Scalable Learning in Distributed Robot Teams

Doctoral Thesis Defense

Ekaterina (Kate) Tolstaya

April 21st, 2021

Penn Engineering

Robots' Localization and Planning are Centralized

"Robot Quadrotors Perform James Bond Theme" YouTube

Large Robot Teams Must Be Decentralized

Bottlenecks in scaling robot teams:

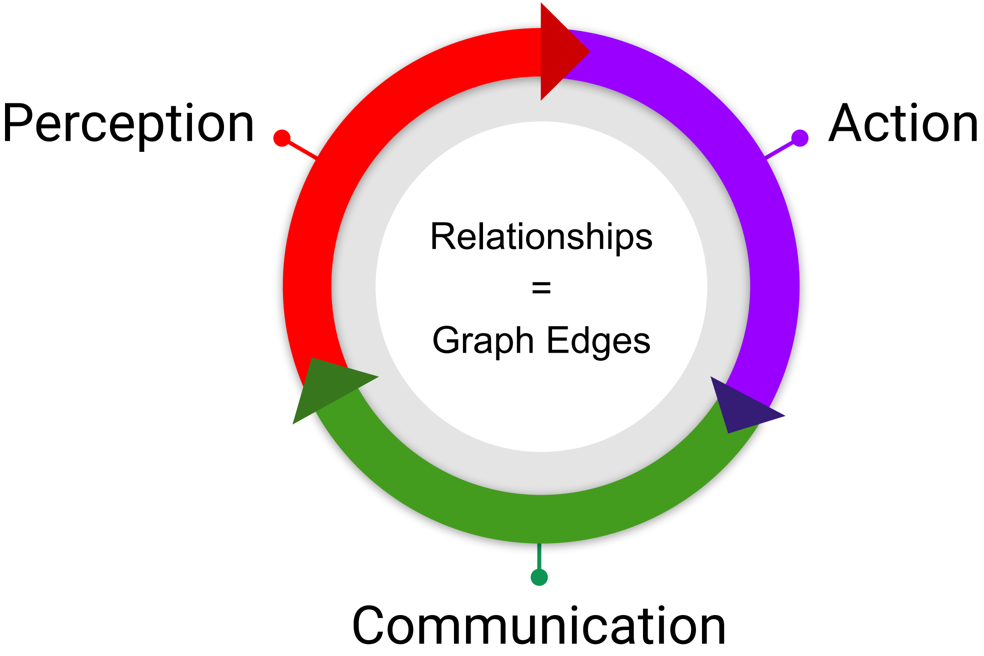
- Coordination
- Control
- Communication

Larger teams can be more efficient and resilient for:

- Exploration
- Mapping
- Search...

“Flock of Birds Create Beautiful Shapes in Sky” YouTube

Each robot in the team...

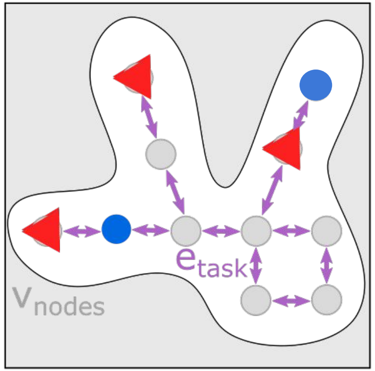


…is a node in a graph.

Outline

1) Coverage

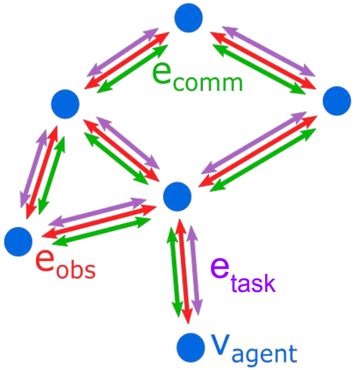
[Tolstaya et al.
IROS ‘21 (submitted)]



Learning to Coordinate

2) Flocking

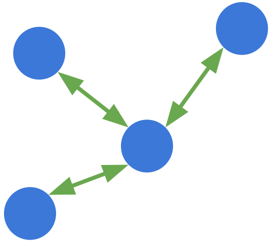
[Tolstaya et al.
CoRL '19]



Learning to Control

3) Data Distribution

[Tolstaya et al.
IROS ‘21 (submitted)]



Learning to Communicate

Defining a Graph Signal

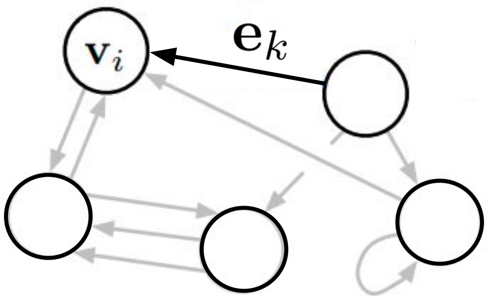
A graph signal $\mathcal{G} = (E, V)$ is defined by

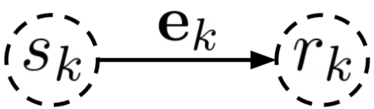
- The set of node features

$$V = \{\mathbf{v}_i\}_{i=1:N^v}$$

- The set of edge features and directed adjacency relationships

$$E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N^e}$$

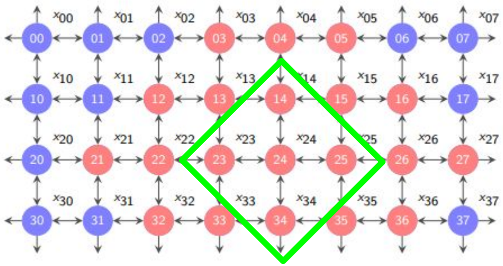




Following the formulation of Battaglia ’18, DeepMind’s Graph Nets framework

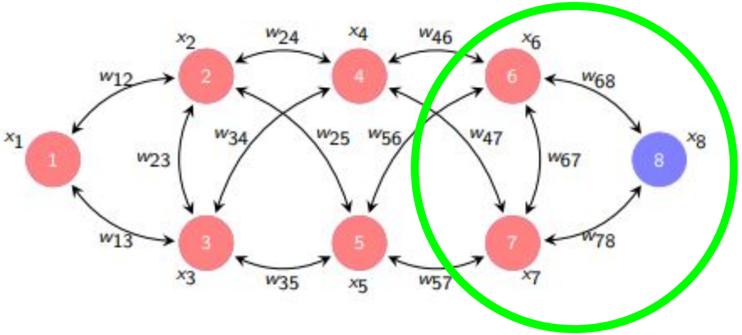
Convolutional NN

Graph Neural Network



CNNs apply filter on a *grid* graph.

Why not any graph?



The graph filter is a decentralized NN

Graph Networks

- Each **Graph Network** block, $GN(E, V)$, performs edge & node updates:

(a) Edge update

(b) Node update

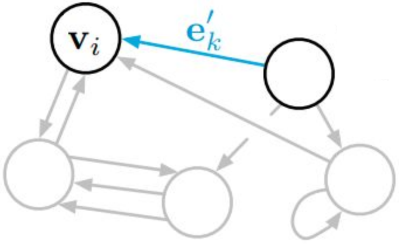
$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}) \quad := f^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$ per-edge update

$$\bar{\mathbf{e}}'_i = \rho^{e \to v}(E'_i)$$ aggregation from edge to vertex

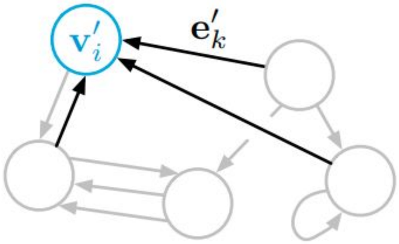
$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i) \quad := f^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i)$$ per-node update

Graph Networks

- Each **Graph Network** block, $GN(E, V)$, performs edge & node updates:



(a) Edge update



(b) Node update

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}) \quad := f^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}) \boxed{= \mathrm{NN}_e([\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}]) \quad \text{or} \quad = \mathbf{v}_{s_k}}$$

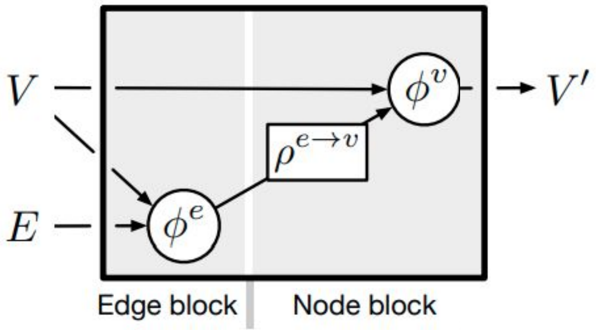
$$\bar{\mathbf{e}}'_i = \rho^{e\rightarrow v}(E'_i) \quad \boxed{= \frac{1}{|\{k : r_k = i\}|} \sum_{k:r_k=i} \mathbf{e}'_k}$$

Must be **permutation invariant**!
mean, sum, max, softmax....

$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i) \quad := f^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i) \quad \boxed{= \mathrm{NN}_v([\bar{\mathbf{e}}'_i, \mathbf{v}_i]) \quad \text{or} \quad = \bar{\mathbf{e}}'_i}$$

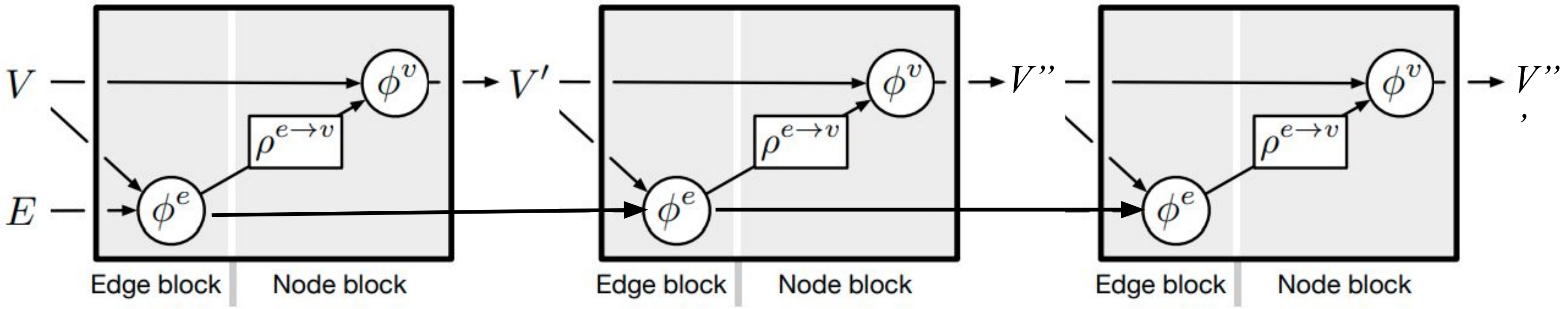
Building an Architecture

One round of graph updates
(a GN Block)



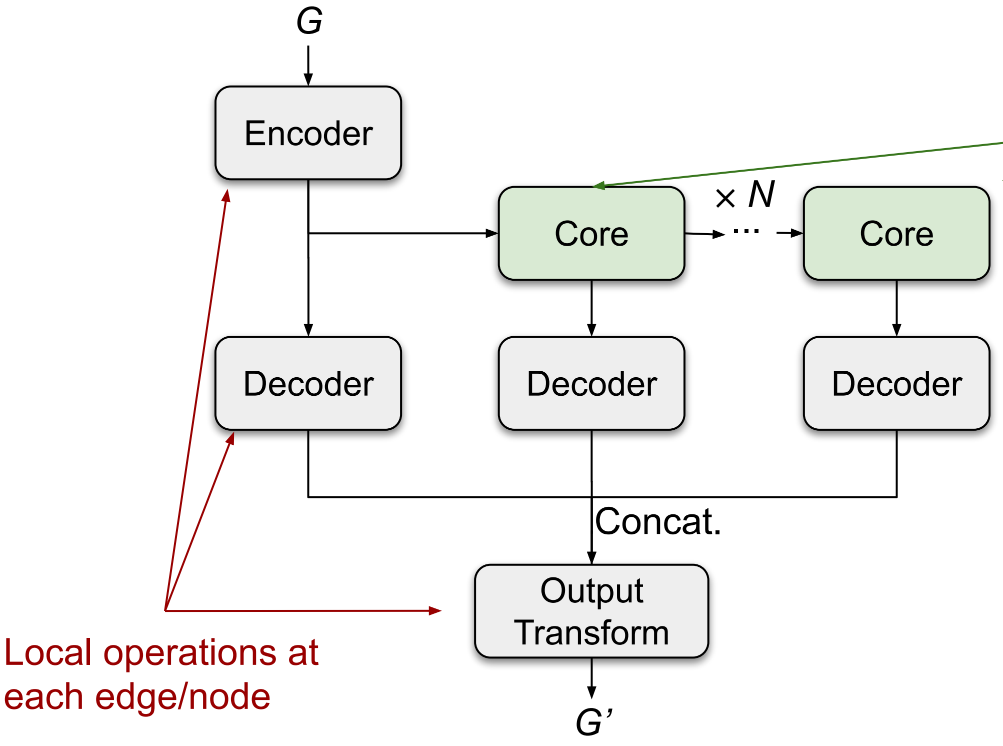
Building an Architecture

Compare to 3 conv layers of width 3, stride 1!

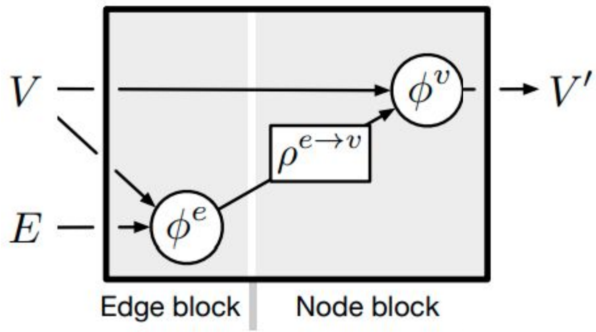


Battaglia '18
Gama '18

Architecture



Graph Network Blocks



Weights are common across GN layers!

$$\mathcal{G}' = f_{\text{out}}\Big(\big[f_{\text{dec}}(f_{\text{enc}}(\mathcal{G})),\ \ f_{\text{dec}}(GN(f_{\text{enc}}(\mathcal{G}))),\ \ f_{\text{dec}}(GN(GN(f_{\text{enc}}(\mathcal{G})))),\ \dots\big]\Big)$$

Learning for Control of Teams

Centralized Training → **Distributed Execution**

Centralized Training

- Observations are centralized
- **Imitation learning**
  - Centralized expert demonstrations
- **Reinforcement learning**
  - One reward signal for the team
  - Centralized value function

Distributed Execution

- Trained policies use only local information
- Synchronization across agents not required

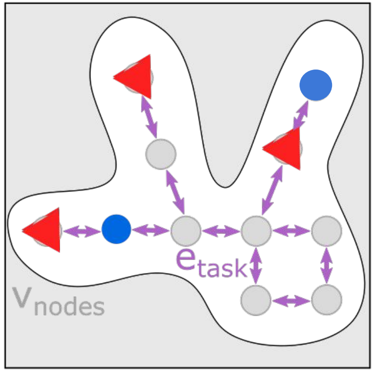
What about distributed learning?

- Composable Learning with Sparse Kernel Representations, Tolstaya et al, 2018 [Slides]

Robot Teams as Graphs

1) Coverage

[Tolstaya et al.
IROS ‘21 (submitted)]



Learning to Coordinate

2) Flocking

[Tolstaya et al.
CoRL '19]

Learning to Control

3) Data Distribution

[Tolstaya et al.
IROS ‘21 (submitted)]

Learning to Communicate

Multi-Robot Coverage and Exploration using Spatial Graph Neural Networks

Ekaterina Tolstaya, James Paulos
Vijay Kumar, Alejandro Ribeiro

Submitted to IROS 2021

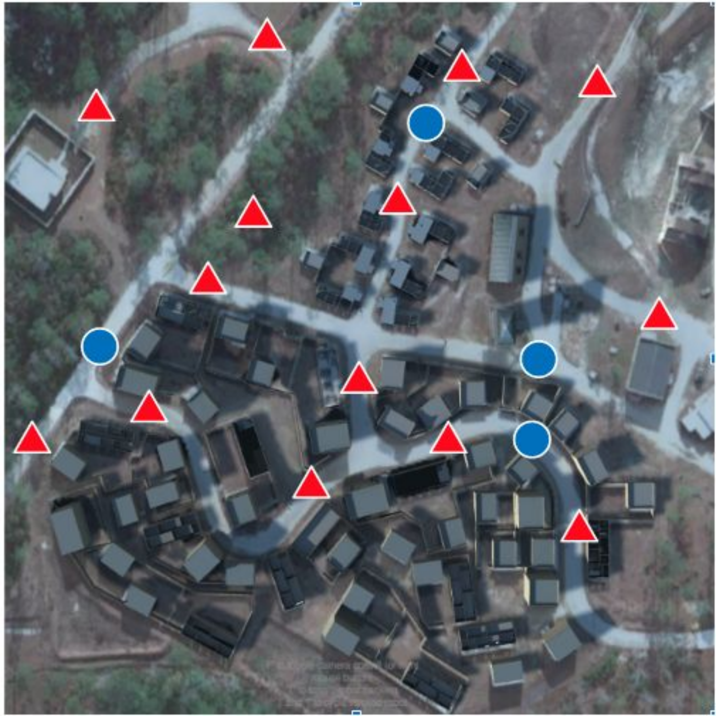
Paper
Task code
Learning code

Multi-Robot Coverage

Navigate a team of robots to visit points of interest in a known map within a time limit

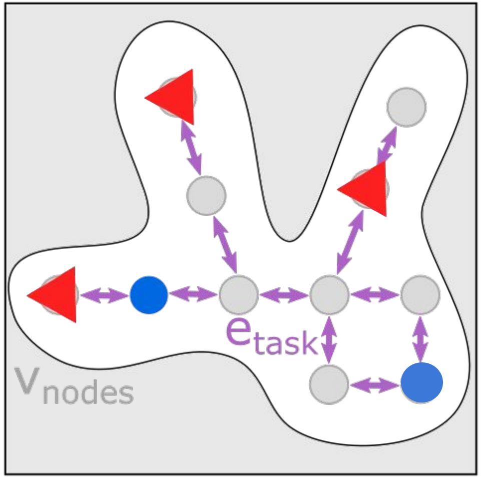
Our approach:

1. Discretize the map to pose coverage as a *Vehicle Routing Problem (VRP)*
2. Generate a dataset of optimal solutions to moderate-size VRPs
   - ➢ Optimizers for VRPs are available, but don't scale to larger maps and teams
3. Train a GNN controller using imitation learning



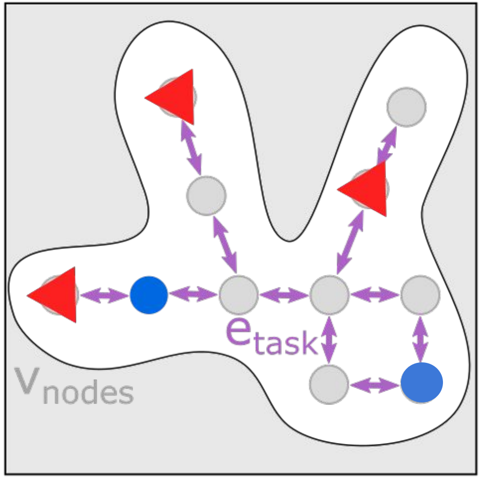
Coverage in a Spatial Graph

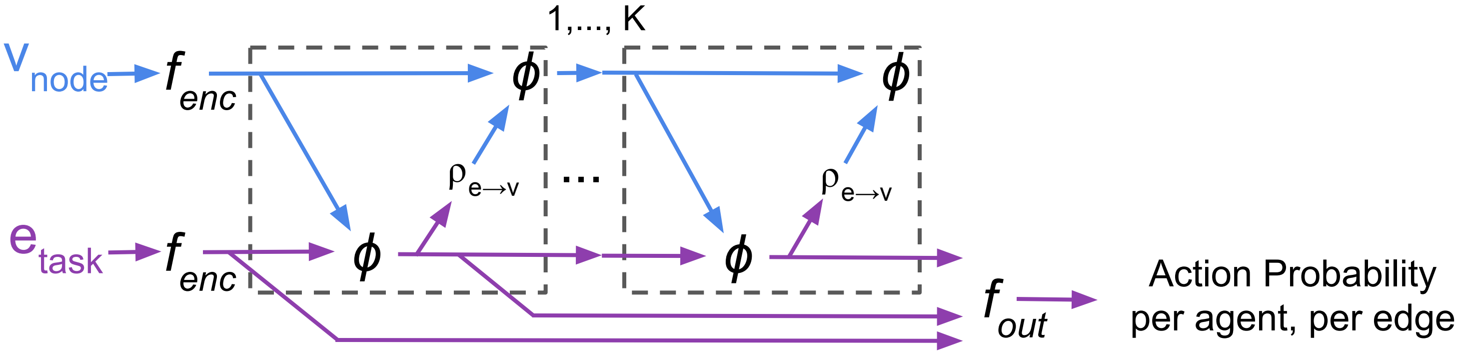
- Capture the structure of the task by *imposing* a graph of waypoint nodes.
- Local aggregations propagate information about points of interest to robots.



Coverage in a Spatial Graph

- Capture the structure of the task by *imposing* a graph of waypoint nodes.
- Local aggregations propagate information about points of interest to robots.

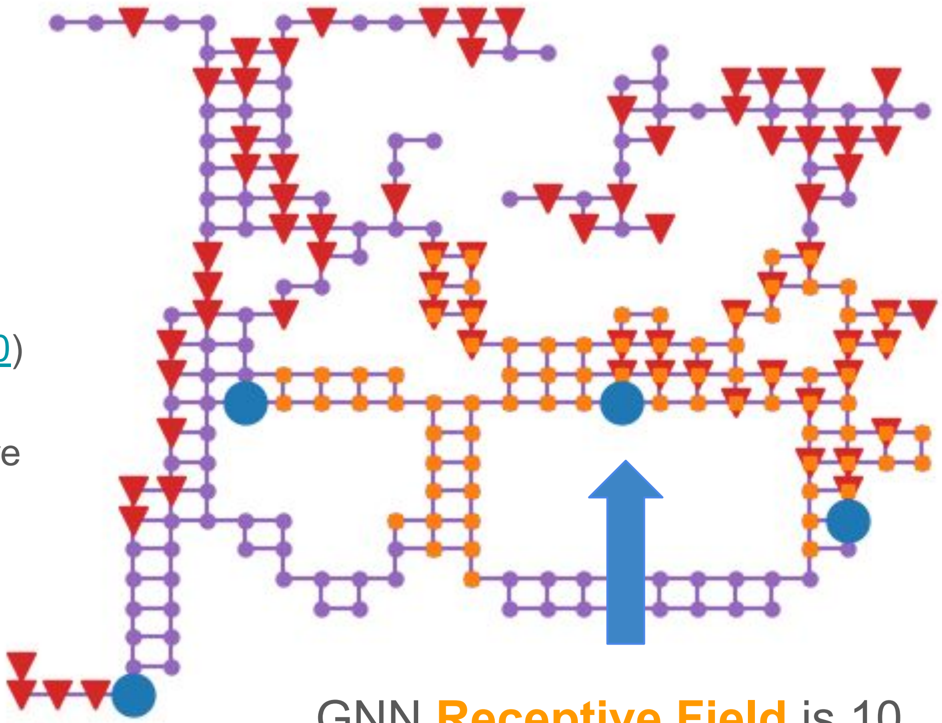




$$\mathcal{G}' = f_{\text{out}}\Big(\big[f_{\text{dec}}(f_{\text{enc}}(\mathcal{G})),\ \ f_{\text{dec}}(GN(f_{\text{enc}}(\mathcal{G}))),\ \ f_{\text{dec}}(GN(GN(f_{\text{enc}}(\mathcal{G})))),\ \ \dots\big]\Big)$$

Locality using a Fixed-Size Receptive Field

- Sparse graph operations use DeepMind's Graph Nets
- Memory ~ O(N+M)
  - For Team Size (N) and Map size (M)
- Compare to
  - Routing using dense GNNs (Sykora '20)
  - Exploration via CNNs (Chen '19)
  - Selection GNNs (Gama '18) that require clustering



● Robots
▼ Points of interest
● Waypoints
■ GNN Receptive Field

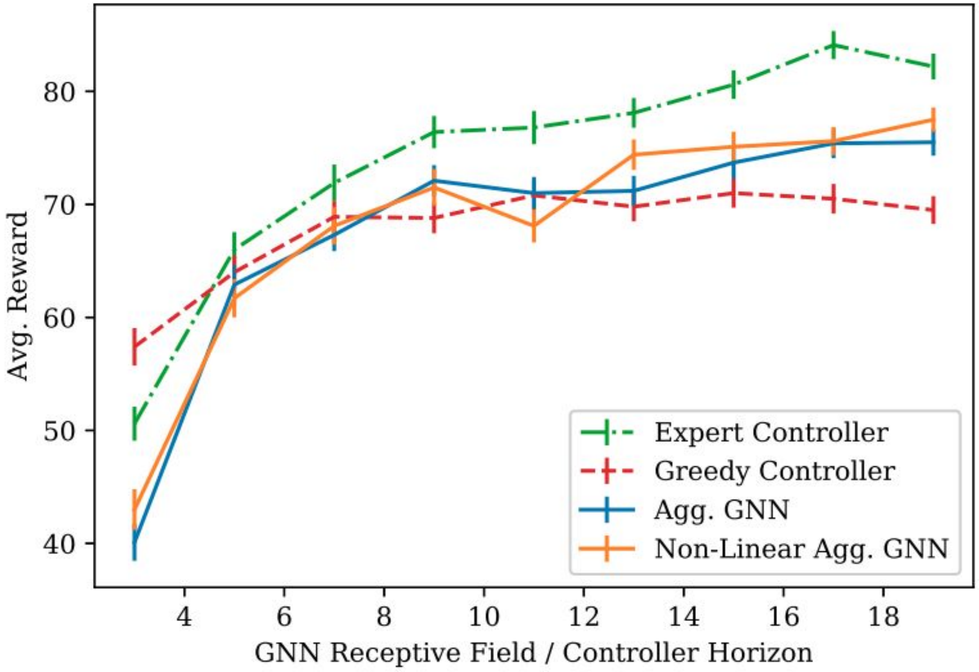
GNN **Receptive Field** is 10 **(K)**

Comparing Receptive Field to Controller Horizon

- GNN inference time ~ O(K)
- VRP solver is not parallelized (Google OR Tools)

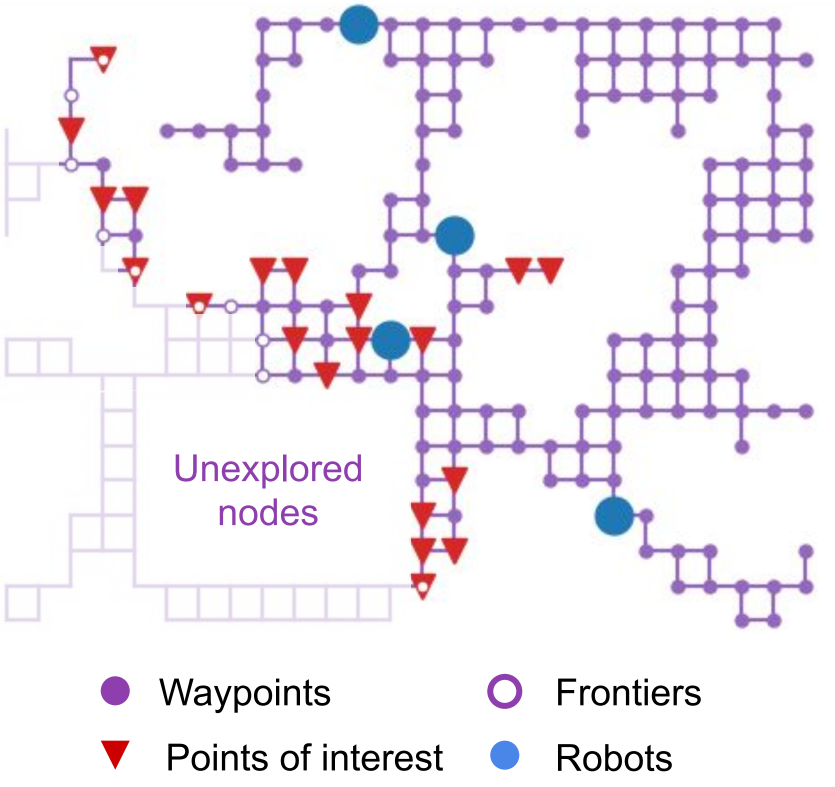
Avg. time per episode (ms)

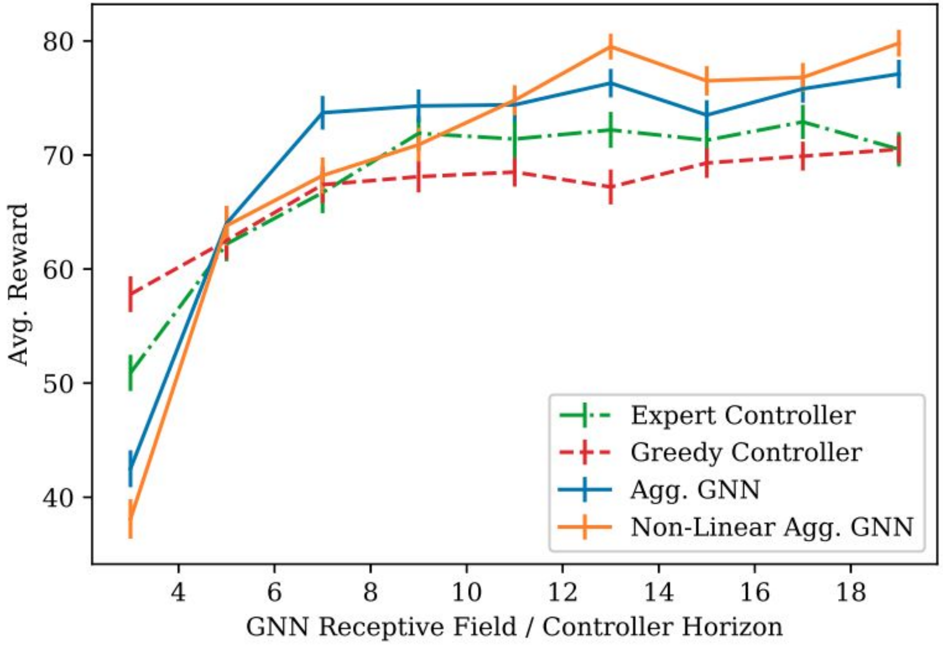
Policy	Receptive Field K=9	K=19	∞
Expert	13 500	23 500	2330
GN-MLP	176	277	-
GN-Linear	133	171	-
Greedy	86.3	142	297



K

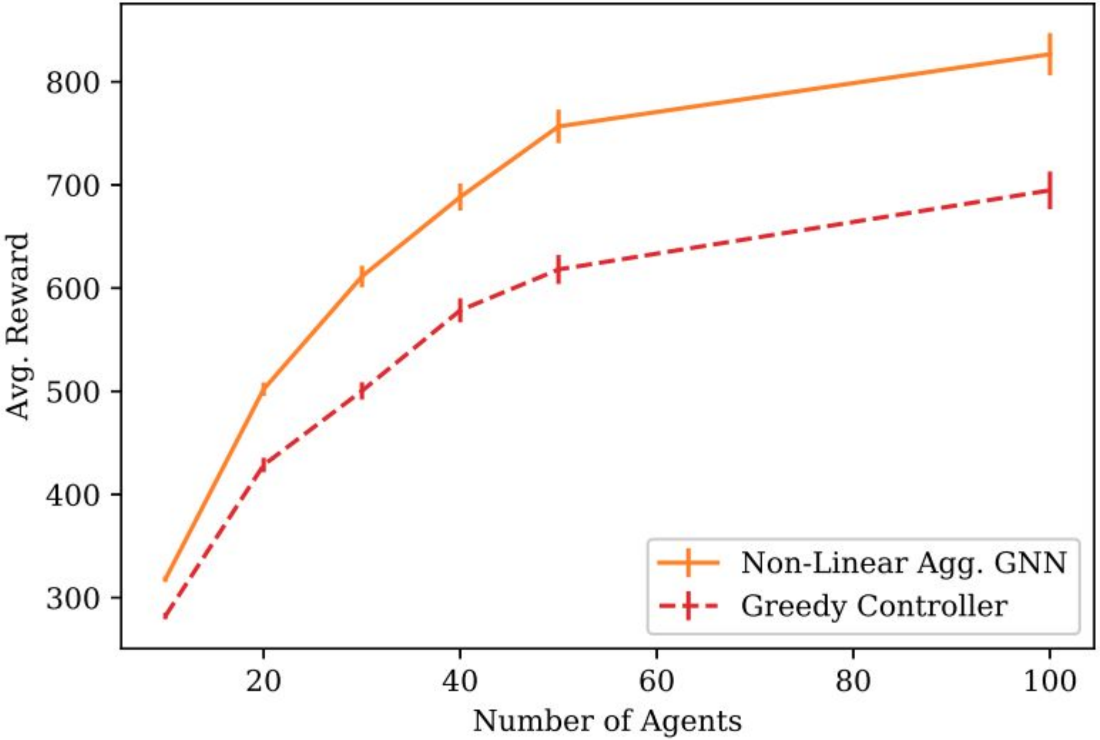
Exploration: Coverage on a growing graph





GNN learns to use **frontier indicators** by imitating an **omniscient** expert.

Generalization to larger teams & maps



Zero-shot generalization to a coverage task with 100 robots and 5659 waypoints.


4 X

Robot Teams as Graphs

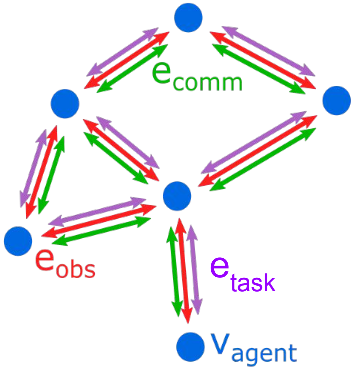
1) Coverage

Tolstaya et al.
IROS '21 (submitted)

Learning to Coordinate

2) Flocking

Tolstaya et al.
CoRL '19



Learning to Control

3) Data Distribution

Tolstaya et al.
IROS '21 (submitted)

Learning to Communicate

Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks

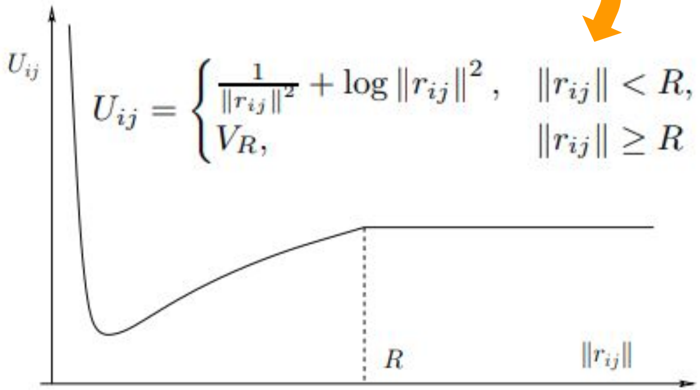
Ekaterina Tolstaya, Fernando Gama, James Paulos,
George Pappas, Vijay Kumar, Alejandro Ribeiro

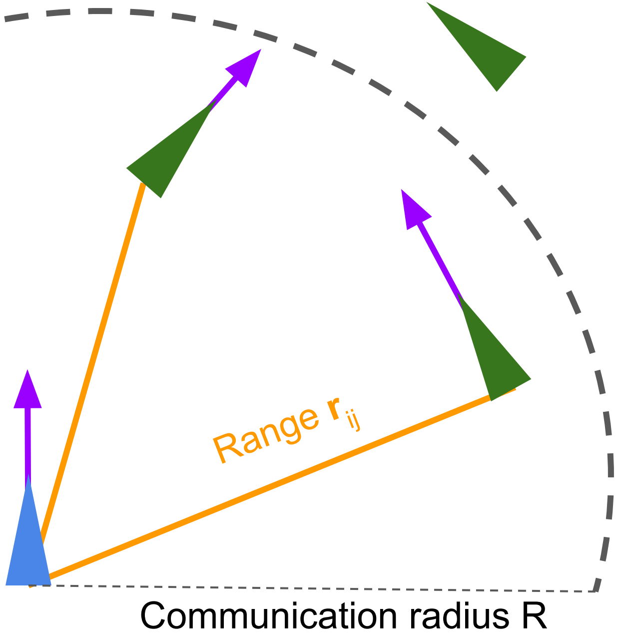
Conference on Robot Learning (CoRL) 2019

Paper
Task Code
Learning Code

Flocking

- Acceleration-controlled robots in 2D
  - Position $\mathbf{r}_i$ and velocity $\mathbf{v}_i$
- Local observations allow agents to:
  - Align **velocities**
  - Maintain regular **spacing**



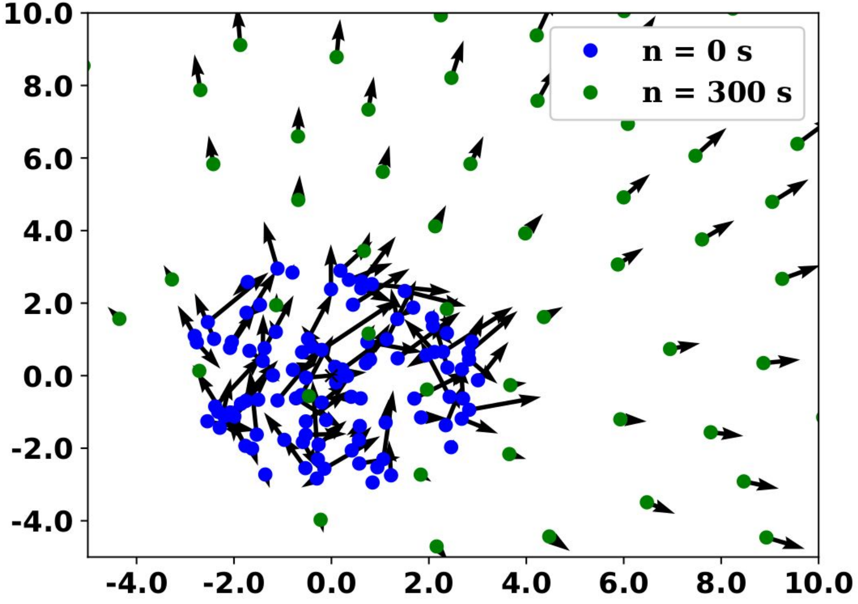


Existing controller:

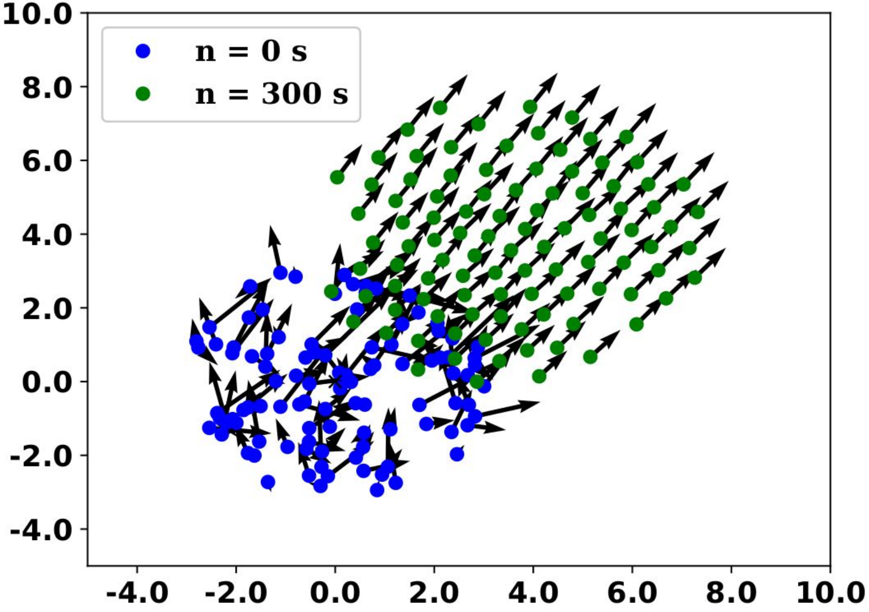
$$u_i = -\sum_{j \in \mathcal{N}_i} (v_i - v_j) - \sum_{j \in \mathcal{N}_i} \nabla_{r_i} U_{ij}$$

"Stable Flocking of Mobile Agents, Part II: Dynamic Topology", Tanner '03

Flocking: What happens when the range is too short?



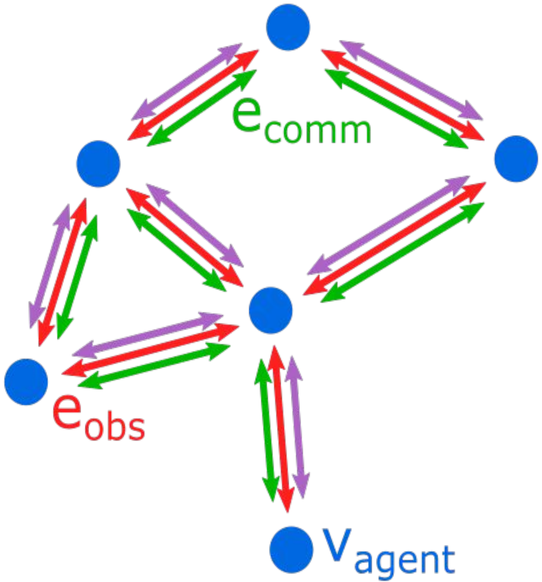
Local controller allows agents to scatter (Tanner, 2003)



GNN controller maintains regular spacing and aligned velocities

Flocking with Delayed Communication

- Observations of relative measurements to neighbors.
- Reward based on variance in agent velocities
- Delayed communication only available with immediate neighbors.



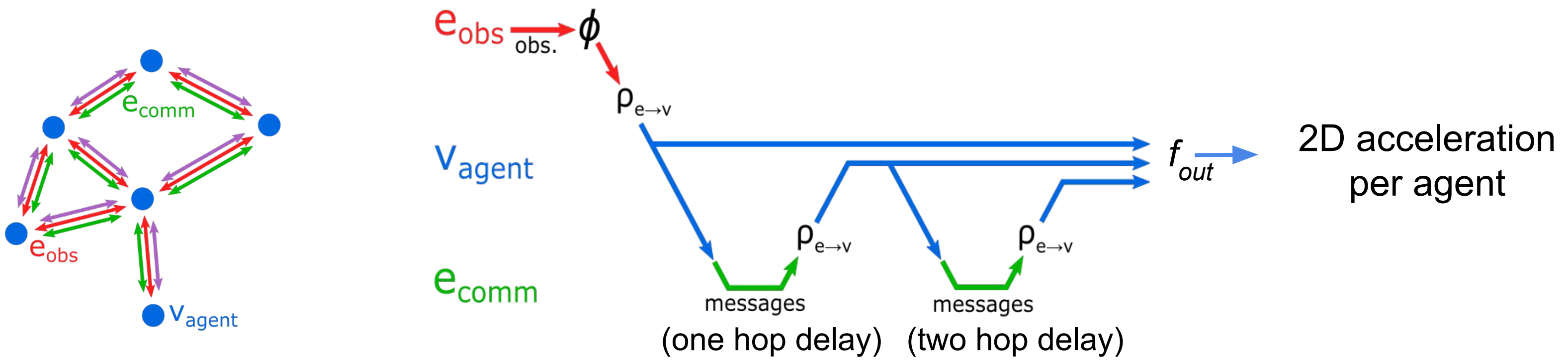
Delayed Aggregation Graph Neural Network

- GNN uses delayed multi-hop aggregations to imitate a centralized expert
- Novelty of our approach:
  - Formalizing inter-agent communication as a multi-hop GNN
  - Modeling communication delays within the GNN
- Implemented using PyTorch

Process V_t using connectivity at time s, E_s

$$GN_s(\mathcal{G}_t) := GN(\{E_s, V_t\})$$

$$G'_t = f_{\text{out}}\left(\left[GN_{\text{obs}}(\mathcal{G}_t),\ \ GN_{t-1}(GN_{\text{obs}}(\mathcal{G}_{t-1})),\ \ GN_{t-1}(GN_{t-2}(GN_{\text{obs}}(\mathcal{G}_{t-2}))),\ \dots\ \right]\right)$$

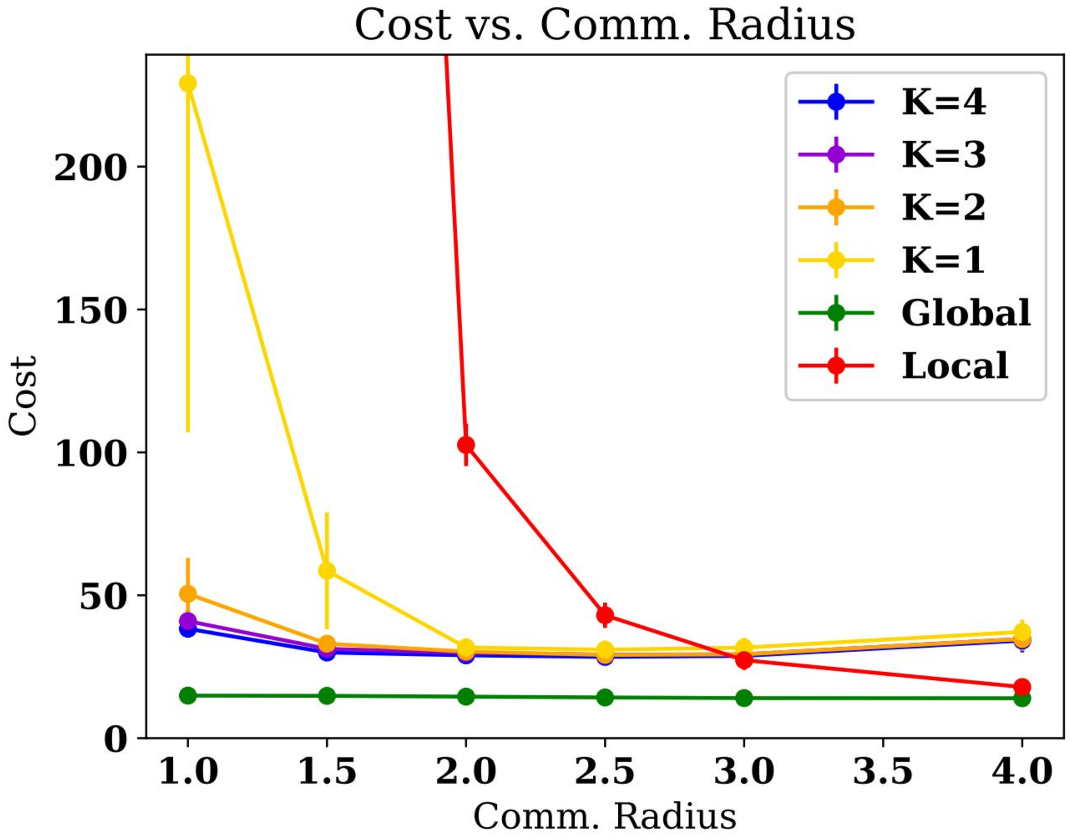


Aggregation helps when **communication is limited**

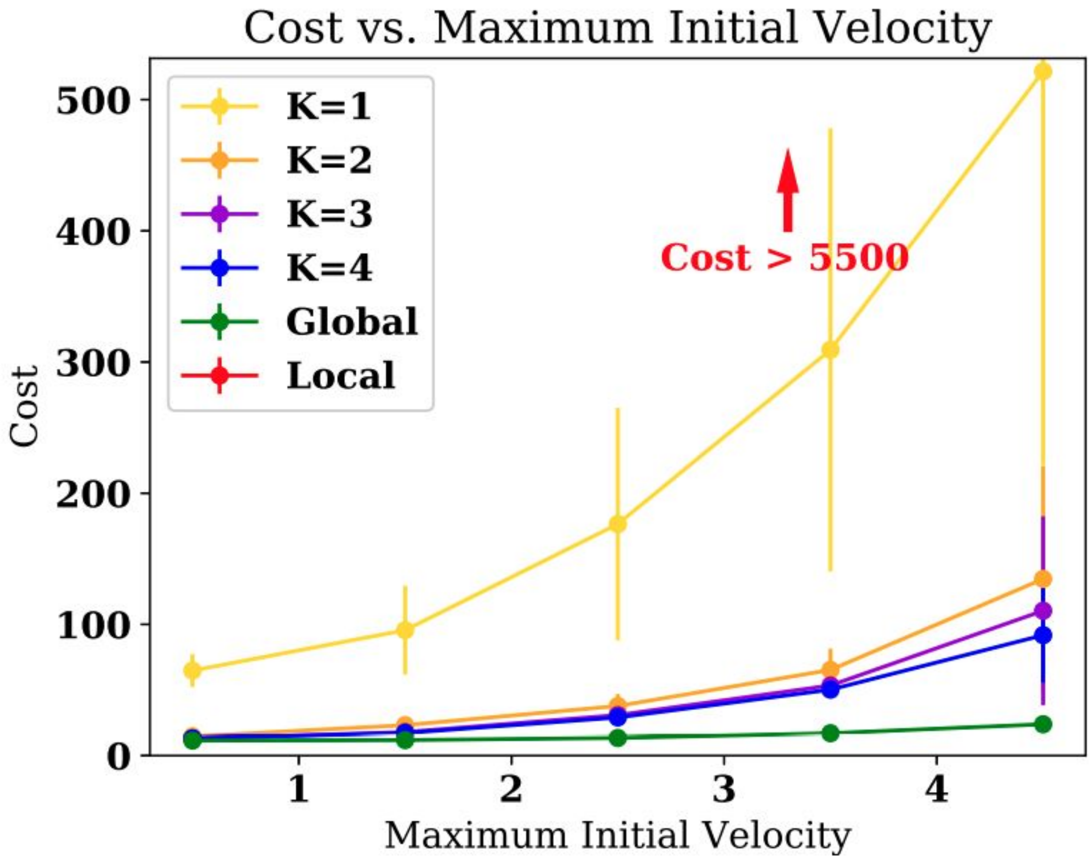
Local Sensors

3-hop Aggregation

Centralized



Aggregation helps when **agents move faster**

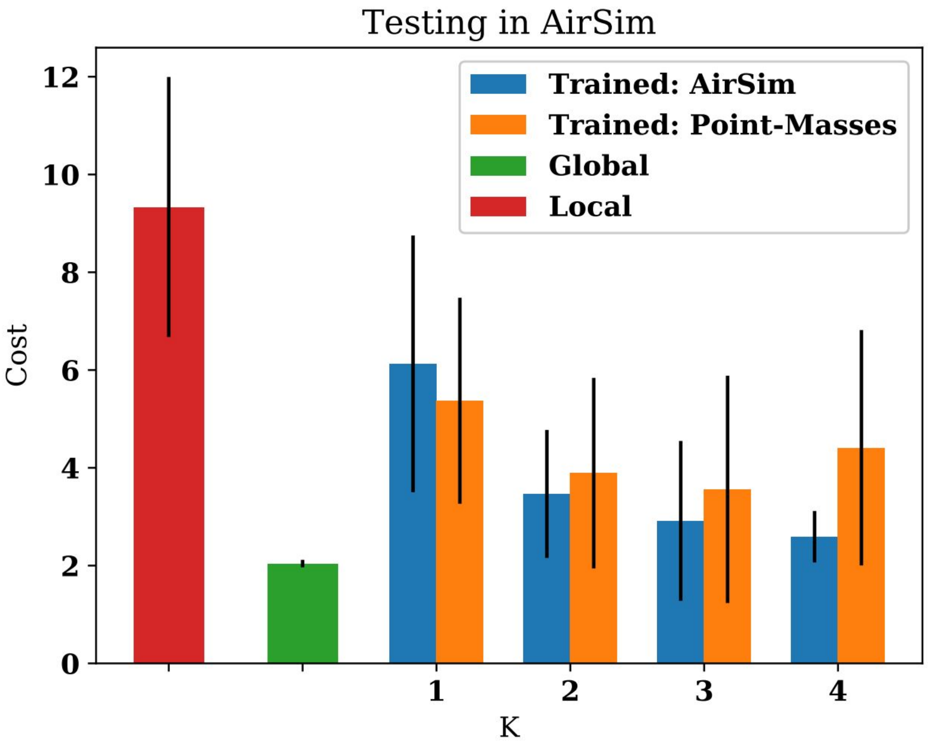


Local Sensors

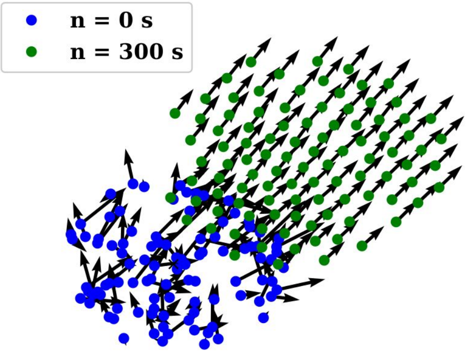
3-hop Aggregation

Centralized

Aggregation GNN with **delays** and **complex dynamics**

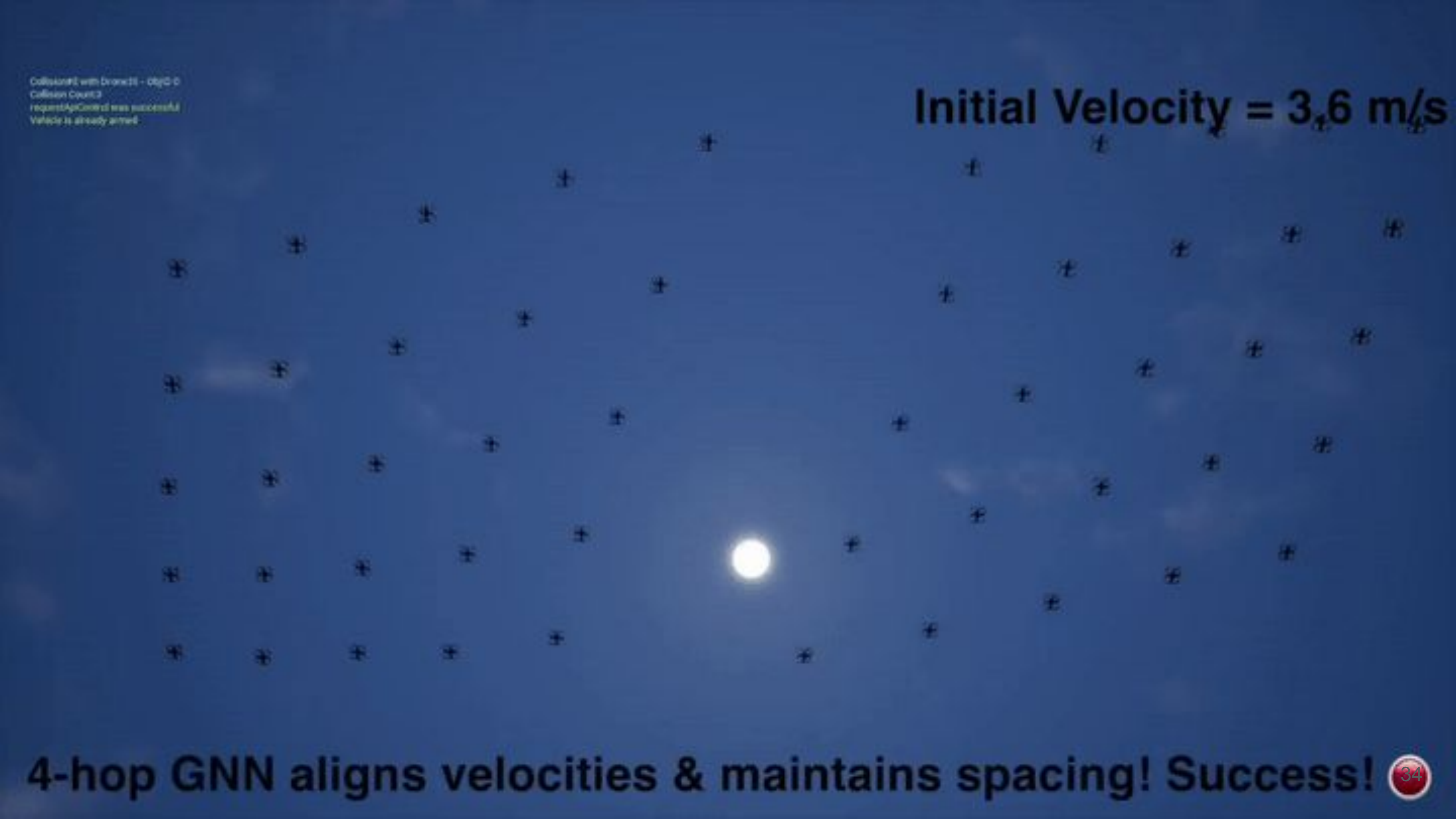


Microsoft AirSim



Point Masses


Initial Velocity = 3.6 m/s


requestApiControl was successful
Vehicle is already armed
Initial Velocity = 3,6 m/s
4-hop GNN aligns velocities & maintains spacing! Success!

Robot Teams as Graphs

1) Coverage

[Tolstaya et al.
IROS ‘21 (submitted)]

Learning to Coordinate

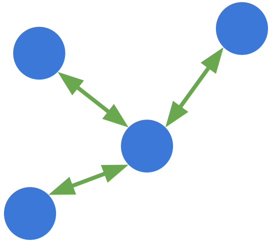
2) Flocking

[Tolstaya et al.
CoRL '19]

Learning to Control

3) Data Distribution

[Tolstaya et al.
IROS ‘21 (submitted)]



Learning to Communicate

Learning Connectivity in Distributed Robot Teams

Ekaterina Tolstaya*, Landon Butler*, Daniel Mox, James Paulos, Vijay Kumar, Alejandro Ribeiro

Submitted to IROS 2021

* Equal contribution

[Paper](#)
[Code](#)

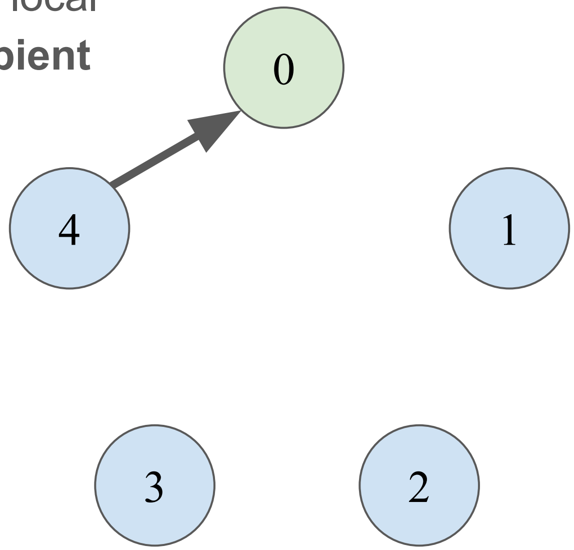
Data Distribution in a Mobile Robot Team

- Infrastructure to provide each robot with up-to-date information about team members, their network, and the mission

- Popular approaches for route discovery in dynamic mesh networks:
  - Flooding (Williams '02)
  - Heuristics to minimize **Age of Information,** network overhead (Tseng '02)

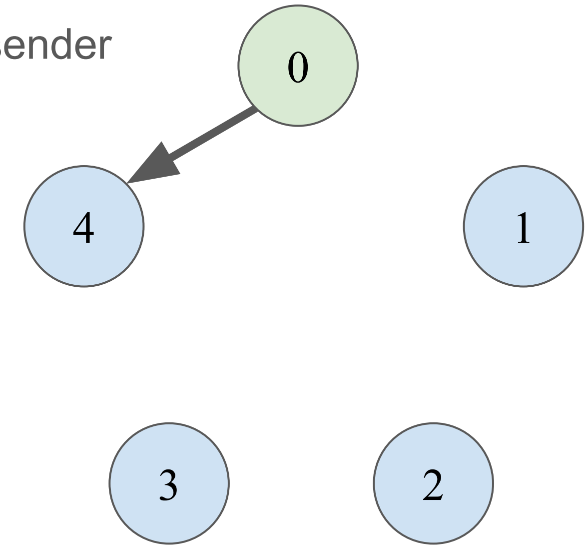
2-way Protocol for Data Distribution

1. Each agent evaluates its local policy to select **one recipient** or not to transmit.



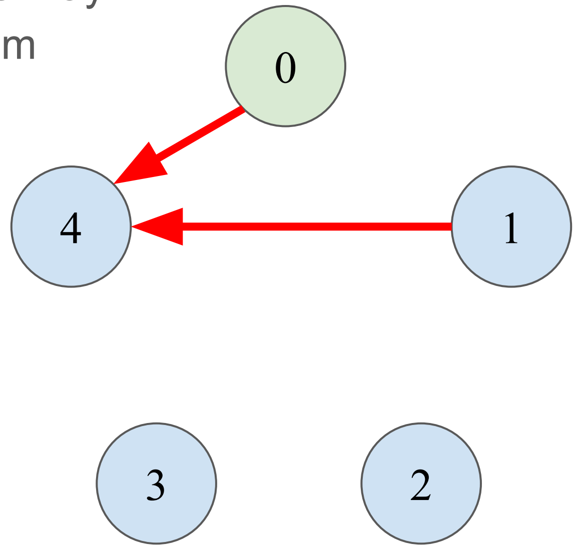
2-way Protocol for Data Distribution

2. Each recipient sends a **response** back to the sender (if successful).



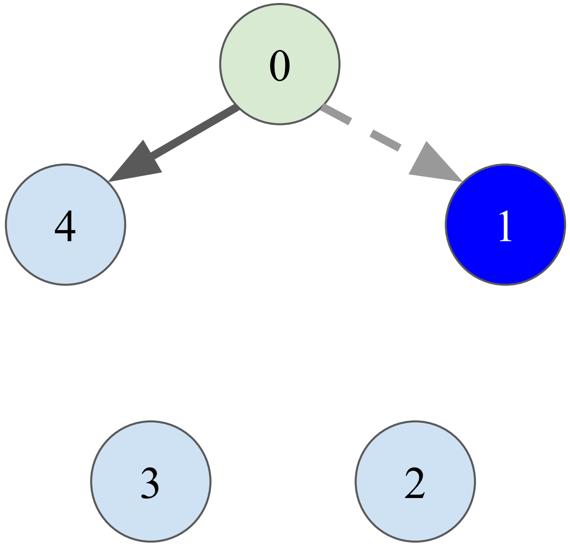
2-way Protocol for Data Distribution

A transmission or response may fail due to **interference** from others.

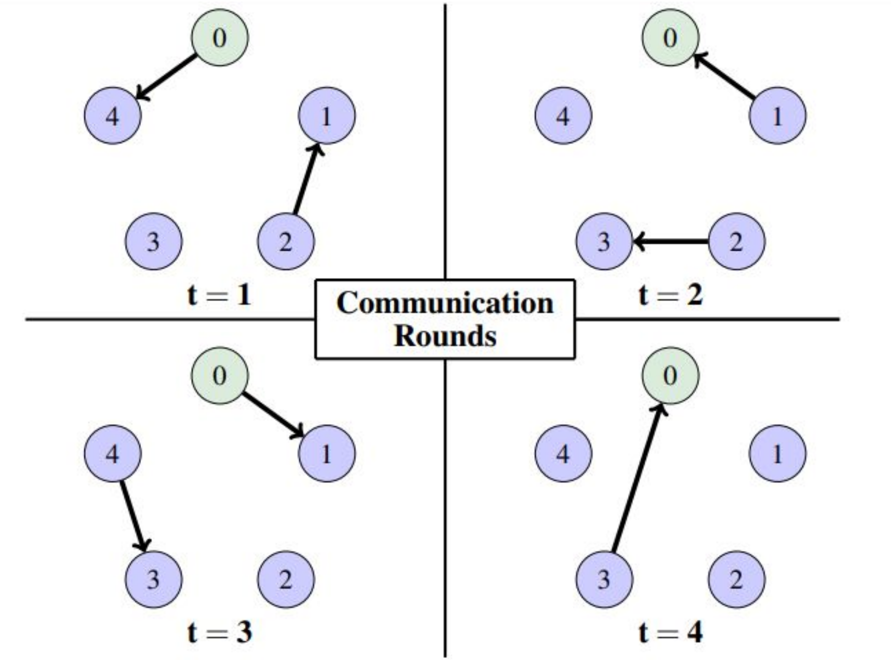


2-way Protocol for Data Distribution

Teammates can **eavesdrop,** or use information from messages directed to others.



Data Distribution in a Mesh Network



Learn a communication policy

$$p(k \in S_t^i) = \pi\left(\left\{T_t^{i,j}, M_t^{i,j}, P_t^{i,j}, L_t^{i,j}\right\}_{j \in \mathcal{A}}\right)$$

To minimize the **Age of Information**

$$\min_{\pi} \; \mathbb{E}_{t \in \mathcal{T},\; i \in \mathcal{A},\; \mathbf{x}_t^i \in \mathcal{X}}\left[t - T_t^{i,j}\right]$$

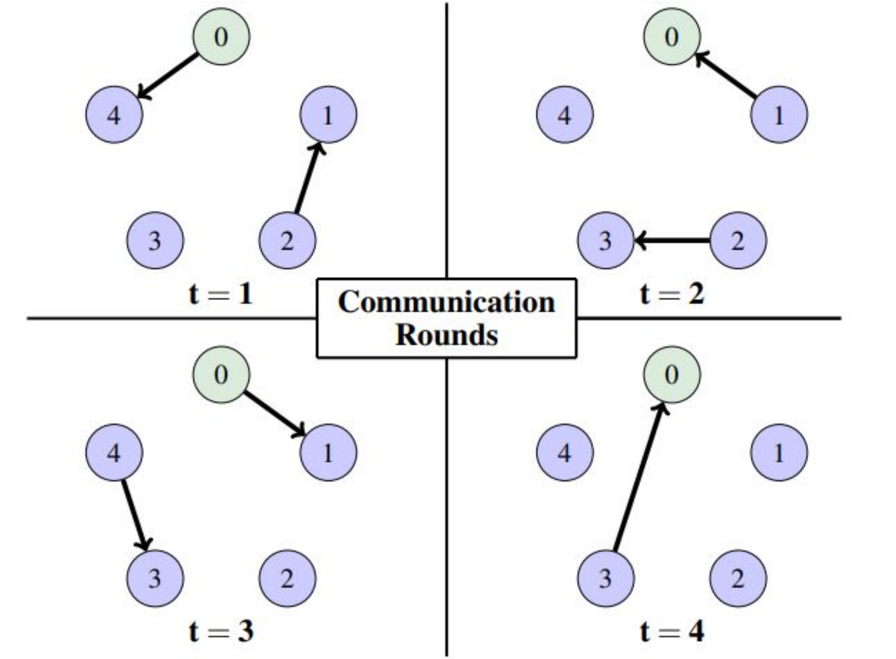
Subject to wireless interference

Packet drops determined by the
Signal to Interference + Noise Ratio

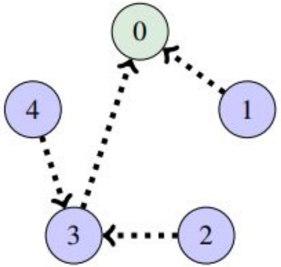
$$\Gamma_t^{i,j} = \frac{\rho_t^i \cdot g_t^{i,j}}{\sigma + \sum_{k \in \mathcal{A} \setminus i} \rho_t^k \cdot g_t^{k,j}}$$

Maintaining Local Data Structures

If an agent receives a message, it updates its local data structure with new data:



Agent 0's Local Data Structure *				
ID	**TS**	**State**	**Parent**	**LC**
0	4	$M_4^{0,0}$		
1	2	$M_4^{0,1}$	0	3
2	2	$M_4^{0,2}$	3	
3	4	$M_4^{0,3}$	0	
4	3	$M_4^{0,4}$	3	1



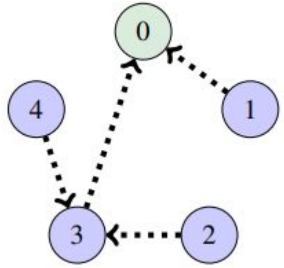
Tree Representation of Agent 0's Local Data Structure at t = 4

$$T_t^{i,k} < T_t^{j,k} \implies (T_t^{i,k} = T_t^{j,k}) \wedge (M_t^{i,k} = M_t^{j,k}) \wedge (P_t^{i,k} = P_t^{j,k}) \quad \forall i,k \in \mathcal{A}, j \in R_t^i$$

Connectivity as a Reinforcement Learning Problem

- Observation
  - Each agent has access only to its local data structure
  - For a team of N agents, we have N graphs with N nodes each
- Action
  - Each agent chooses 1 next link, or to not communicate
- Reward
  - -1 × Age of Information, average over all agents, timesteps

Agent 0's Local Data Structure *				
ID	**TS**	**State**	**Parent**	**LC**
0	4	$M_4^{0,0}$		
1	2	$M_4^{0,1}$	0	3
2	2	$M_4^{0,2}$	3	
3	4	$M_4^{0,3}$	0	
4	3	$M_4^{0,4}$	3	1

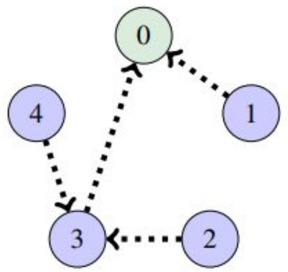


Agent 0's Local Data Structure at time t=*4*

Connectivity as a Reinforcement Learning Problem

- Observation
  - Each agent has access only to its local data structure
  - For a team of N agents, we have N graphs with N nodes each
- Action
  - Each agent chooses 1 next link, or to not communicate
- Reward
  - -1 × Age of Information, average over all agents, timesteps
- Centralized training via Proximal policy optimization (Schulman '17)
- Inference can be decentralized since the policy uses only local data

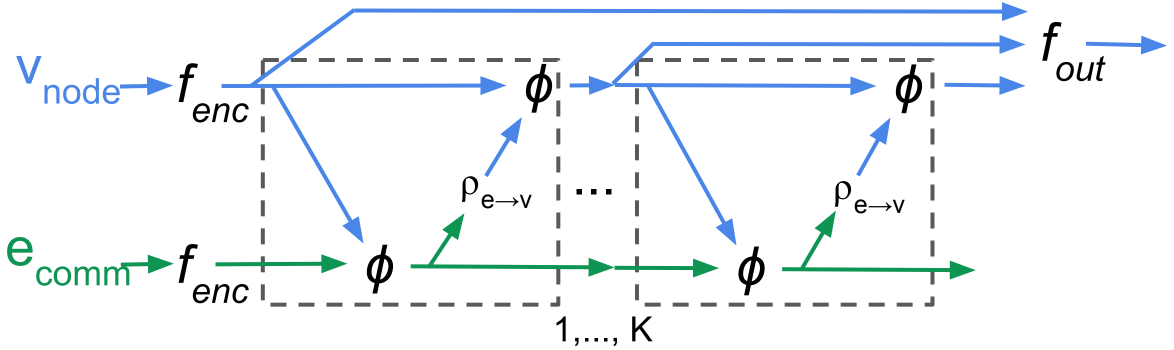
Agent 0's Local Data Structure *				
ID	**TS**	**State**	**Parent**	**LC**
0	4	$M_4^{0,0}$		
1	2	$M_4^{0,1}$	0	3
2	2	$M_4^{0,2}$	3	
3	4	$M_4^{0,3}$	0	
4	3	$M_4^{0,4}$	3	1



Agent 0's Local Data Structure
at time t=*4*

Graph Neural Network Architecture

- Value and policy models are parametrized as GNNs
- Implemented using DeepMind's Graph Nets in TensorFlow



Policy:
N Action Prob. per agent
Value Function:
Sum of 1 value per node

$$\mathcal{G}' = f_{\text{out}}\Big(\big[f_{\text{dec}}(f_{\text{enc}}(\mathcal{G})),\;\; f_{\text{dec}}(GN(f_{\text{enc}}(\mathcal{G}))),\;\; f_{\text{dec}}(GN(GN(f_{\text{enc}}(\mathcal{G})))),\;\; \ldots\,\big]\Big)$$

GN, f_{dec}, f_{enc} 3 layer MLP with 64 hidden units

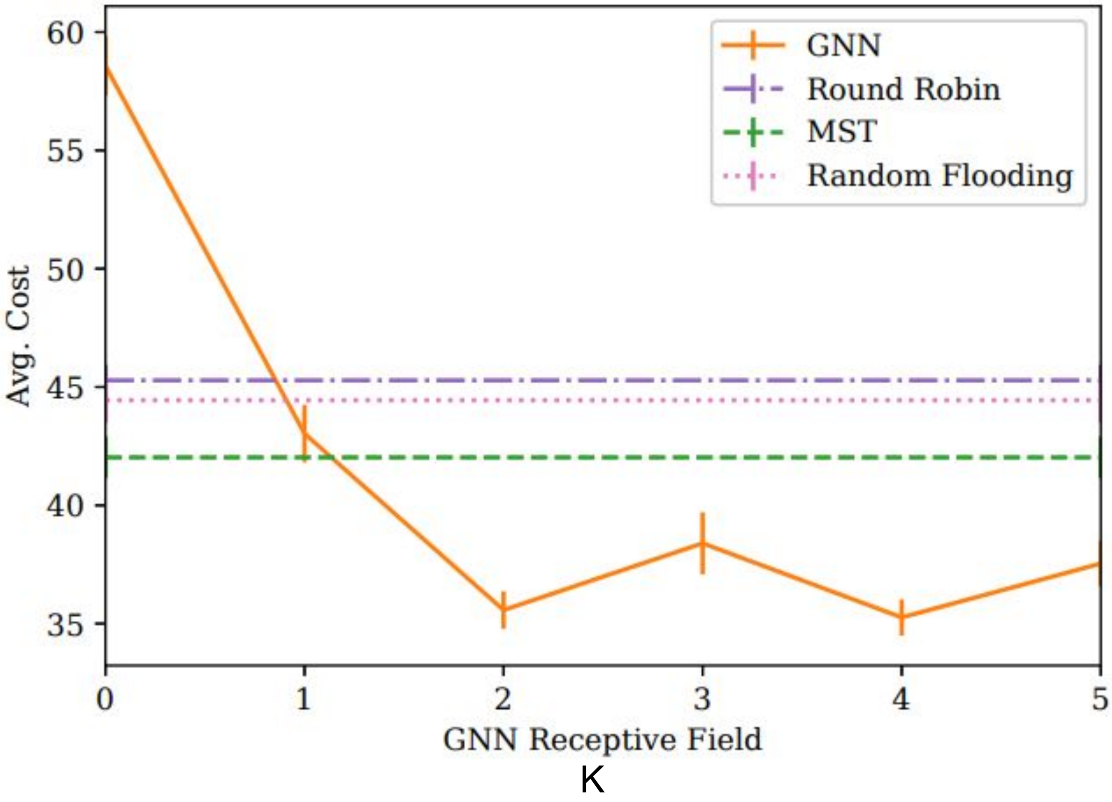
GNN Receptive Field

Stationary agents

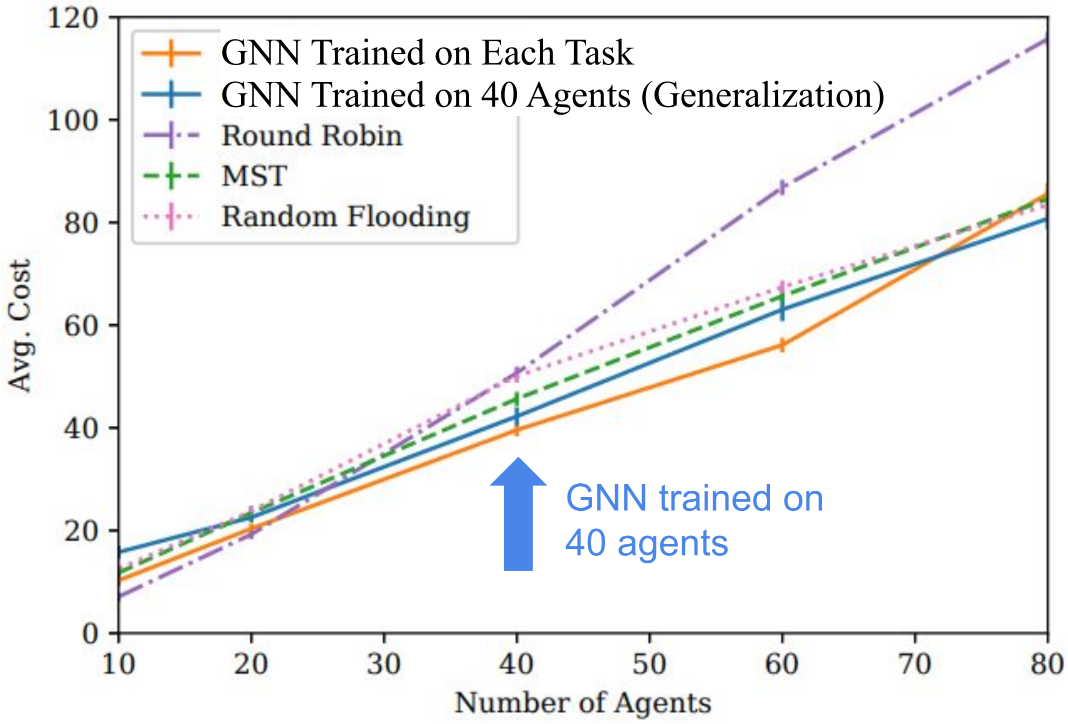
Existing approaches:

- Round Robin, [Miao 2016]
- Minimum Spanning Tree (MST), [Tseng ‘02]
- Random flooding, [Williams ‘02]

Inference time ~ O(K), where K is the receptive field of the GNN



Generalization to Large Mobile Teams



Memory for centralized training scales with $O(N^2)$, where N is number of agents.

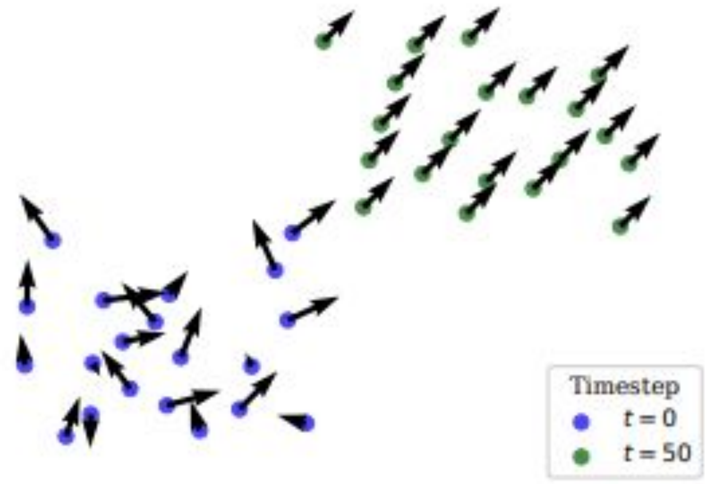
Flocking (Revisited)

We implement the decentralized controller with delayed information provided by the data distribution algorithm:

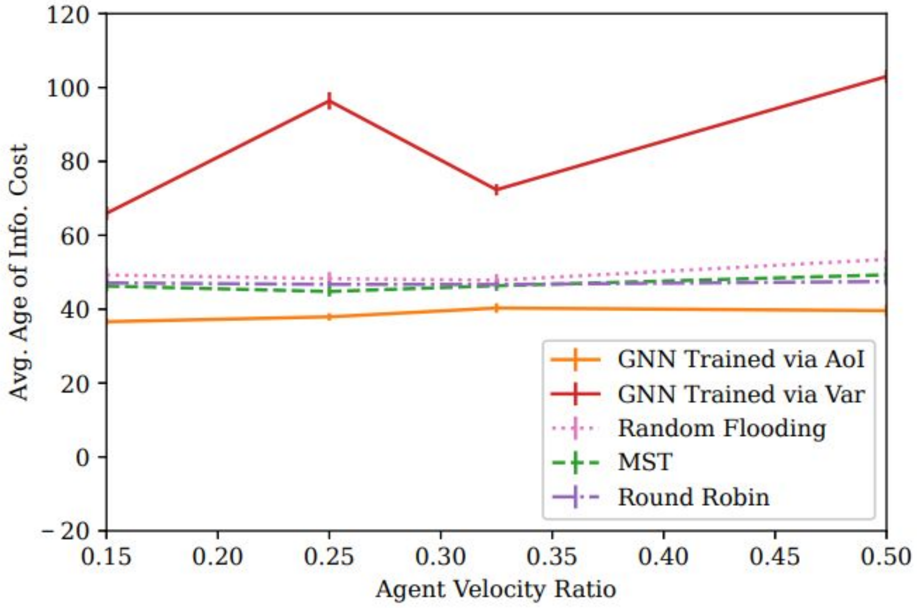
$$u_i = -\sum_{j \in M_i} (v_i - v_j)$$

Which reward function is more informative for training the communication policy?

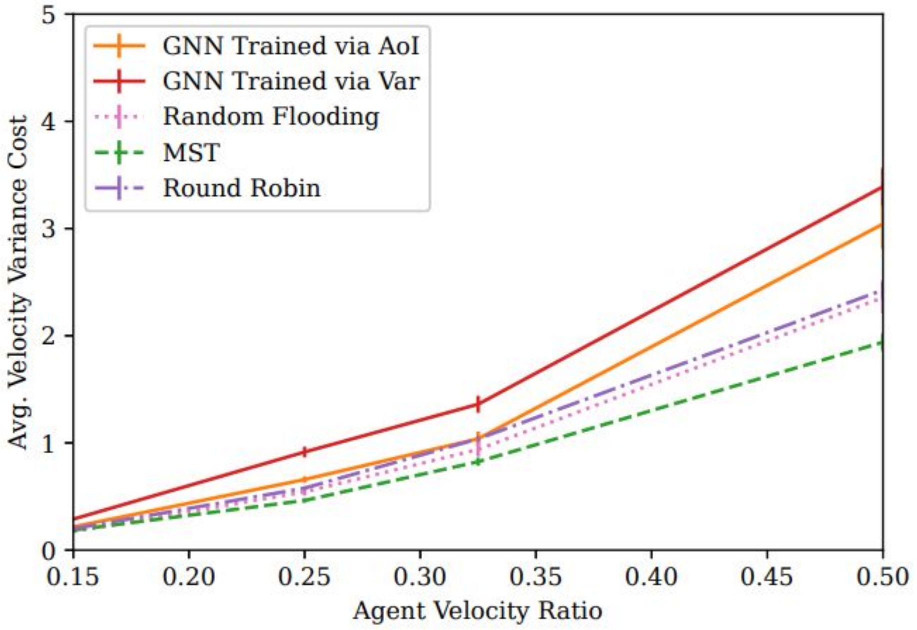
- Age of Information?
- Variance in Velocities?



Age of Information Reward



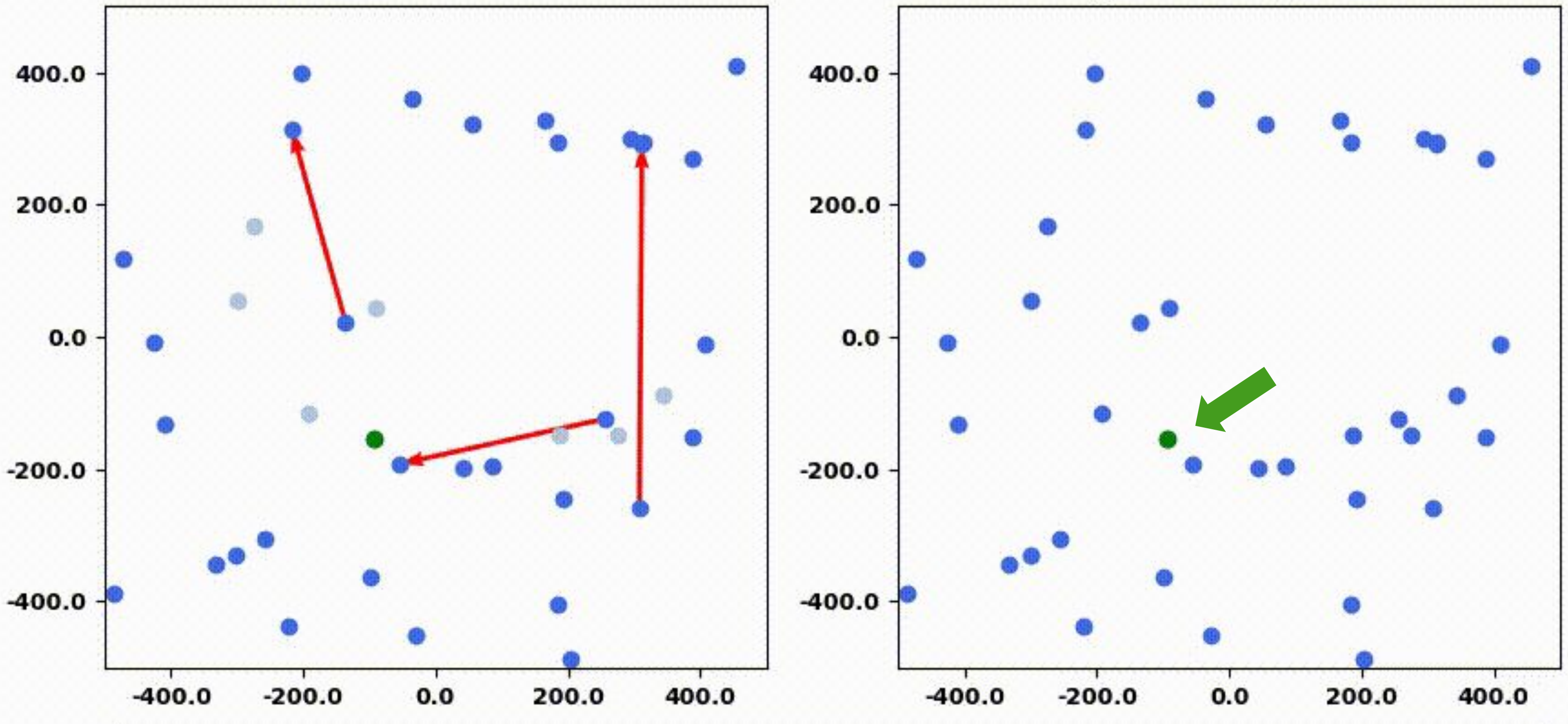
(a) Testing Age of Information Cost



(b) Testing Velocity Variance Cost

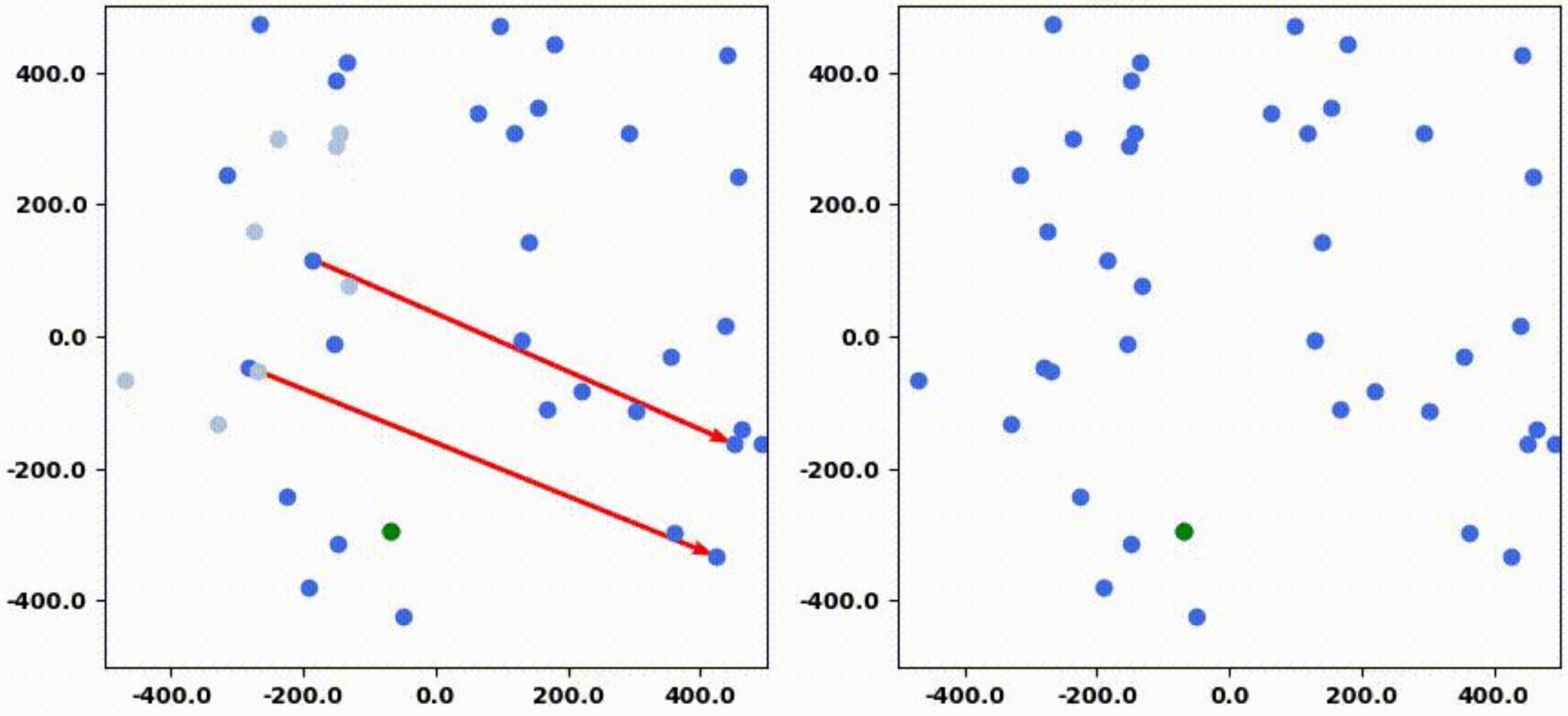
Network Interference

Agent 0's Buffer Tree



Mean AoI: 0.97 | Mean Hops: 0.00 | Mean TX Dist: 574.74 | Comm %: 0.0 | Connected Network: False

Network Interference | Agent 0's Buffer Tree



Mean AoI: 0.97 | Mean Hops: 0.01 | Mean TX Dist: 547.67 | Comm %: 0.0 | Connected Network: False

Graph Neural Networks for Scalable Robot Teams

- Graph Neural Networks enable scalable controllers for coordination, control and communication.
- Centralized training and distributed deployment is an effective tool for scalability to large teams.

- Continuing challenges in multi-agent systems
  - Hardware and real-time inference for physical deployments
  - Human-centered systems
  - Non-cooperative or adversarial tasks

Penn Engineering
GRASP LABORATORY
General Robotics, Automation, Sensing & Perception Lab

Thank you!