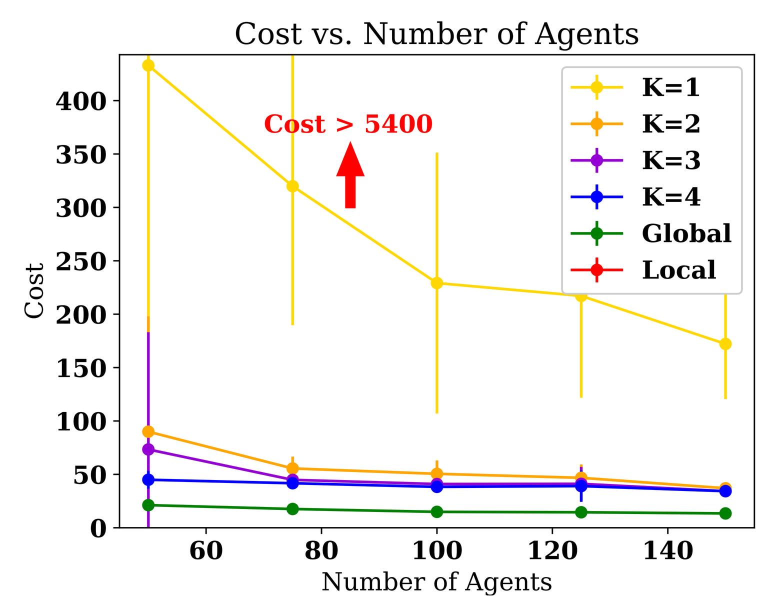
(b) Flock positions using the baseline

Point Mass Results

- The flock's maximum initial velocities, communication radius, and the number of agents are key parameters affecting the cost.







- Aggregation is most useful for fast-moving agents and small communication ranges.

F. Gama, A. G. Marques, A. Ribeiro, and G. Leus, *Aggregation graph neural networks*, 44th Acoustics, Speech and Signal Processing (ICASSP), IEEE Int. Conf. on (Brighton, UK), IEEE, 12-17 May 2019.

Herbert G Tanner, Ali Jadbabaie, and George J Pappas, *Stable flocking of mobile agents part ii: dynamic topology*, Decision and Control, 42nd Int. Conf. on, vol. 2, IEEE, 2003, pp. 2016–2021.