Learning Heterogeneous Multi-agent Coordination and Cooperation for Joint Perception-Control Tasks

Special Problem Student: Xiyang Wu
Supervision: Esmaeil Seraj (PhD Student) and Matthew Gombolay (PhD)

Institute for Robotics & Intelligent Machines (IRIM)
November 30th, 2020

CORE
ROBOTICS LAB

Georgia Institute of Technology

Background

- **FireCommander Environment [1]**
  - Heterogenous Multi-agent: Multiple agents taking different tasks exists in this environment
  - Sequential Multi-task: Fire fronts need to be sensed first before they could be put out
- **Coordination and Cooperation in Multi-agent Team**
  - Cooperation between agents both play the same and different role helps to improve the overall performance of the MARL method
- **Experience Sharing Between the Heterogenous Team**
  - Accelerate the training process by selecting the experiences that are most likely to lead to high reward
  - Find out the correlation between tasks taken by sub-agent teams that plays different roles.

Current Progress

- **Current Progress**
  - **Proposed Environment**
    - Simple validation environment: Predator and Prey
    - FireCommander: Heterogenous with and without the Priorized Targets
    - Reward Function Design
  - **Benchmark Algorithms [2]**
    - Fully Independent Methods: IDQN [3]
    - Learning Communication: CommNet [4], DIAL [5]
    - Learning Cooperation: QMIX [6], COMA [7]
    - Attention Based MARL Methods: G2ANet [8]
  - **Experience Sharing Method**
    - Perception to Action (P2A) Sharing: Meta-Learning
    - Action to Perception (A2P) Sharing: Attention Mechanism

Current Progress

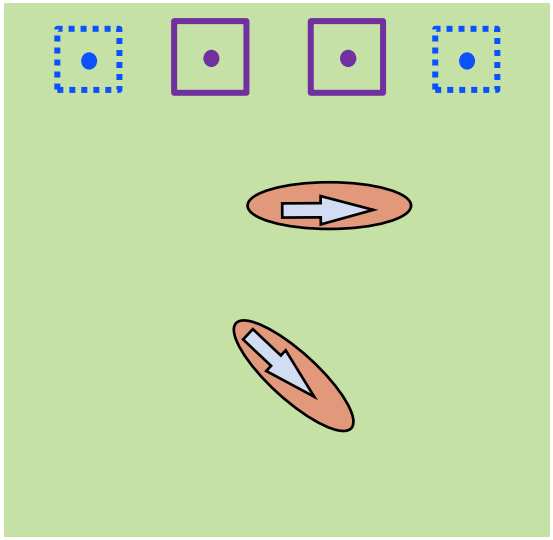
- **Checklist for Current Progress**

Environment	Benchmarks	Baselines	Writing
Done:	**Done:**	**Done:**	**Done:**
• FireCommander Environment Implementation	• IDQN	• Separated COMA	• Environment Description
• Reward Function Design	• CommNet	**To Do:**	• Benchmark Description and Result
To Do:	• QMIX	• Priorized Experience Replay in COMA	**To Do:**
• Predator and Prey	• COMA	• Attention Mechanism	• Experiments
	To Do:	• Meta-Learning Experience Mapping	• Mathematical Proof and Validation
	• G2ANet	• Full Experience Sharing Module	

Environment

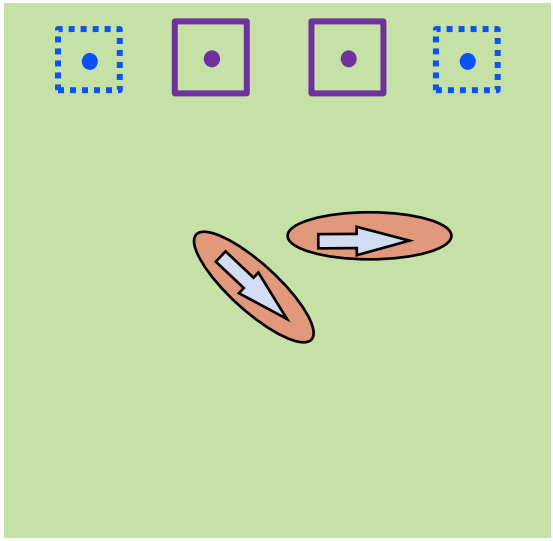
- **Predator and Prey (Competitive and Cooperation)**
  - **Three kinds of agents:** Prey, Scout Predator, Action Predator
  - **General Policy:** Action predator could not detect the prey unless being sensed by the scout predator
- **Fire Commander (Cooperation)**
  - **Two kinds of agents:** Perception Agent, Action Agent
  - **General Policy:** Action agent could not detect the prey unless being sensed by the scout predator
  - **Discrete State Space:** 84 * 84 grid world while each block could only be occupied by one kind of component: fire, agent, etc
  - **Discrete Action Space:** Agent could only change their position or height by 1 at each time
    - Perception: Forward, Backward, Left, Right; Up, Down (Height Range: 3 - 12)
    - Action: Forward, Backward, Left, Right (Height Fixed: 5)
  - **Probabilistic Action:** Both perception and action could partially complete their task, like sensing or pruning 90% active fire fronts within the region, while the probability for the perception agent is determined by its height and the probability for the perception agent is fixed

Heterogeneous Scenario



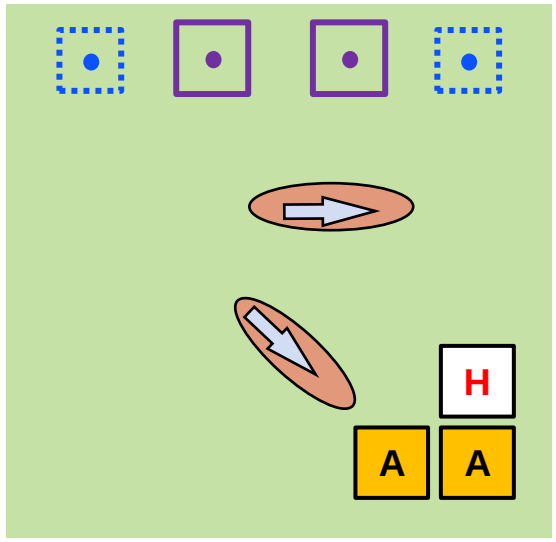
Scenario #1

- **Perception Agents:** (40, 10), (60, 10)
- **Action Agents:** (20, 10), (80, 10)
- **Fire Regions:** (70, 20), (70, 70)
  - Initial Fire Front Number: 2, 2
  - Wind Direction: 90°, 45°
  - Fuel Coefficient: 5, 5



Scenario #2

- **Perception Agents:** (40, 20), (60, 10)
- **Action Agents:** (20, 10), (80, 10)
- **Fire Regions:** (40, 50), (60, 50)
  - Initial Fire Front Number: 2, 2
  - Wind Direction: 90°, 45°
  - Fuel Coefficient: 5, 5



Scenario with Targets

- **Perception Agents:** (40, 10), (60, 10)
- **Action Agents:** (20, 10), (80, 10)
- **Fire Regions:** (70, 20), (70, 70)
  - Initial Fire Front Number: 2, 2
  - Wind Direction: 90°, 45°
  - Fuel Coefficient: 5, 5
- **Targets:**
  - House: (65, 75), (75, 75)
  - Hospitals: (75, 65)
  - Target Size: 10 Pixel

Reward Policy

- **General Idea**
  - Encourage the agents to explore the environment and put out the fire as soon as possible
    - Temporal penalty on agent team
  - Ensure the balance between the positive and negative reward introduced by the variation fire status
    - Select the weights on the perception, action and fire propagation reward so that the terminal reward equals to 0 when all the fire fronts have been sensed and pruned
  - Protect the facilities with priority
    - Set serious penalty when the fire fronts propagate into the facilities
  - Enhance fire pruning action over perception
    - Set higher reward made by pruned fire fronts over sensed ones

Reward Details

- **Positive Reward**
  - **Perception Score**
    - The number of sensed fire spots among all the fire spots generated

$$Number\ of\ \ Discovered\ Firespots \times 0.5$$

  - **Action Score**
    - The number of pruned fire spots in all the sensed fire spots

$$Number\ of\ Pruned\ Firespots \times 1.5$$

  - **Success Reward:**
    - The great reward offered when the ratio of pruned fire with all the fire spots generated reaches 95%

$$Episode\ Length \times Number\ of\ agents$$

Reward Details

- **Negative Reward**
  - **Temporal Penalty**
    - The number of sensed fire spots among all the fire spots generated

$$Passed\ Time\ Step\ Number \times 0.05$$

  - **Fire Propagation Penalty**
    - Total number of fire spots that have been generated ever, including the active and pruned ones

$$(Number\ of\ Pruned\ Firespots\ + Number\ of\ Active\ Firespots) \times 2.0$$

  - **Facility Damage Penalty**
    - The number of pruned fire spots in all the sensed fire spots

$$Total\ Number\ of\ Acitve\ Firespots\ in\ Houses \times 2.0$$
$$Total\ Number\ of\ Acitve\ Firespots\ in\ Hospitals \times 5.0$$

Benchmarks

- **Fully Independent Methods:**
  - Fully eliminate the communication module to verify the necessity for the cooperation between agents
    - Independent Deep Q-Learning (IDQN)
- **Learning Communication:**
  - Suppose the communication module exists between agents and try to learn the optimal information exchange policy when communicating
    - Differentiable Inter-Agent Learning (DIAL), CommNet
- **Learning Cooperation**
  - Introduce the ideas from multi-agent learning into MARL, including the
    - Actor-critic Based Methods: Counterfactual multi-agent policy gradients (COMA)
    - Value Based Methods: QMIX
- **Attention-based MARL Methods**
  - Implement the attention mechanism into the MARL to determine when and who to communicate
    - Game Abstraction Mechanism based on Two-stage Attention Network (G2ANet)

Input Vector Design

- **Perception Agent:**
  - Reads the observation from the environment and the state conveyed from peers as the input

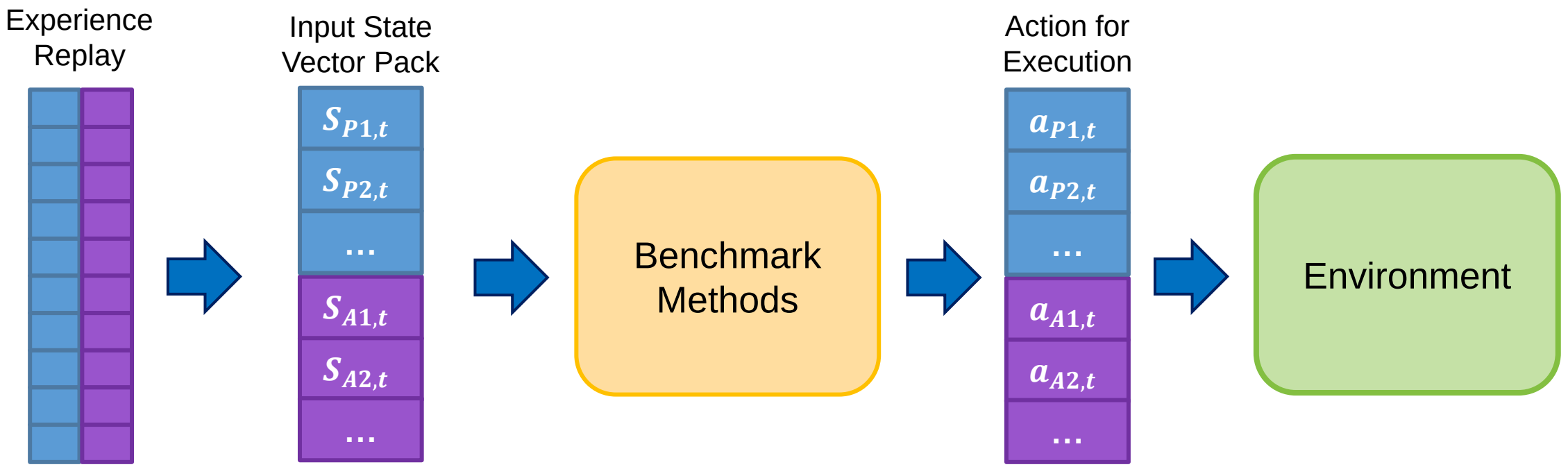
Environment Observation Vector	State Observation Vector	Action Vector	Agent ID
512-bit vector Encoded from 25 × 25 observation matrix via CNN	One-hot vectors for state Position: Two 84-bit for (x,y) Height: 9-bit, indicating 3 - 12 Agent type: [1, 0] for Perception Complete ratio: 100-bit in Percentage	6-bit one-hot vector	4-bit one-hot vector

- **Action Agent:**
  - Only reads the state conveyed from peers as the input

Zero Vector	State Observation Vector	Action Vector	Agent ID
512-bit Zero vector Occupies the space initially for 25 × 25 observation matrix	One-hot vectors for state Position: Two 84-bit for (x,y) Height: Fixed 9-bit Agent type: [0, 1] for Action Complete ratio: 100-bit in Percentage	6-bit one-hot vector	4-bit one-hot vector

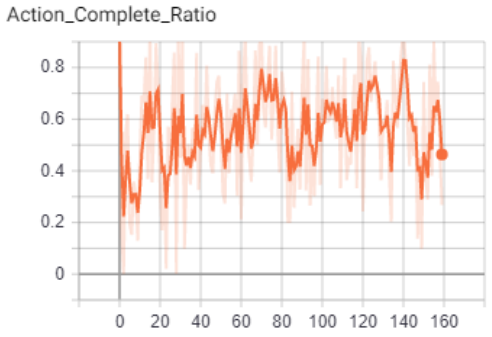
Benchmarks

- **Training Framework:**
  - Combine the state of the heterogenous agents into a single batch, only differentiate by the agent type label
  - The network read the whole batch and generate the action for execution separately in the environment

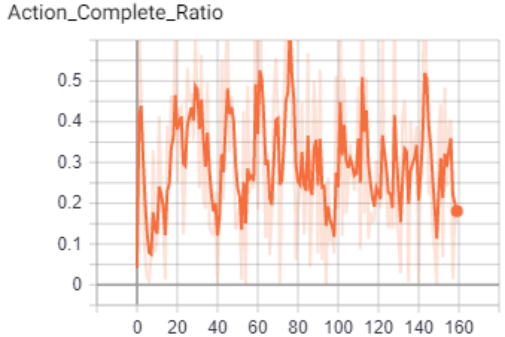


Performance Criteria

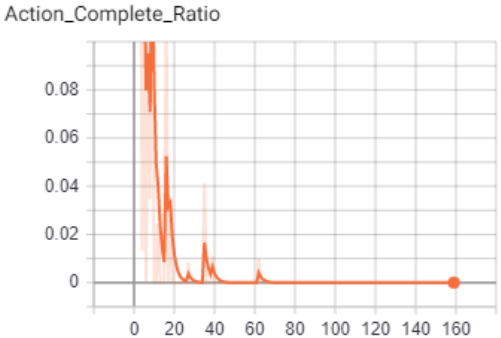
- **Action Performance Ratio:**
  - The number of pruned fire fronts versus all fire fronts that have been generated at the end of each game



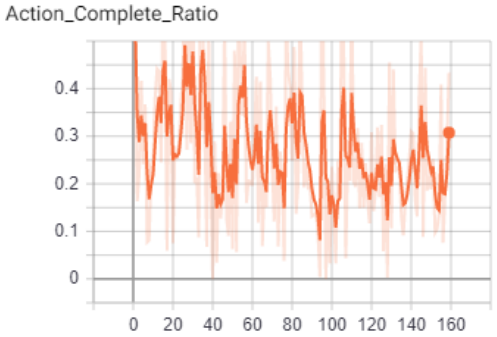
COMA



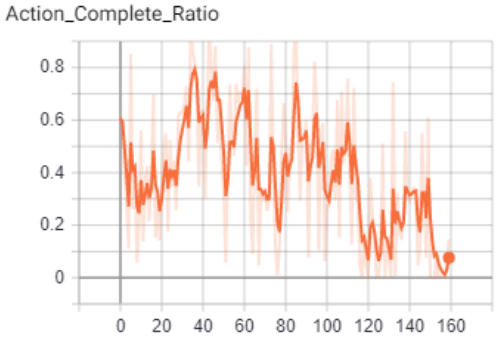
QMIX



DIAL



CommNet

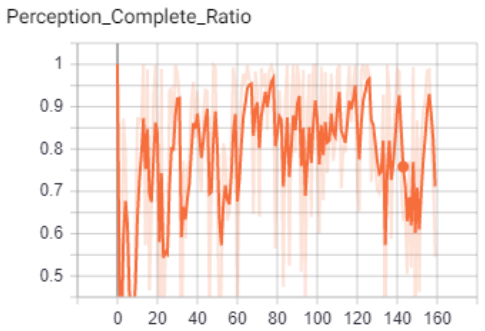


IDQN

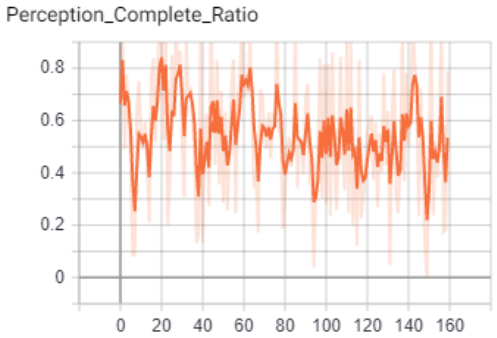
Note: All results comes from the training result in 11/11/2020

Performance Criteria

- **Perception Performance Ratio:**
  - The number of sensed fire fronts versus all fire fronts that have been generated at the end of each game

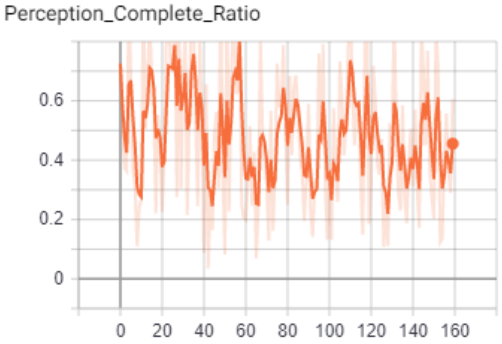


COMA

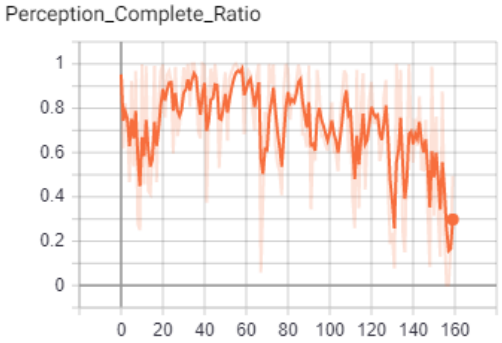


QMIX

DIAL



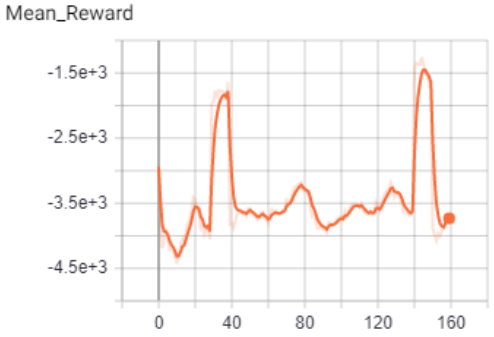
CommNet



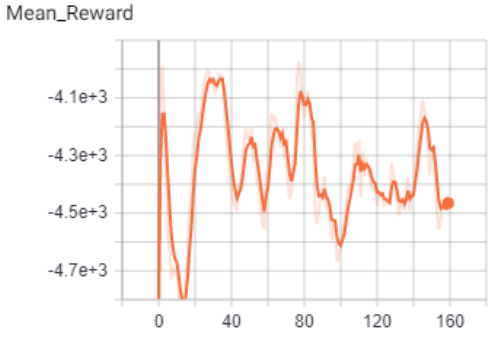
IDQN

Performance Criteria

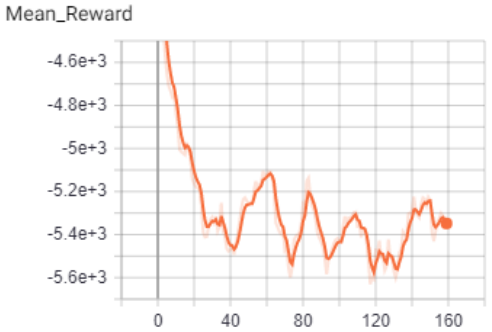
- **Reward Variation:**
  - The number of sensed fire fronts versus all fire fronts that have been generated at the end of each game



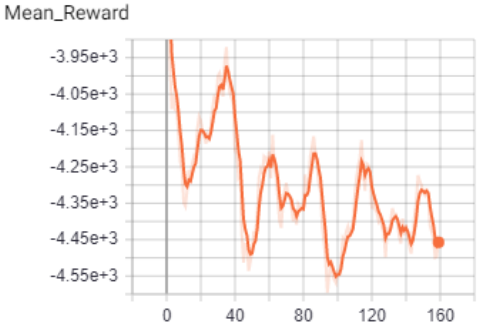
COMA



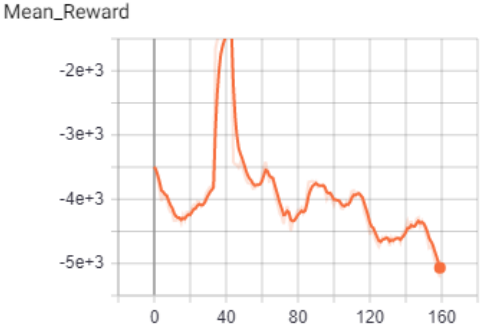
QMIX



DIAL



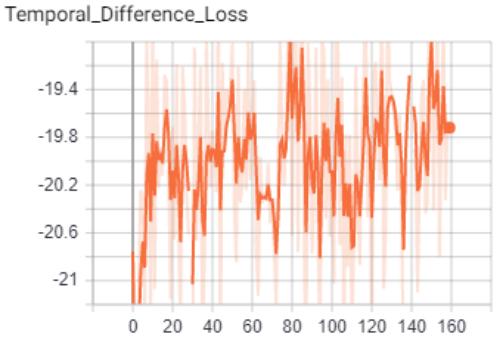
CommNet



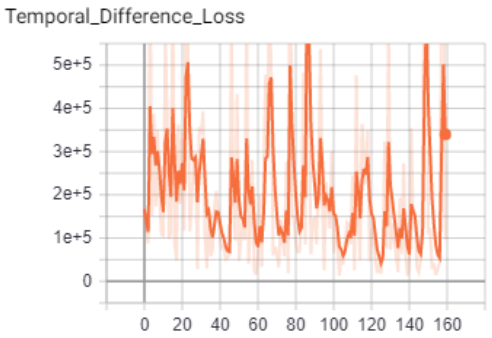
IDQN

Performance Criteria

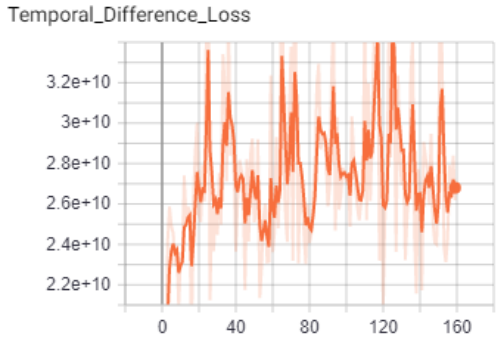
- **TD-Error:**
  - The number of sensed fire fronts versus all fire fronts that have been generated at the end of each game



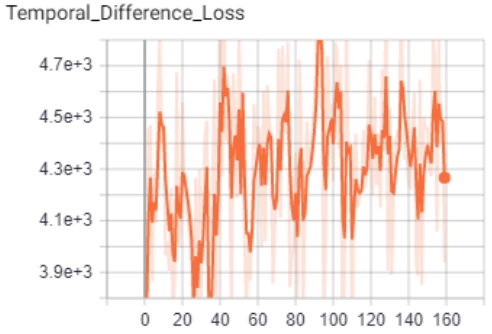
COMA



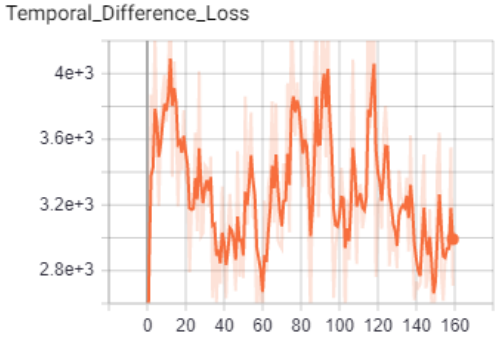
QMIX



DIAL



CommNet

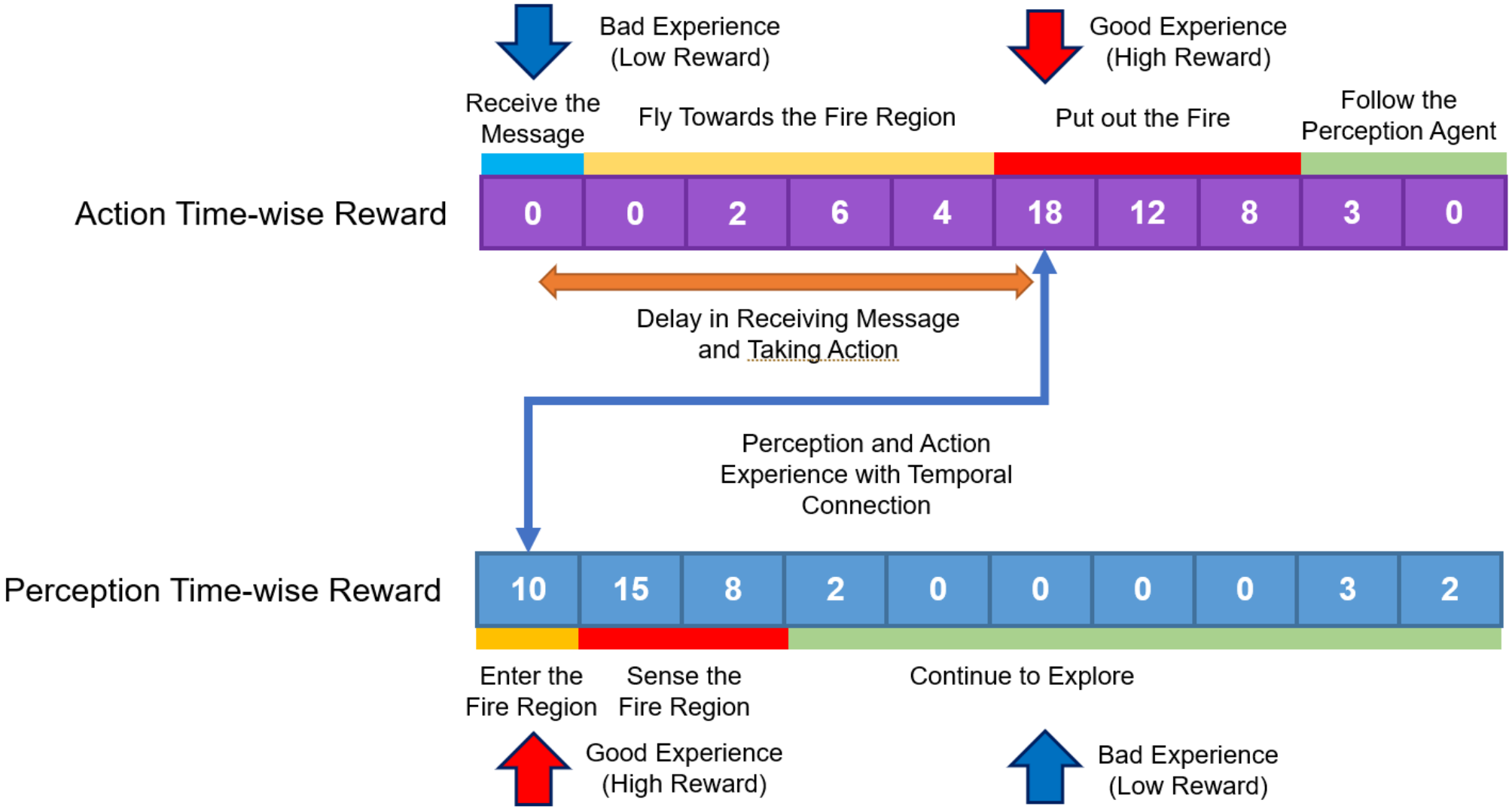


IDQN

Benchmark Weaknesses

- **Heterogenous Interface:**
  - Use the same network to control heterogenous team, while agents with different types may interface with each other
- **Ignoring the Temporal Correlation in Tasks:**
  - Fire fronts need to be sensed before they could be pruned, but action agents may fly over the target region before the region has been sensed, which makes the method non-optimal
- **Catastrophic Forgetting:**
  - The training is unstable that the optimal parameters may be hard to be stored for later execution, easy to stuck around 50 – 60 % action complete ratio

Temporal Correlation



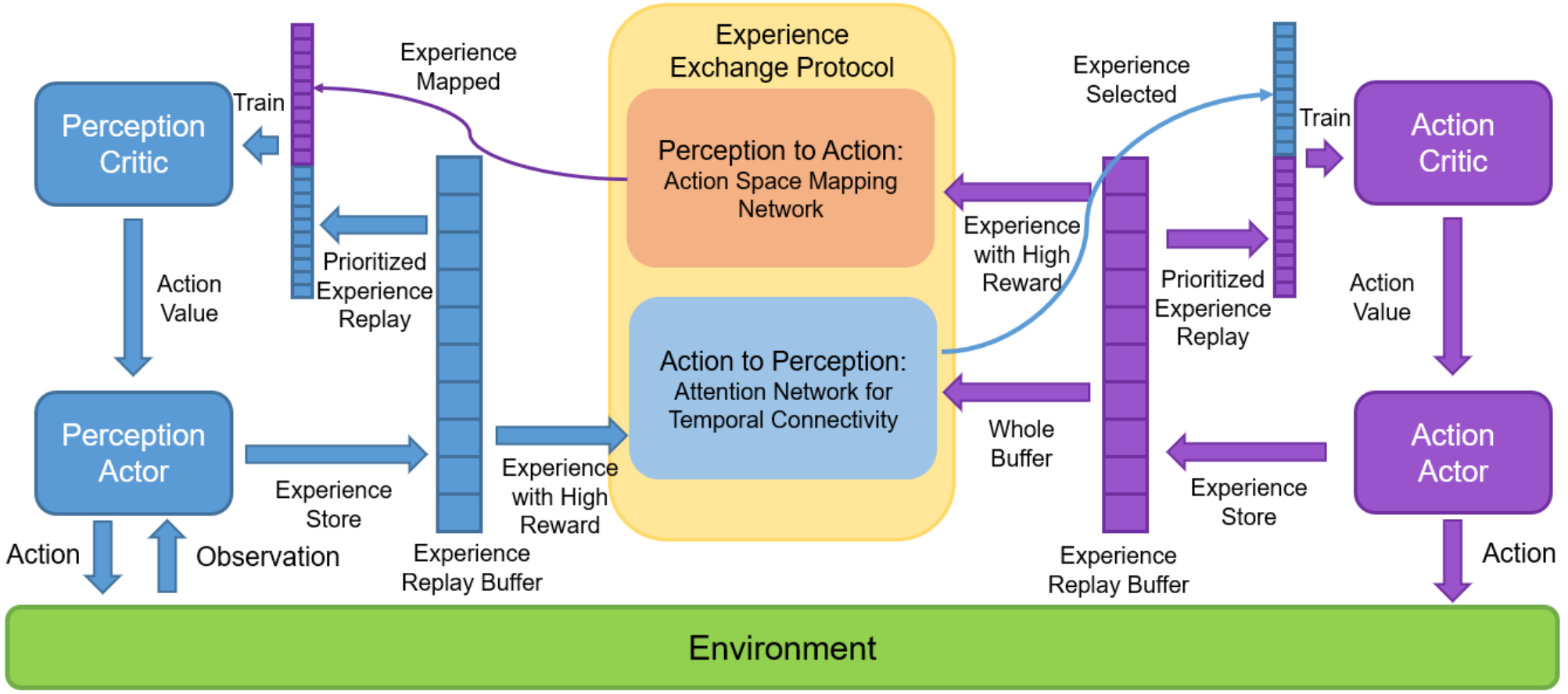
Temporal Correlation

- **Phenomena**
  - During the training process of the benchmark method, there's an obvious delay between the fire being sensed by perception agents and being pruned by action agents, which is much larger than the world size (1000 Time Steps v.s. 84 Pixels)
  - Due the propagation nature of the fire fronts, the exploration time for the perception agent could be much more the exploitation time, which is typically continuous. This leads to several possible results:
    - More of the time the perception agents are doing low reward action, the sparse high-reward action may be easily forgotten by the agents
    - Since the action agent only reads the state information sent by their peers, the low-reward experience may interface the learning process of the agents
    - Even though the action agents have learnt the high reward action, once they arrive the target region earlier than the perception agents, this action is still meaningless

Experience Sharing Framework

- **Motivation**
  - Separate the network for heterogenous agents to avoid the interface, but still take advantage of the benchmark method in controlling homogenous agent team
    - Use separate MARL networks for perception and action agent sub-team
  - Extract the high-reward experience (“Good” Experience) for critic network training
    - Use priorized experience replay for critics
  - Learn the sequential tasks with heterogenous sharing network
    - Use different mechanism for Action to Perception (A2P) and Perception to Action (P2A) experience sharing
  - Focus on the temporal correlation between the tasks
    - Attention mechanism experience sharing (Action to Perception, A2P)
  - Exploit the high-value perception experience to accelerate the training of the action agent
    - Meta-learning experience sharing (Perception to Action, P2A)

General Experience Sharing Framework



General Experience Sharing Framework

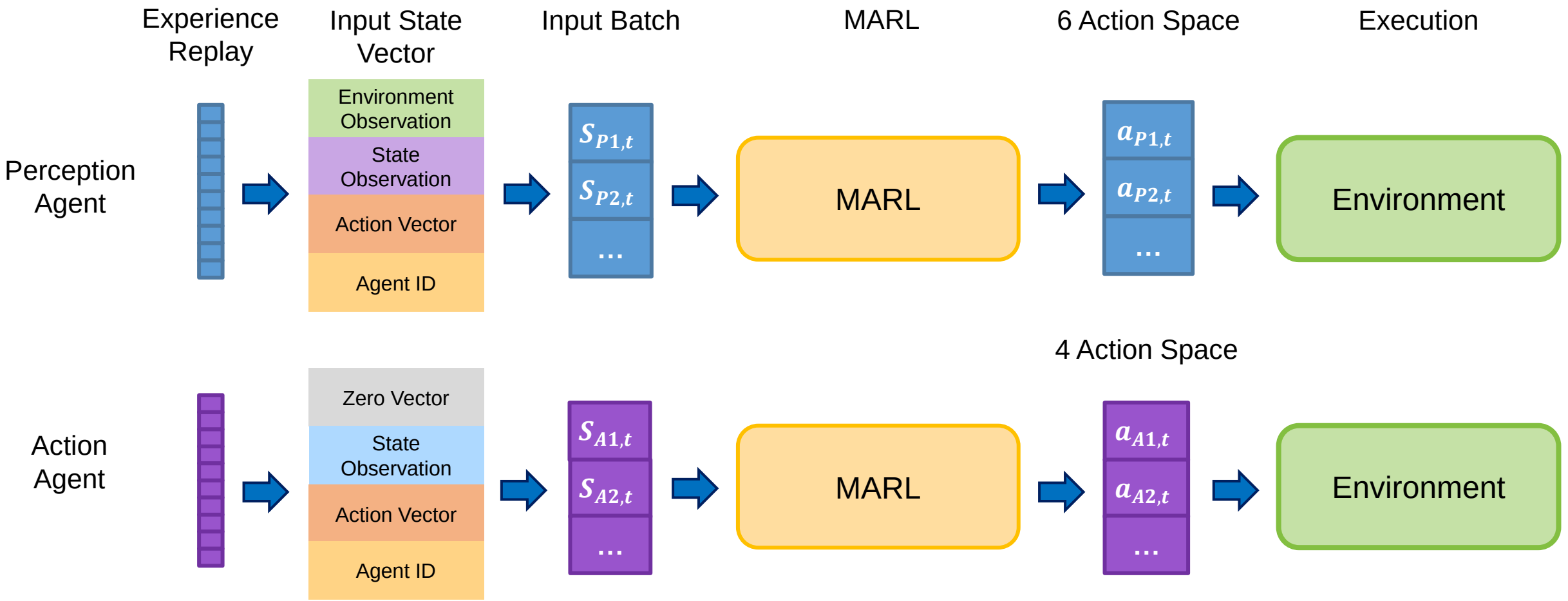
- **Interpretation**
  - For each kind of agents, use the actor-critic framework, while the critic is centralized in each sub-team and the actors are decentralized for execution
  - The experience replay buffer is based on the execution record for each kind of agents, i.e. each sub homogenous agent team is controlled by the backbone methods derived from the benchmarks (e.g. COMA)
  - Following the backbone methods, the experience replay is sequential for actor network, while the priorized experience sharing and replay only happens on the critic network for each sub-agent team
  - When training the critic network, some of the experiences come from the peer sub agent team via sharing while other come from the priorized experience replay in its own buffer

Perception to Action Mapping: Meta Learning

- **Why Meta-Learning**
  - "Learn to Learn"
  - Similarities and Differences Between the Perception and Action Tasks
    - Similarities:
      1. Same Execution MARL network within the homogenous sub agent team
      2. Similarity in the task space (Perception: All fire fronts have been generated, Action: All sensed fire fronts), while the spatial position of the fire front in the real world is fixed at each time step
    - Differences:
      1. Different Input (State Vector) and Output (Action Space), Different agent number
      2. Different task space in the given time step that fire fronts must be sensed first before they could be pruned. Temporal delay exists in the sequential tasks
  - **Motivation:** Exploit the high-reward experience made by the perception agent to train the action agent and help to maximize the reward of the action agent, which is the terminal purpose of the game

Perception to Action Mapping: Meta Learning

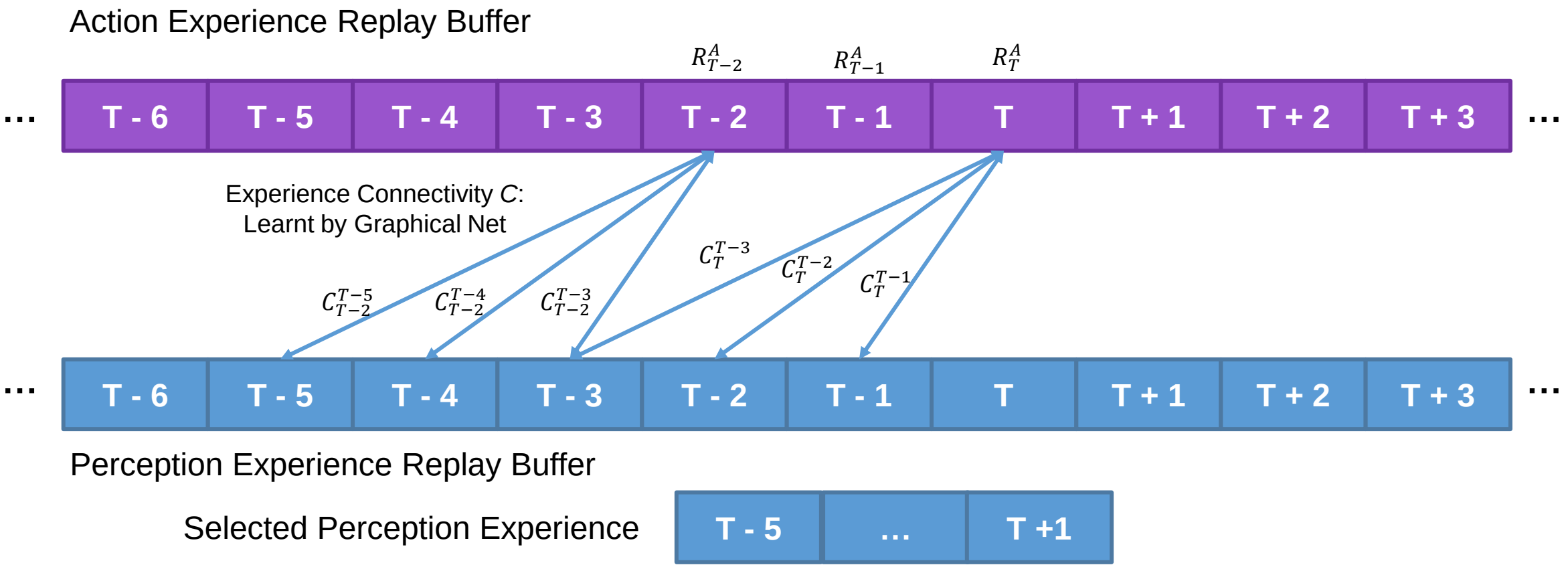
- **Similarities and Differences Between the Perception and Action Tasks**



Action to Perception Mapping: Attention Mechanism

- **Why Attention Mechanism**
  - The perception experience and action experience have temporal correlation
  - The high-reward action for the perception agents is most likely leads to the high-reward action for the action agents several steps later, but the reverse statement is not true
  - Sometimes, some relatively low reward action may reveal more potential to contribute to high reward action for the action agents later (e.g. Find and enter a new fire region) than others, which should be encouraged
- **Attention Mechanism:**
  - The feedback for the Perception to Action mapping
  - Select the experiences in the perception agent's experience replay buffer that are most likely to contribute to the high-reward action for action agents

Action to Perception Mapping: Attention Mechanism



Score for perception agent's experience at time t: $E(t=T) = \sum_{i=1}^{L} R_T^A \, C_T^{T-i}$

Experience selected: Top K experience with highest $E(t)$

Baselines

- **Backbone Algorithm: COMA [7]**
  - Separate perception and action agent team into two homogenous MARL sub-team
  - Take the perception and action task as independent tasks
  - Use the centralized critic network for each sub agent team with decentralized actor network for each agent in this sub-team

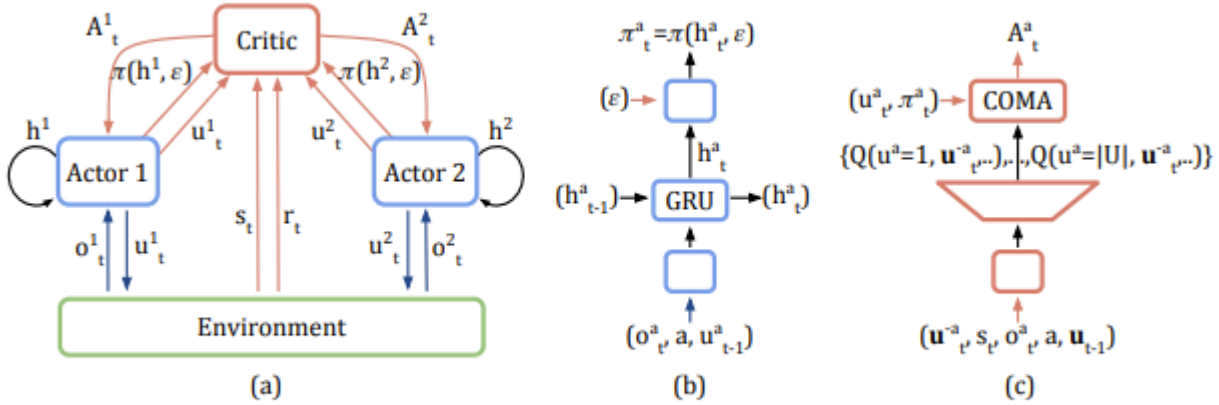


Figure 1: In (a), information flow between the decentralised actors, the environment and the centralised critic in COMA; red arrows and components are only required during centralised learning. In (b) and (c), architectures of the actor and critic.

Baselines

- **Backbone Algorithm: COMA**
- **Baselines:**
  - Separated COMA for Perception and Action agent team without communication
  - Priorized experience replay for critic with top k reward
  - Attention based experience sharing (A2P)
  - Meta learning experience sharing (P2A)
  - **Full experience sharing module (Final Result)**

Baselines

- **Baselines:**
  - **Separated COMA for Perception and Action agent team without communication**
  - **Priorized experience replay for critic with top k reward**
    - Only select several experience to train the critic network, like the top k experience with highest reward
    - For actor network training, follow the temporal sequential order in the experience replay buffer
  - **Attention based experience sharing (A2P)**
    - Perception to Action: N/A, use top 2k priorized experiences to train the action agent's critic network
    - Action to Perception: k experiences selected via attention mechanism, k priorized experiences
  - **Meta learning experience sharing (P2A)**
    - Perception to Action: k experiences mapped from the perception agents, k priorized experiences
    - Action to Perception: N/A, use top 2k priorized experiences to train the perception agent's critic network
  - **Full experience sharing module (Final Result)**
    - Perception to Action: k experiences mapped from the perception agents, k priorized experiences
    - Action to Perception: k experiences selected via attention mechanism, k priorized experiences

Problem To Be Answered

- **Parameters in Top k Attention Mechanism**
  - How to determine the value of k? Is there any relationship between k and experience replay buffer size?
  - How to determine the ratio between the experiences selected with the attention or meta-learning mechanism and the prioritized policy?
- **Possible Solutions:**
  - Use importance weights before experience replay instead of eliminating experiences
  - Use SOFT top-k operator [9] instead of top-k operator

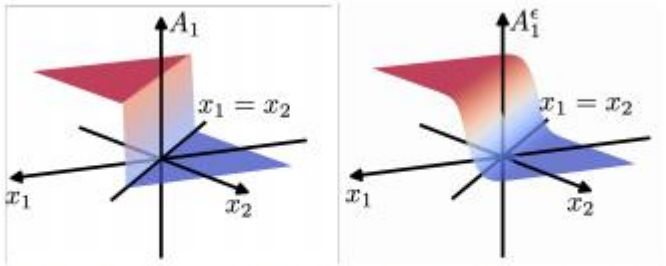


Figure 1: Indicator vector with respect to input scores. Left: original top-k operator; right: SOFT top-k operator.

Problem To Be Answered

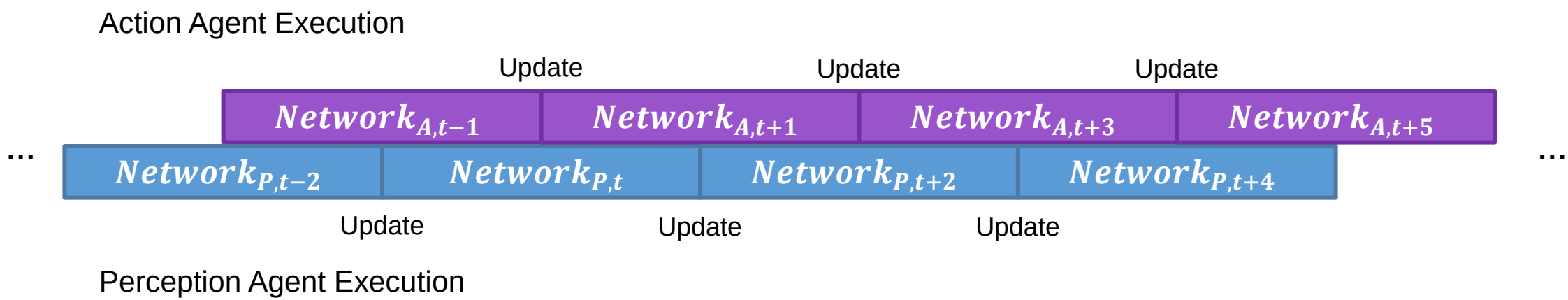
- **Justify the Top-k mechanism**
  - *Prioritized experience replay* [10] said that merely select the experiences with top k reward may change the distribution in the experience generation, so that they proposed the experience sampling method with SumTree and uniform sample in the range divided by SumTree
  - Taking heuristic knowledge into sampling, like selecting the experiences with top k reward, or the most recent k experiences, even for the critics, may not be valid
- **Possible Solutions:**
  - Add compensation coefficient to the reward or TD-error during the loss computation that represents the importance of the given experience

Problem To Be Answered

- **Meta-Learning Structure**
  - How to utilize the meta-learning framework in training the critic network for the action agent?
  - What to share? The parameters for the trained perception critic network in the last updating period or only the experience?
  - How to incorporate the previous action critic and the new shared network?
- **Possible Solutions:**
  - When sharing experience, duplicate current perception critic network and use the selected action agent's experience to update the parameters in the duplicated network

Problem To Be Answered

- **Delay in Taking Action**
  - Once the near-policy has been learnt, chances are that the
- **Possible Solutions:**
  - Relay experience replay and updating policy, i.e. only update the network for one kind of agents at a specific network back-propagation, leaving the other network static



Reference

[1] Seraj, Esmaeil, Xiyang Wu, and Matthew Gombolay. "FireCommander: An Interactive, Probabilistic Multi-agent Environment for Joint Perception-Action Tasks." arXiv preprint arXiv:2011.00165 (2020).

[2] Hernandez-Leal, Pablo, Bilal Kartal, and Matthew E. Taylor. "A survey and critique of multiagent deep reinforcement learning." Autonomous Agents and Multi-Agent Systems 33.6 (2019): 750-797.

[3] Tan, Ming. "Multi-agent reinforcement learning: Independent vs. cooperative agents." Proceedings of the tenth international conference on machine learning. 1993.

[4] Sukhbaatar, Sainbayar, and Rob Fergus. "Learning multiagent communication with backpropagation." Advances in Neural Information Processing Systems. 2016.

[5] Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

[6] Rashid, Tabish, et al. "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning." arXiv preprint arXiv:1803.11485 (2018).

[7] Foerster, Jakob N., et al. "Counterfactual multi-agent policy gradients." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[8] Liu, Yong, et al. "Multi-Agent Game Abstraction via Graph Attention Neural Network." AAAI. 2020.

Reference

[9] Operator with Optimal Transport." arXiv preprint arXiv:2002.06504 (2020).

[10] Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015).