Neural Tree Expansion for Prioritized Multi-Agent Path Deconfliction

Khushal Brahmbhatt
Andrew Festa
Kavinda Senewiratne

Kavi - outline of slides

Problem:
- picture of wildland fire
- Consequences of wildfires - lives lost, assets destroyed, getting worse every year (global warming)
- why monitoring fronts is important (high risk decision making - minimal error, latest info needed)
- Current methods and why they are insufficient (satellite - low temporal res, terrestrial - no communication)

Related Work (method and issues with them)
- Heuristic based methods - prioritization heuristics (path known before hand, priority assigned on capability of robot and not the task)
- Conflict based search - fail to consider priority, centralized
- Mixed integer solutions

Solution
- NTE - imitation learning
- Pictorial algo
- Explain each component
- Briefly touch on what we are changing in NTE and why - path confliction

Contributions
- Reward function

Try get it down to 3:30

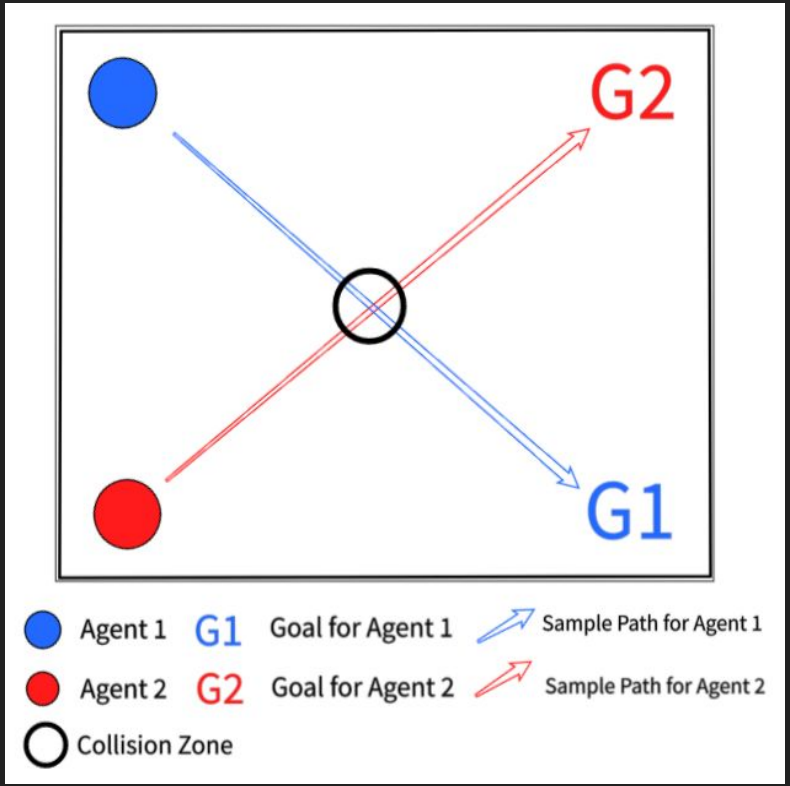
Background - Wildland Fires

- Countless civilian and firefighter **lives lost**.
- $10Bn in assets damaged in the US.
- Destruction of natural habitats.

Problem - Monitoring Firefronts with Multiple Robots

- Fire Fronts help predict spread of fire.
- Decision making under extreme conditions.
- **Paths could conflict.**

Path Confliction



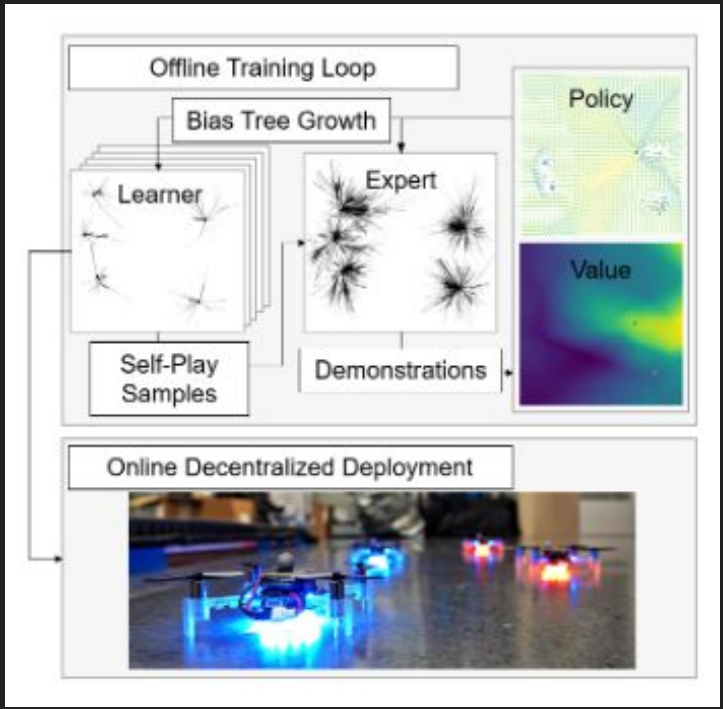
Our Focus

1. Decentralization
2. Focus on priority

Related Work (Multi-Agent Path Deconfliction)

- Heuristic based methods
  - Path known beforehand
  - Priority assigned on capability of robot and not the task
- Conflict based search
  - Fail to consider priority
  - Centralized Mixed integer solutions

Method



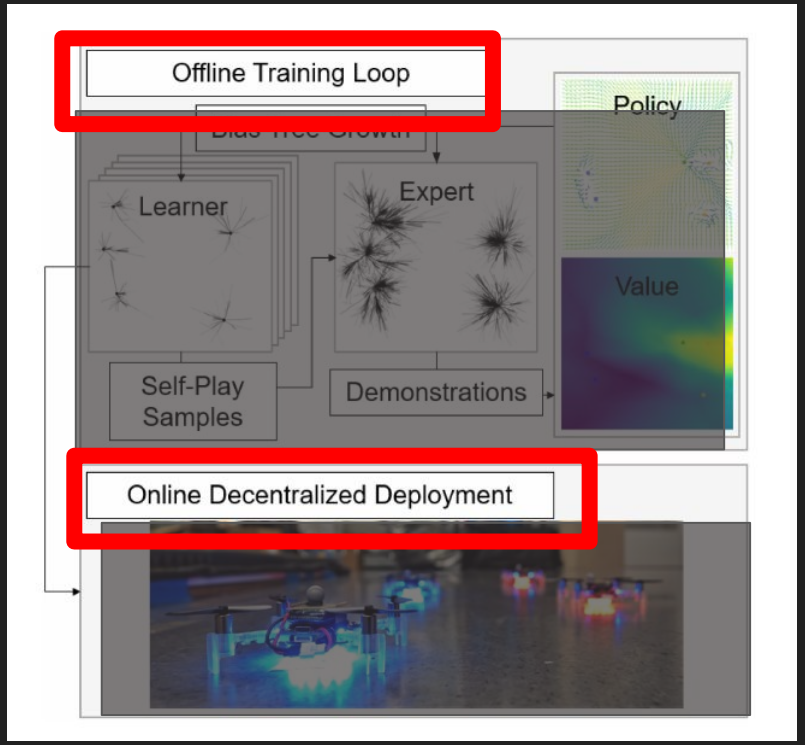
Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Benjamin Riviere, Wolfgang Hoenig, Matthew Anderson, and Soon-Jo Chung

(Imitation Learning)

Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

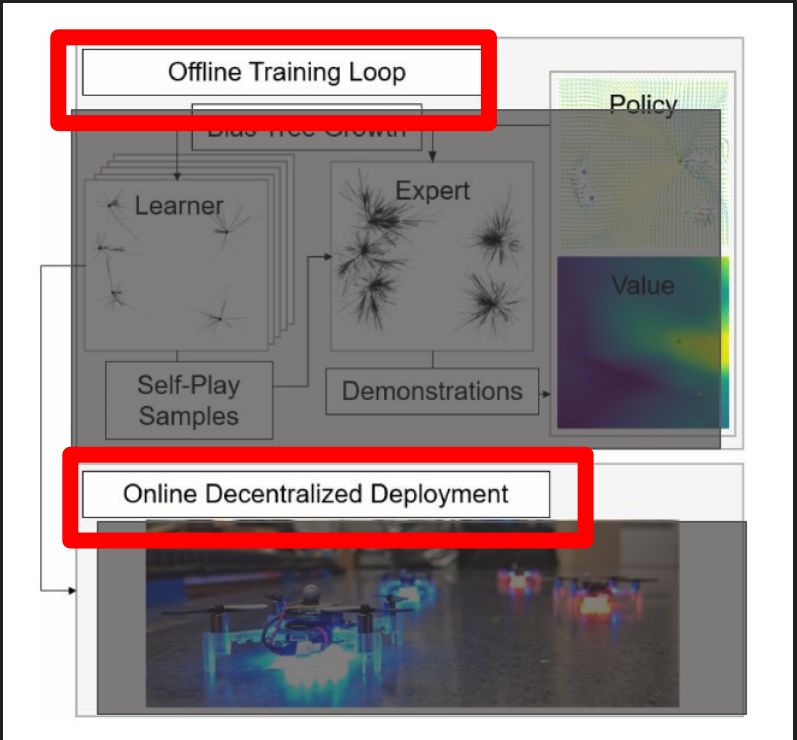
Method



Policies for Robots

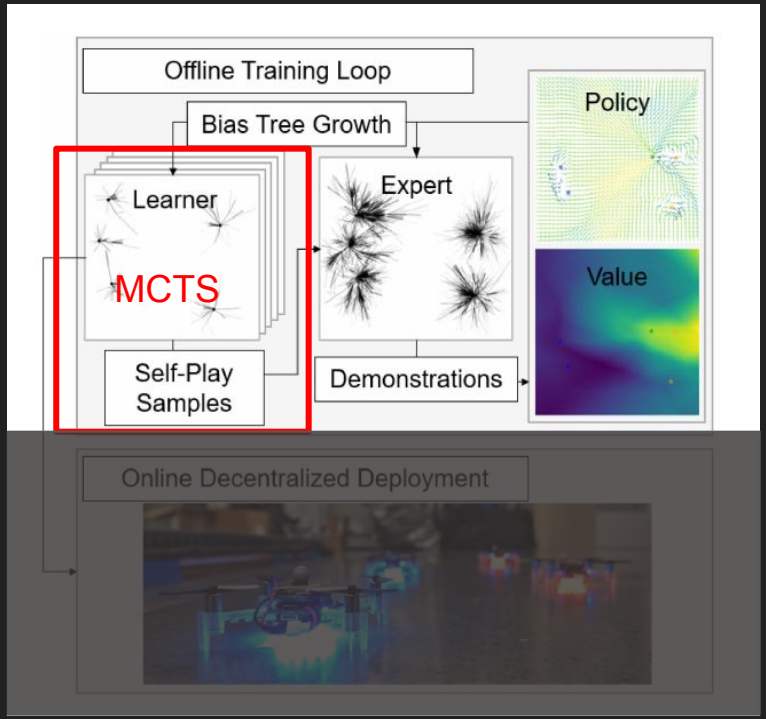
Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Method



Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Method



Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

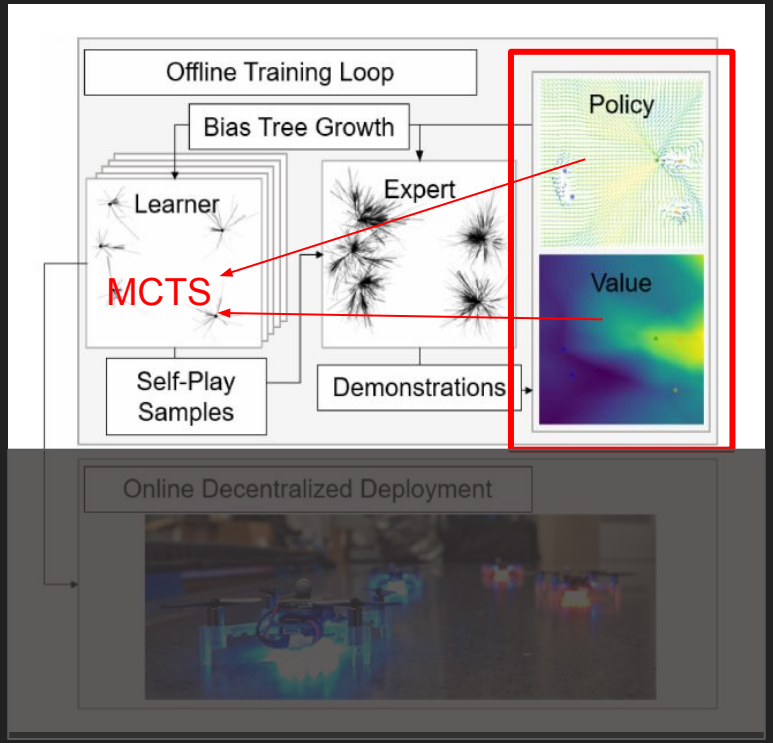
Method

Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Method

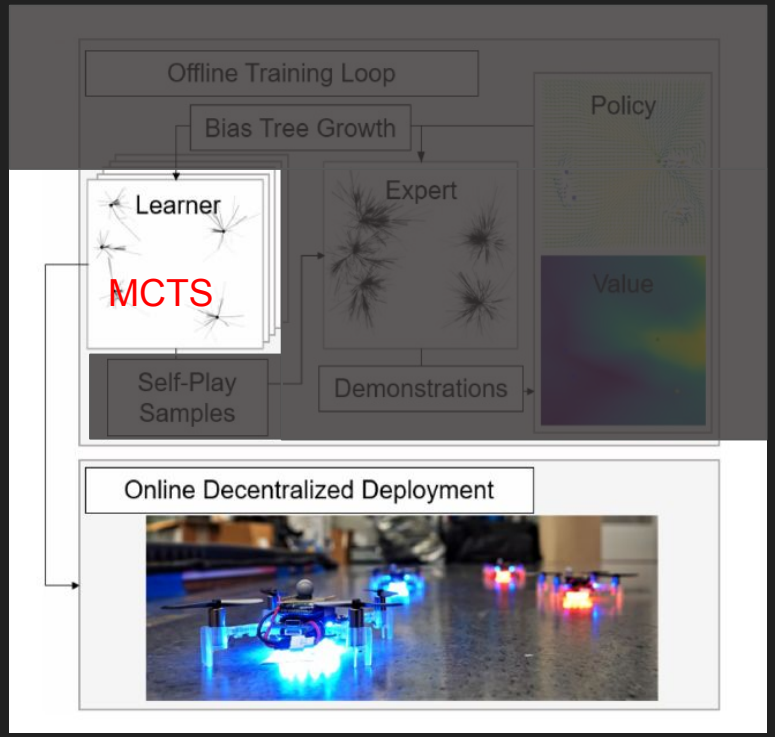
Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Method



Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Method



Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Our Contribution - Reward Function

Source: Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environment

Approach

- PUCT search
- Agents trained with local knowledge (sensing radius)
- Polar representation for surroundings and movement
- Explicit agent prioritization in reward function

Polynomial Upper Confidence Trees (PUCT)

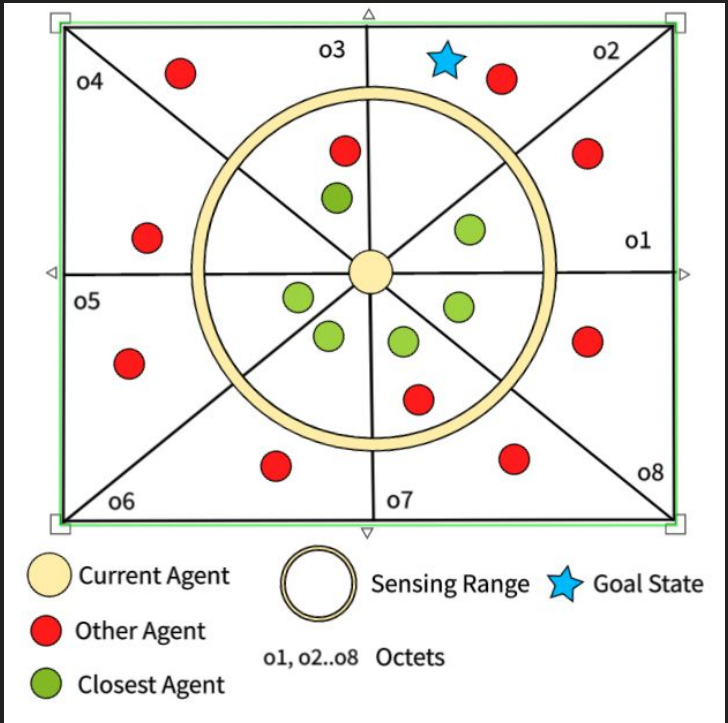
- Used in AlphaZero
- Variant of MCTS
  - Selection
  - Expansion
  - Simulation
  - Backpropagation
- Think MiniMax

State Representation

$$s_t = <x_{ri}, y_{ri}, p_{ri}, x_{gi}, y_{gi}, [d_1 \ldots d_8], [p_1 \ldots p_8]>$$

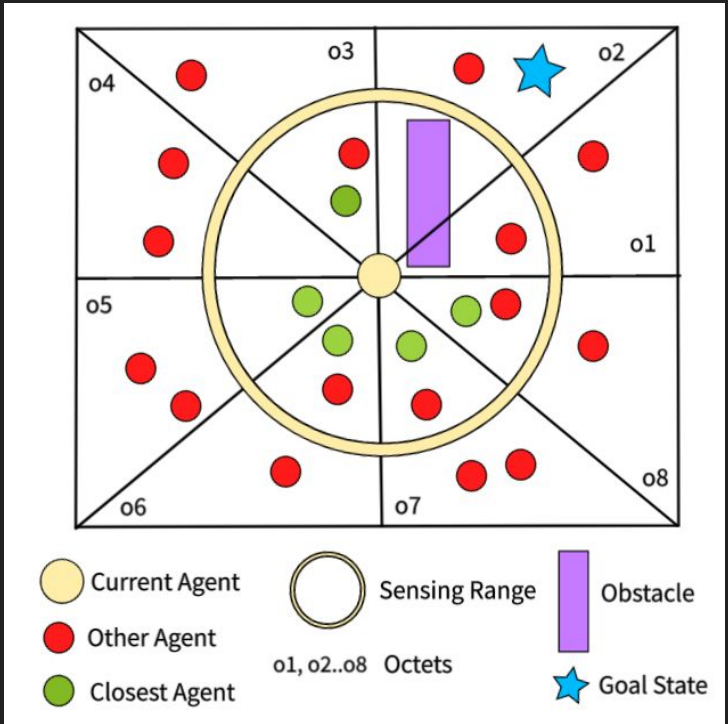
- x_{ri} x-coordinate of robot *i*
- y_{ri} y-coordinate of robot *i*
- p_{ri} priority of robot *i*
- y_{gi} x-coordinate of the goal of robot *i*
- y_{gi} y-coordinate of the goal of robot *i*
- $[d_1...d_8]$ distance to the closest agent in each octant $[o_1...o_8]$
- $[p_1...p_8]$ priority of the closest agent in each octant $[o_1...o_8]$

State Examples



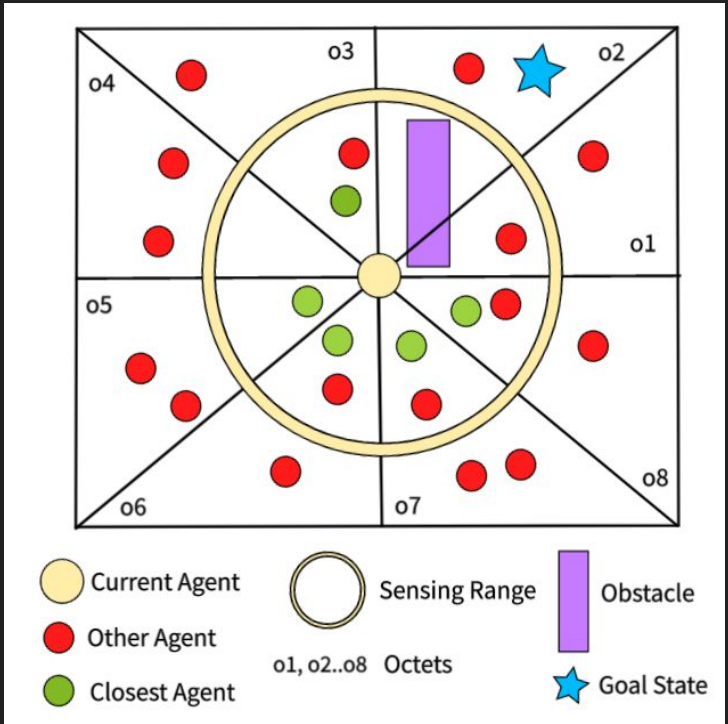
- x_{ri} 0
- y_{ri} 0
- p_{ri} 1
- x_{gi} 6
- y_{gi} 1
- $[d_1...d_8]$ [2.8, 4, 1.3, 4, 1, 0.9, 1.1, 1.2]
- $[p_1...p_8]$ [1, 0, 0.4, 0, 0.2, 0.6, 0.5, 0.5]

State Examples



- x_{ri} 0
- y_{ri} 0
- p_{ri} 1
- x_{gi} 8
- y_{gi} 1
- $[d_1...d_8]$ [0.3, 0.7, 1.3, 4, 1, 0.9, 1.1, 1.2]
- $[p_1...p_8]$ [2, 2, 0.4, 0, 0.2, 0.6, 0.5, 0.5]

State Examples



- Information in each octant considers closest agent in that octant
- Regions with no agents?
- Regions with an obstacle?

Reward Function

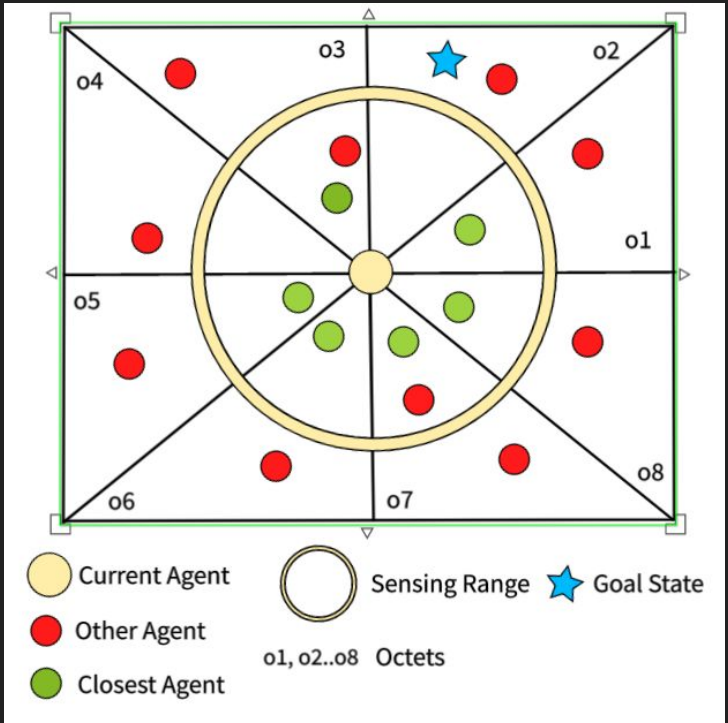
$$R_x = \frac{1}{d_g} + \sum_{i=1}^{8} (P_x - P_i)\left(1 - \frac{d_{xi}}{r_x}\right)$$

- R_x reward for agent *x*
- d_g distance between agent *x* and its goal
- P_x priority for agent *x*
- P_i closest agent in octant *i*
- d_{xi} distance between agent x and the closest agent in octant *i*
- r_x sensing range of agent *x*

But that’s just math

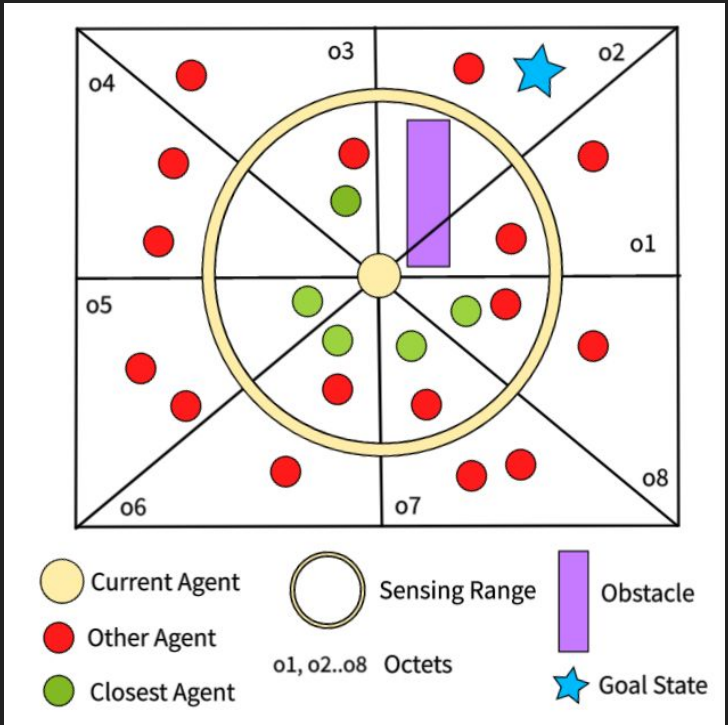
- Incentivize moving towards goal
- Yield right of way when approaching another agent with a higher priority
- Continue with current trajectory if approaching agent with lower priority
- Avoid obstacles

Reward Examples



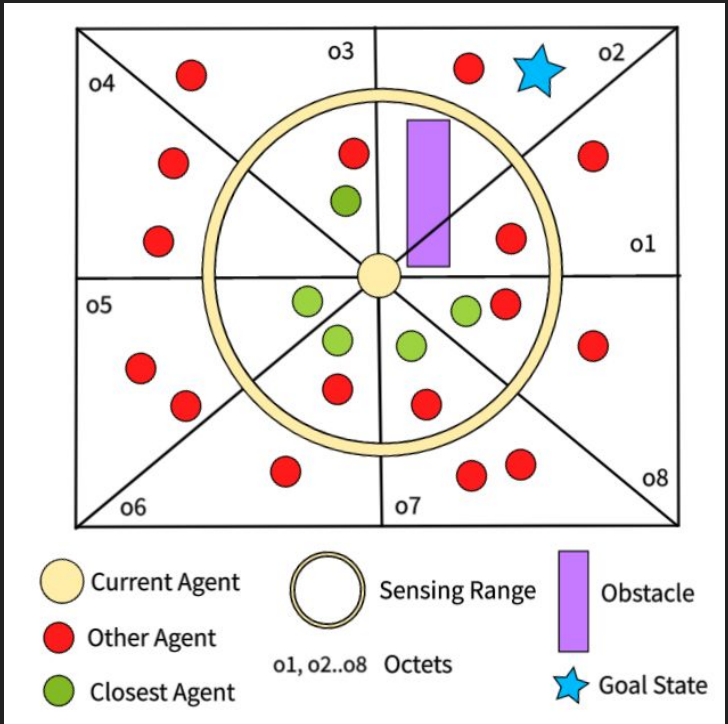
- a_0 1
- a_1 1
- a_2 1
- a_3 1
- a_4 1
- a_5 1
- a_6 1
- a_7 1
- a_8 1

Reward Examples



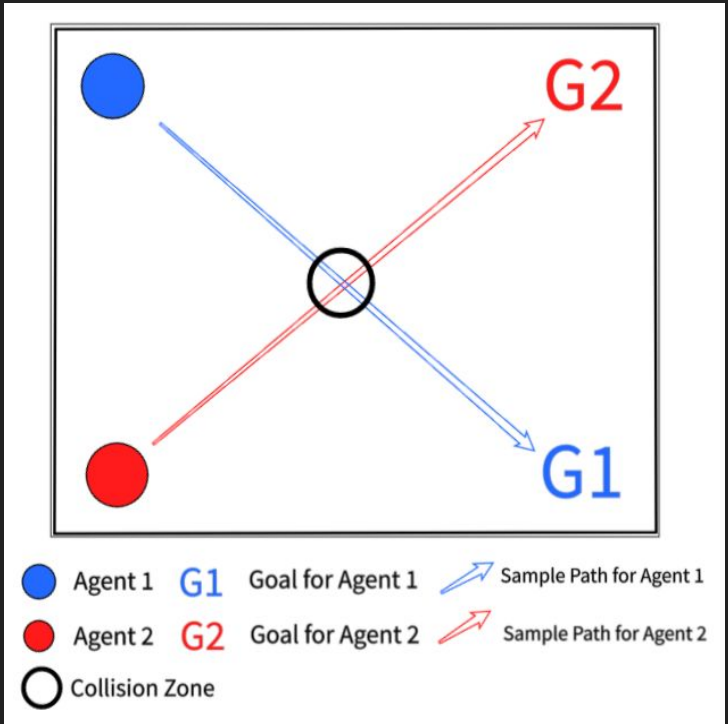
- a_0 1
- a_1 1
- a_2 1
- a_3 1
- a_4 1
- a_5 1
- a_6 1
- a_7 1
- a_8 1

Reward Examples



- Move towards goal
- Move in direction with no agents
- Move away from agents with high priority
- Avoid obstacles!

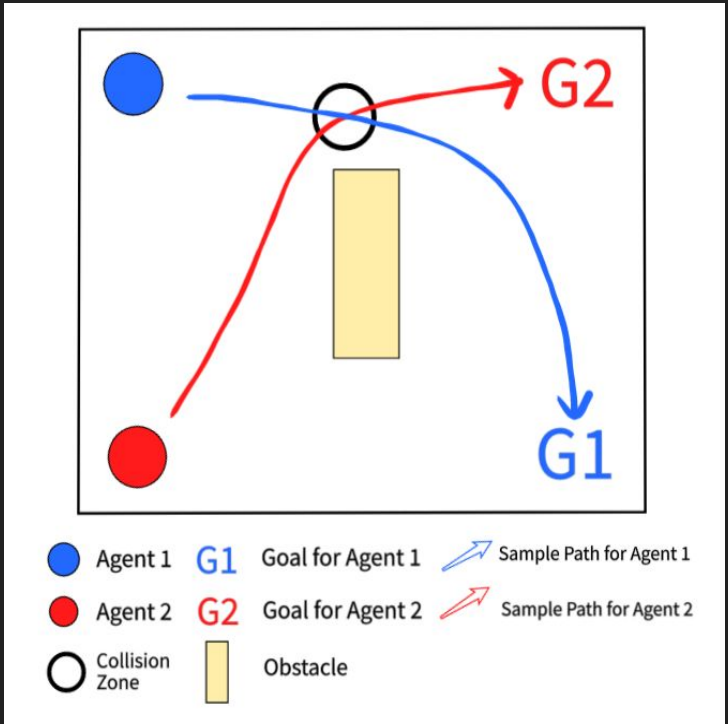
Experiments



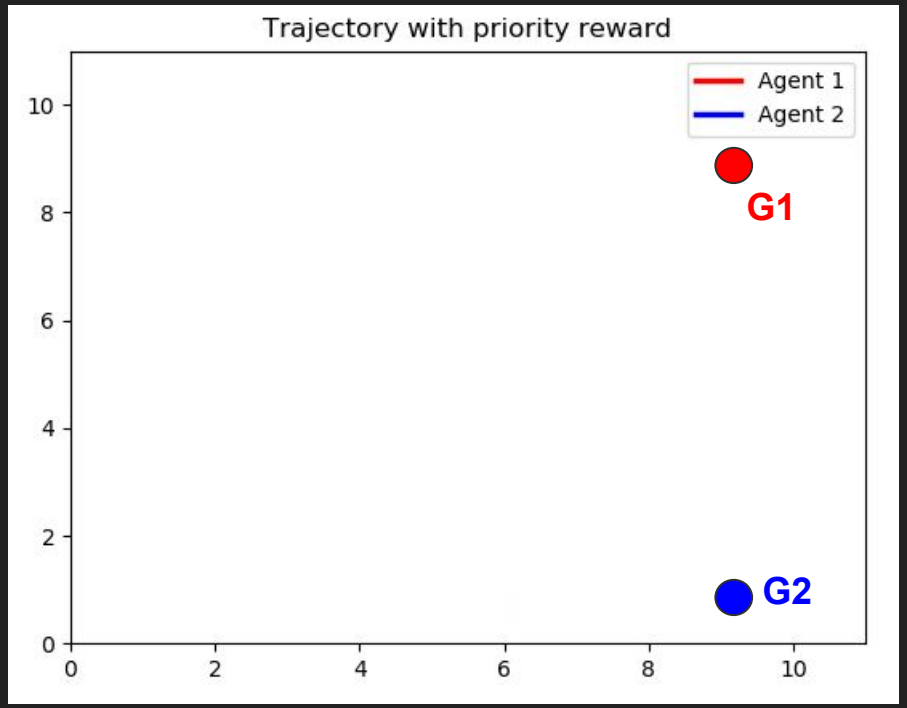
- Training iterations: 10/50
  - Expert demos epochs: 100
  - Expert demos iterations: 100
  - Search depth : 25

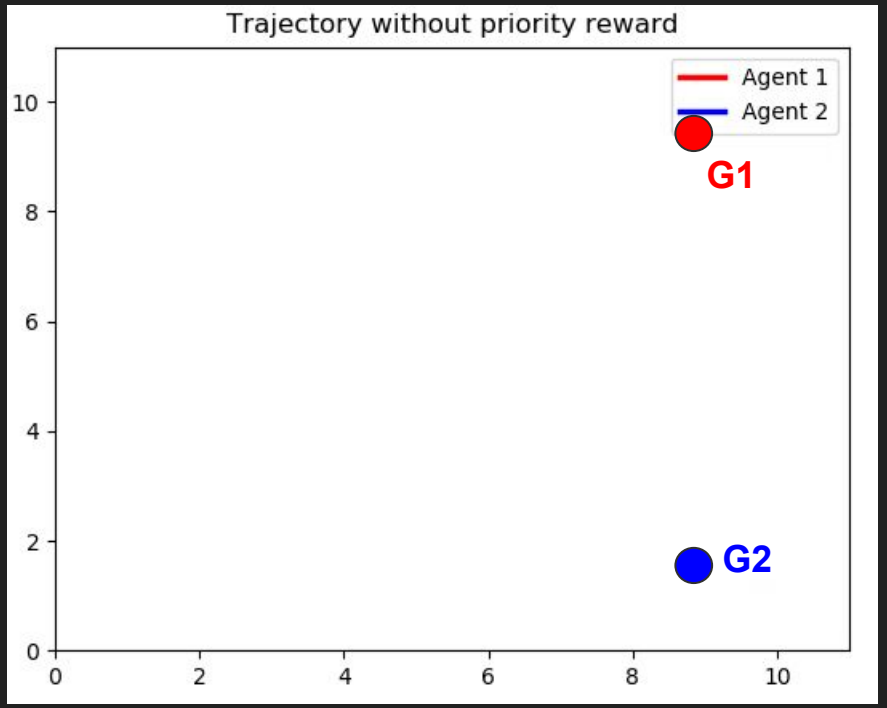
- Learning rate: 0.001

Experiments

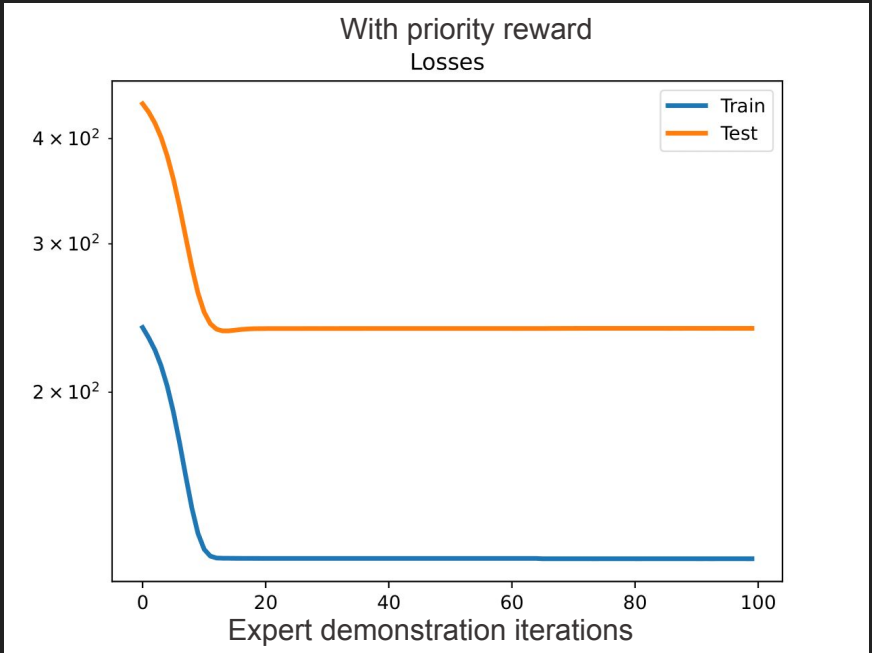


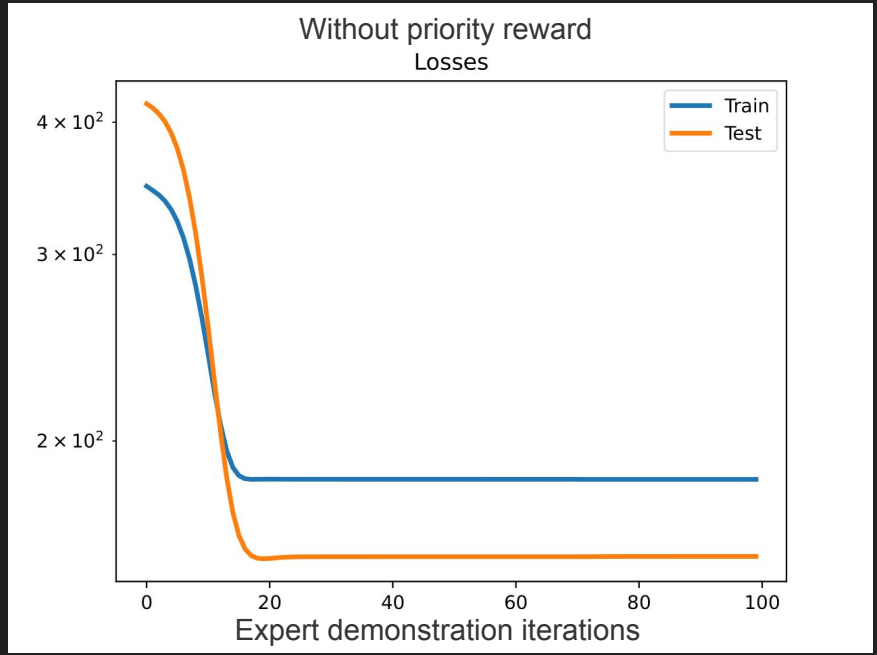
Results - Trajectories





Results - Value Learning curves





Current Limitations and Next Steps

- Priority weightage
- All agents have the same speed
- All agents have the same sensing range

More! More! More!

- Scale up number of agents
- Larger map with obstacles
- System throughput?
- Common goals?
- Variable speeds/Continuous action space?

Questions?