Summary of: Learning Decentralized Controllers for Robotic Swarms with Graph Neural Networks

Written by: Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro

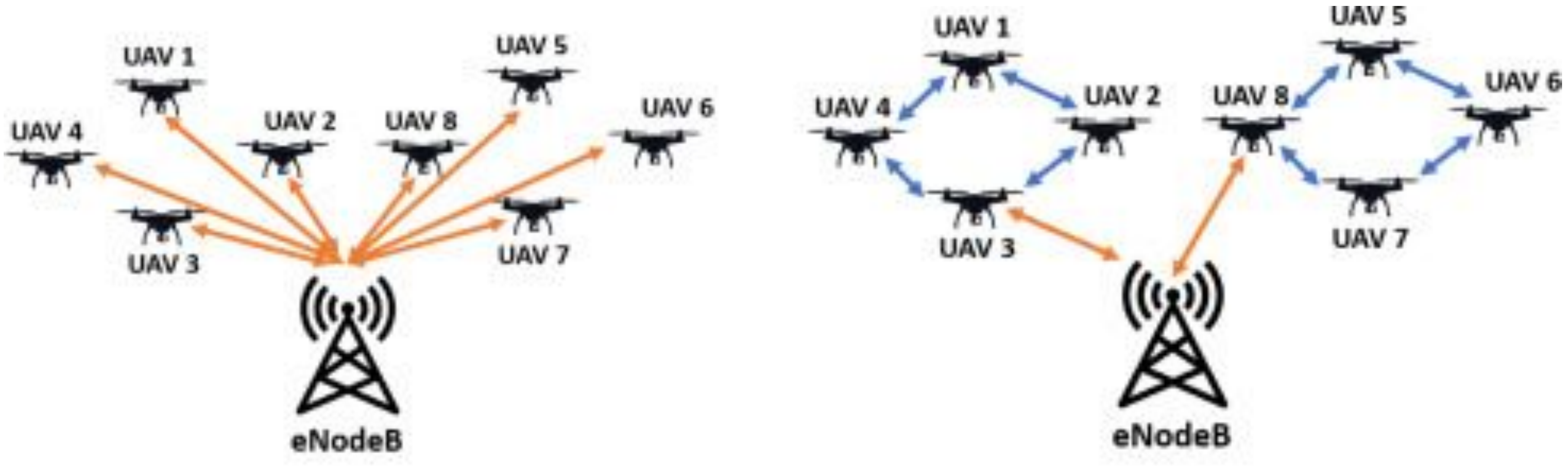
Presented by: Austin Stone

Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks
Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, Alejandro Ribeiro
Conference on Robot Learning, 2020, pp. 671-682

Outline

- Motivation
- Background
- Model
- Theoretical Results
- Simulation Results
- Summary
- Future Work

Motivation



(a) Centralized Model (b) Decentralized Model

Relevant Background Literature

Referenced Papers:

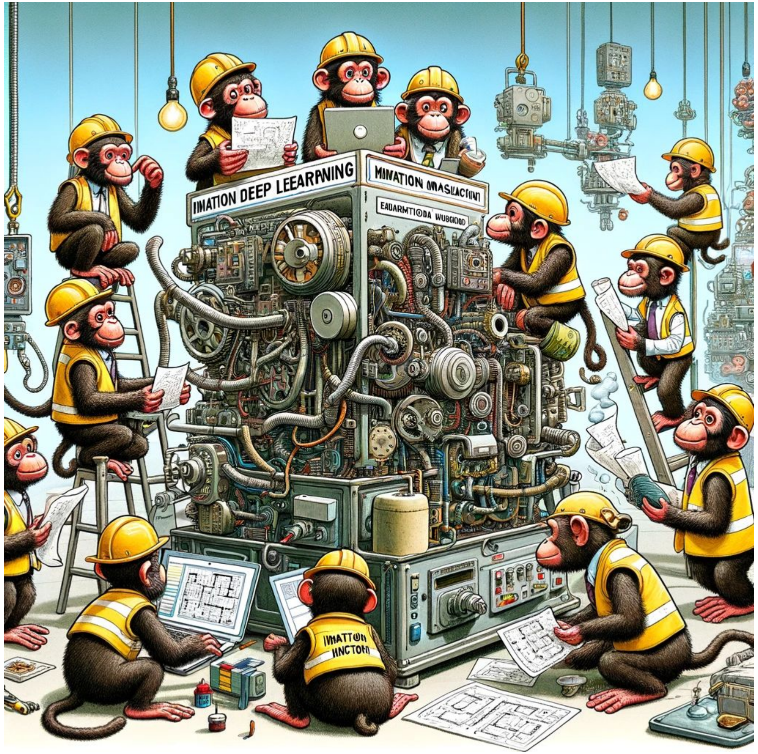
- Imitation Control - J. Paulos, S. W. Chen, D. Shishika, and K. Vijay. Decentralization of multiagent policies by learning what to communicate. In 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, May 2019.
- Aggregation NN - F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. Convolutional neural network architectures for signals supported on graphs. IEEE Trans. Signal Process., 67(4):1034–1049, Feb. 2019.
- Local Controller/Global Controller - H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Stable flocking of mobile agents part ii: dynamic topology. In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on, volume 2, pages 2016–2021. IEEE, 2003.

Main Contribution:

- Train Graph NN to do flocking

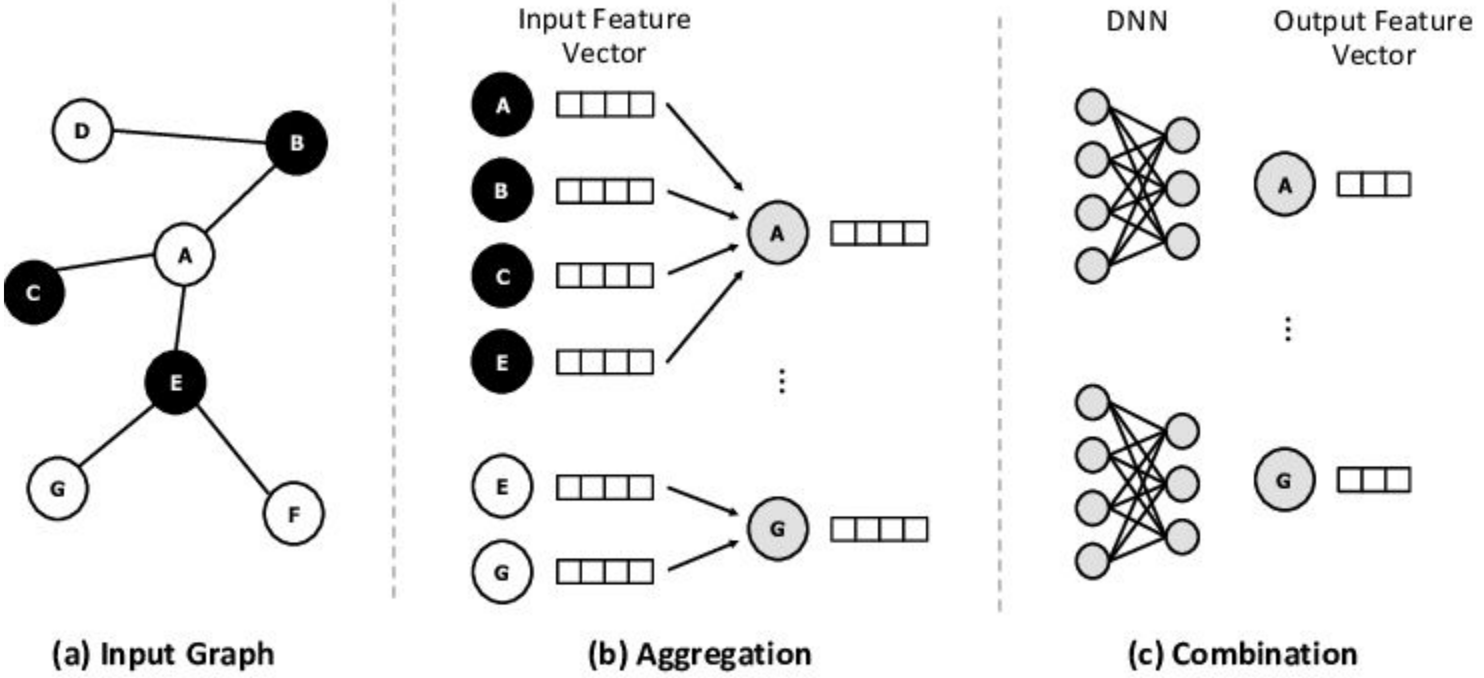
Imitation Control

Monkey See, Monkey Do



Presenter generated with Dalle3

Aggregation Graph Neural Networks



Global Controller

$$U(r_i, r_j) = \begin{cases} \frac{1}{\|r_{ij}\|^2} - \log(\|r_{ij}\|^2) & \text{if } \|r_{ij}\| > \rho \\ \frac{1}{\rho^2} - \log(\rho^2) & \text{otherwise} \end{cases}$$

$$\nabla_{r_i} U(r_i, r_j) = \begin{cases} -\frac{2r_{ij}}{\|r_{ij}\|^4} + \frac{2r_{ij}}{\|r_{ij}\|^2} & \text{if } \|r_{ij}\| > \rho \\ 0 & \text{otherwise} \end{cases}$$

$$u_i^* = -\sum_{j=1}^{N} (v_i - v_j) - \sum_{j=1}^{N} \nabla_{r_i} U(r_i, r_j).$$

Local Controller

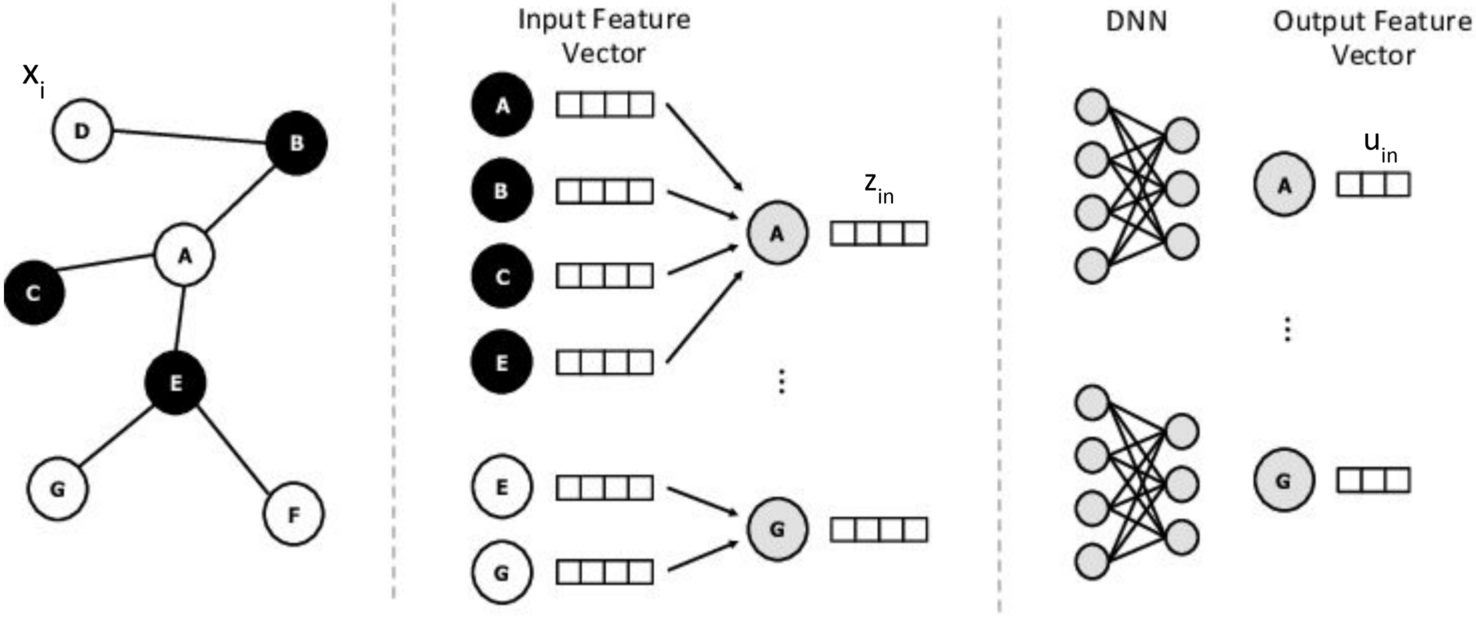
$$u_i^\dagger = -\sum_{j \in \mathcal{N}_i} (v_i - v_j) - \sum_{j \in \mathcal{N}_i} \nabla_{r_i} U(r_i, r_j).$$

Back to the Future!

Summary

$$\sigma^{(\ell)}\left(H^{(\ell)} z_{in}^{(\ell-1)}\right)$$

y



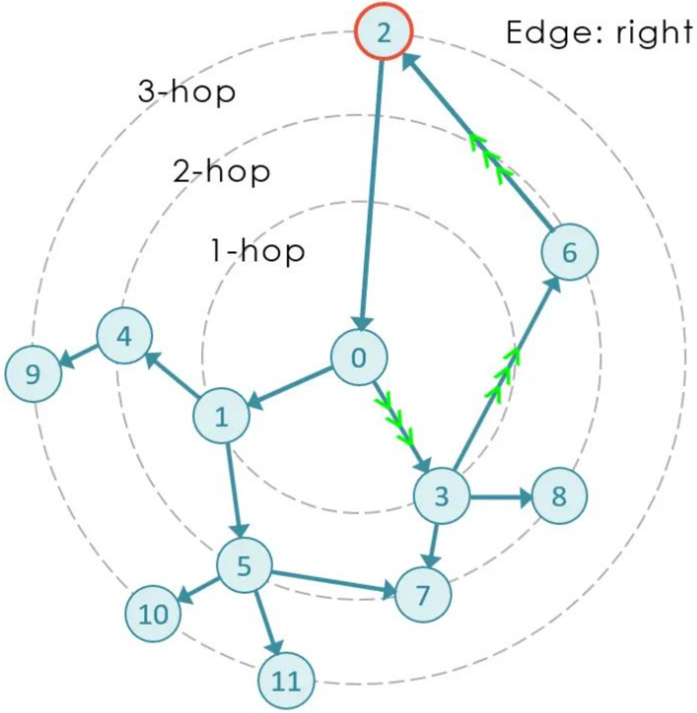
(a) Input Graph (b) Aggregation (c) Combination

S_n

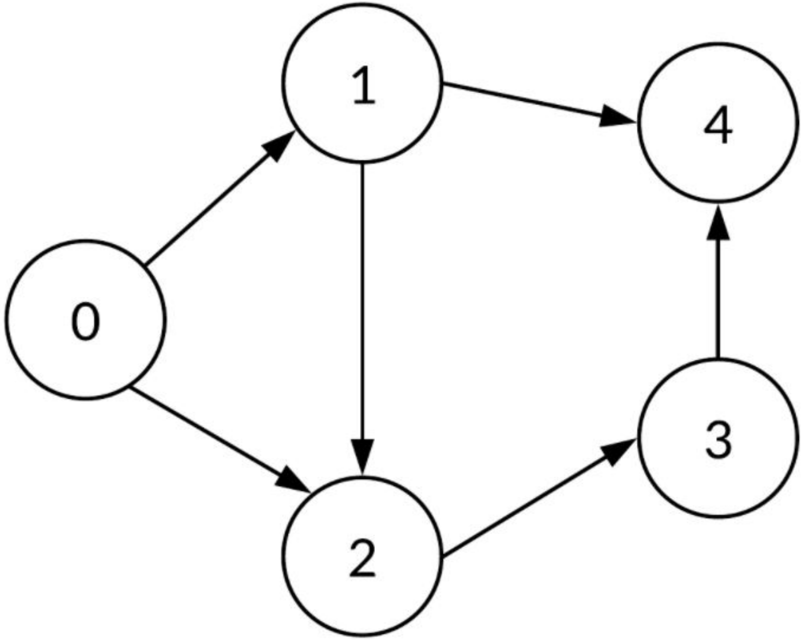
k-hops

$$\mathcal{N}_{in}^{k} = \{j' \in \mathcal{N}_{j(n-1)}^{k-1} \,|\, j' \in \mathcal{N}_{in}\}$$

$$\mathcal{H}_{in} = \bigcup_{k=0}^{K-1} \{x_{j(n-k)} \,:\, j \in \mathcal{N}_{in}^{k}\}$$



Graph Shift Operator



~~Adjacency Matrix~~ Graph Shift Operator [S_n]

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Feature Matrix

$$y_{kn} = S_n y_{(k-1)(n-1)},$$

$$[y_{0n}]_i = [x_n]_i = x_{in}^T$$

$$z_{in} = \left[[y_{0n}]_i\,; [y_{1n}]_i\,; \ldots ; \left[y_{(K-1)n}\right]_i\right]$$

Feature Vector

$$[x_n]_i = \left[\sum_{j \in \mathcal{N}_i} (v_{i,n} - v_{j,n}), \sum_{j \in \mathcal{N}_i} \frac{r_{ij,n}}{\|r_{ij,n}\|^4}, \sum_{j \in \mathcal{N}_i} \frac{r_{ij,n}}{\|r_{ij,n}\|^2} \right]$$

Final Control / Loss Function

$$z_{in}^{(\ell)} = \sigma^{(\ell)}\left(H^{(\ell)} z_{in}^{(\ell-1)}\right)$$

$$H^* = \operatorname*{argmin}_H \sum_{(x_n, \pi^*(x_n)) \in \mathcal{T}} \mathcal{L}(u_n, u_n^*)$$

DAgger (Dataset Aggregation)

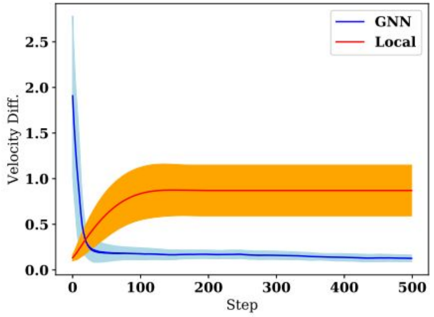
For each iteration:

- Generate a trajectory where, for each timestep, the expert policy is selected with probability β and the learner's policy is selected with probability 1 - β.
- Train using the generated trajectory.
- Decay β by a factor of γ until it reaches β_{min}.

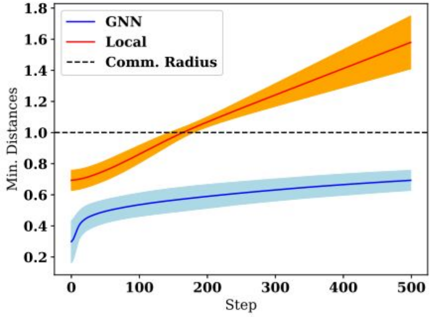
Theoretical Results

- Leans on the previous theoretical results demonstrating global controller convergence.
- Doesn’t show much theory as to why training works.

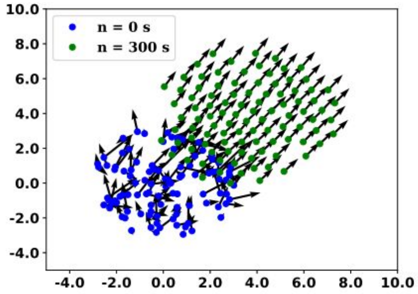
Simulation Results



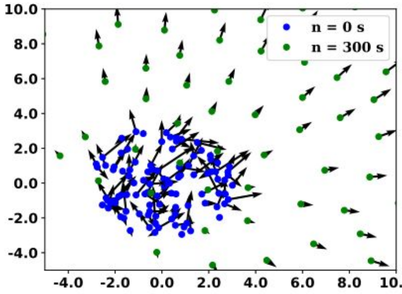
(a) Average difference in velocities



(b) Average minimum distance to a neighbor

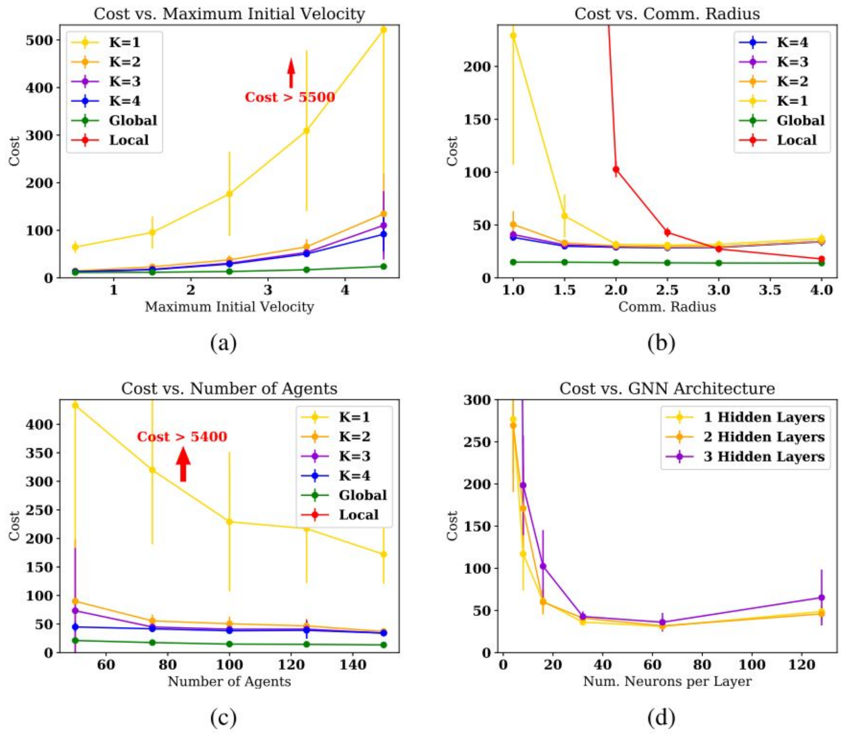


(c) Flock positions using the GNN



(d) Flock positions using the local controller

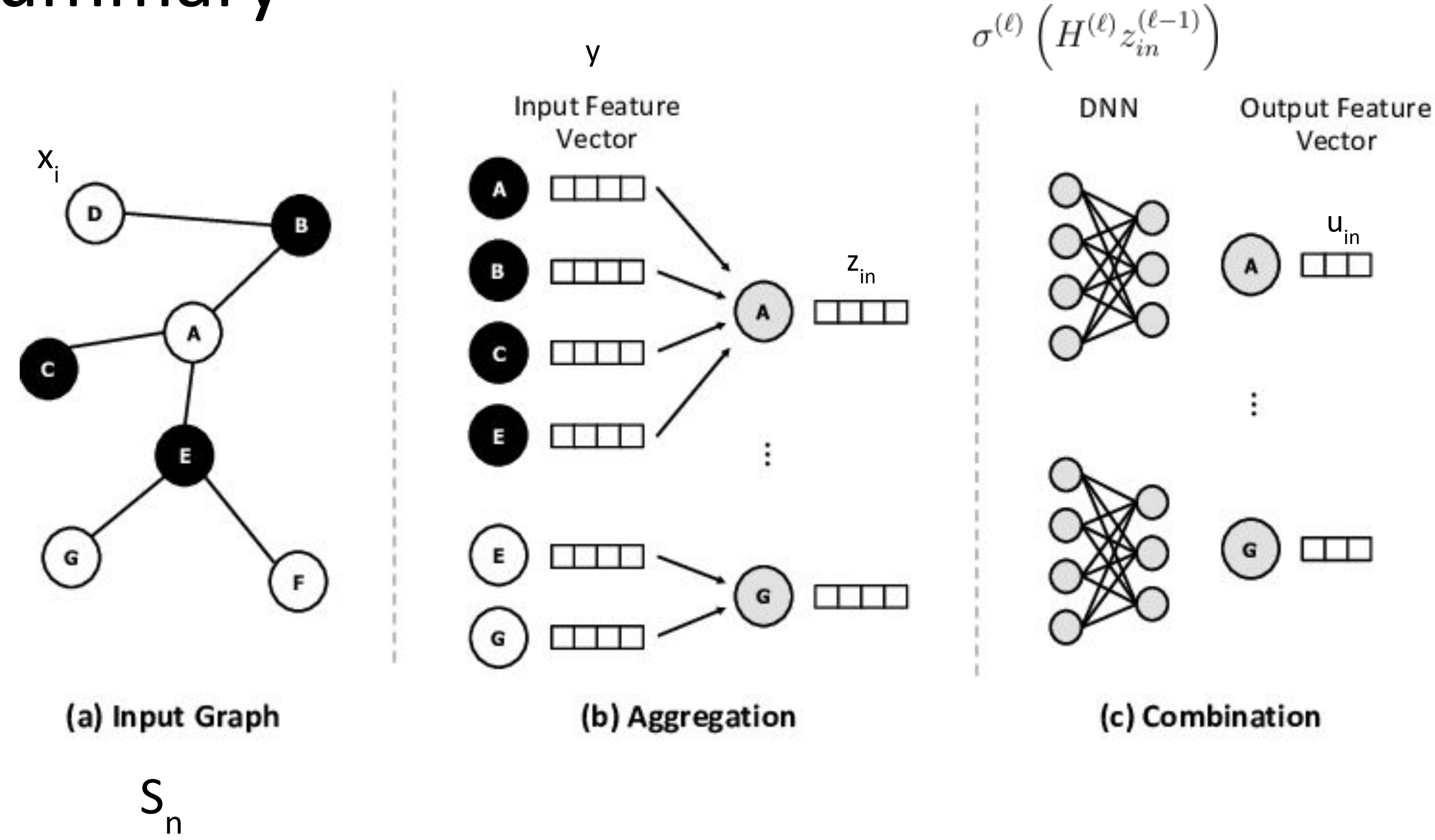
Simulation Results (Continued)



My Results/Their Results Again

Their code can be found at:
https://github.com/katetolstaya/multiagent_gnn_policies/tree/master

Summary



$\sigma^{(\ell)}\left(H^{(\ell)} z_{in}^{(\ell-1)}\right)$
y
Input Feature Vector
DNN
Output Feature Vector
x_i
D
B
A
C
E
G
F
z_{in}
u_{in}
(a) Input Graph
(b) Aggregation
(c) Combination
S_n

Future Work

- Hardware verification
- Many other imitation learning tasks
- Different Graph NN Architectures