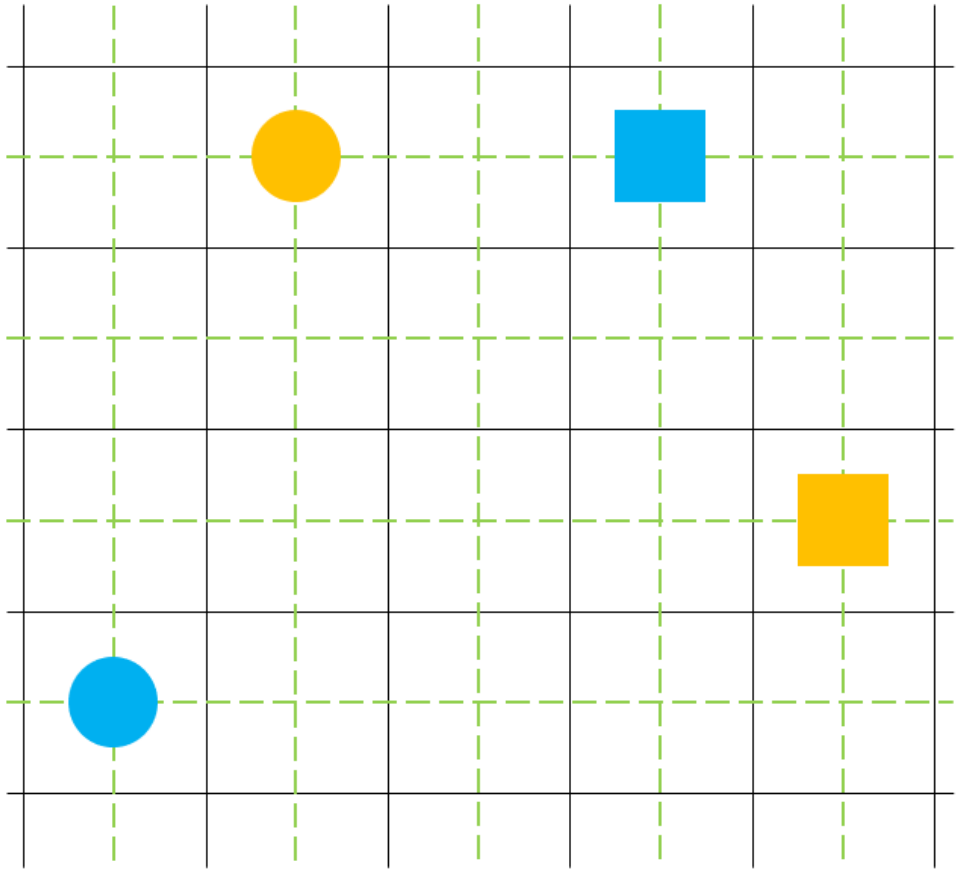
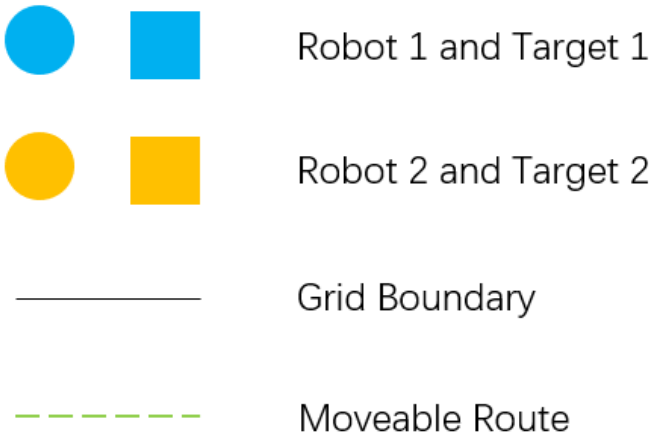
One-shot Multi-agent Path Finding

Hongjun Chen

2020.9.29

Path Finding(Environment)

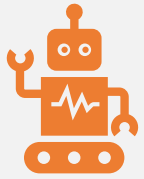




(Small) Environment:
rectangular grid map： M × N
robots： X

Movement:
—up, down, left, right, standstill
—**Asynchronous**
—Uncertain time-consuming

Task



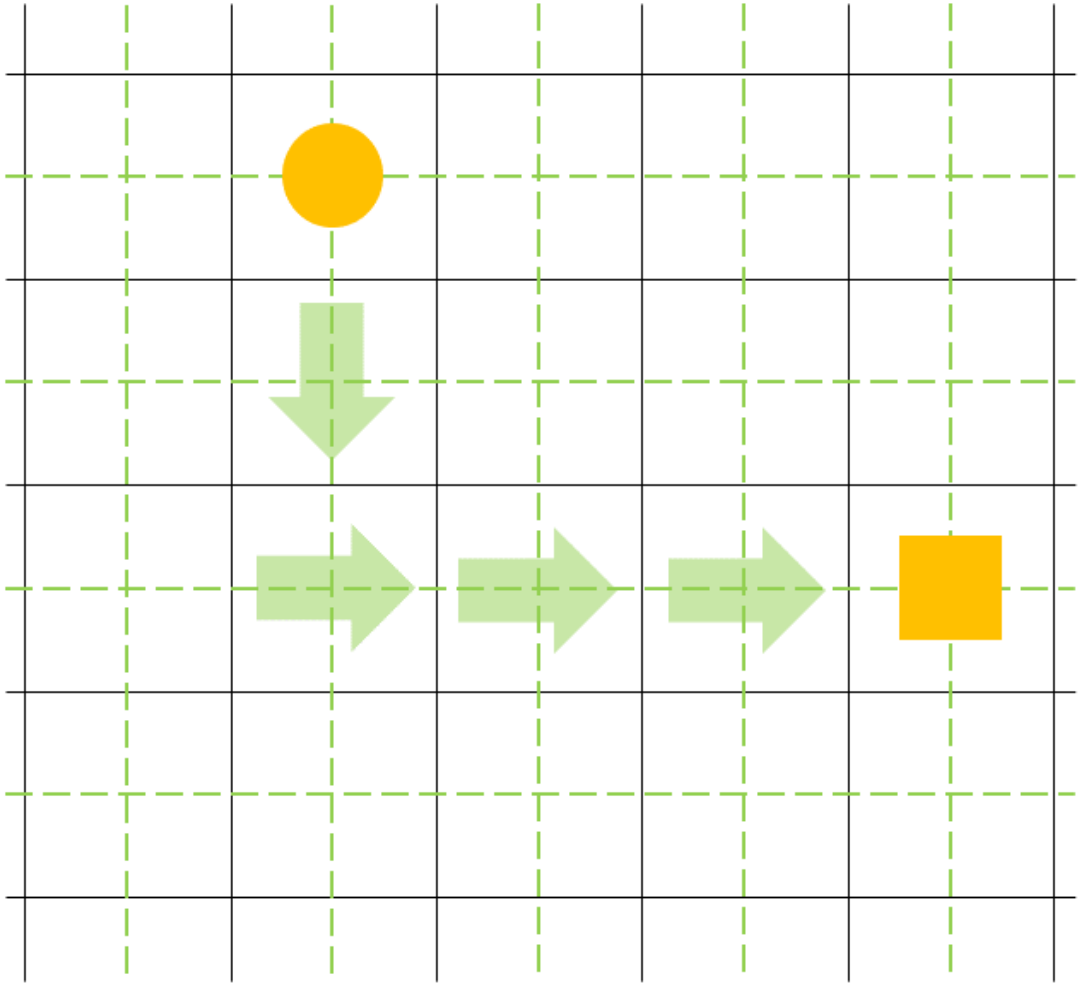
Route Planning

Avoiding confliction between

- Robot & Robot
- Robot & Boundary

To overmatch baseline algorithm(A-Star) with less step-cost

Path Finding (Static Solution)

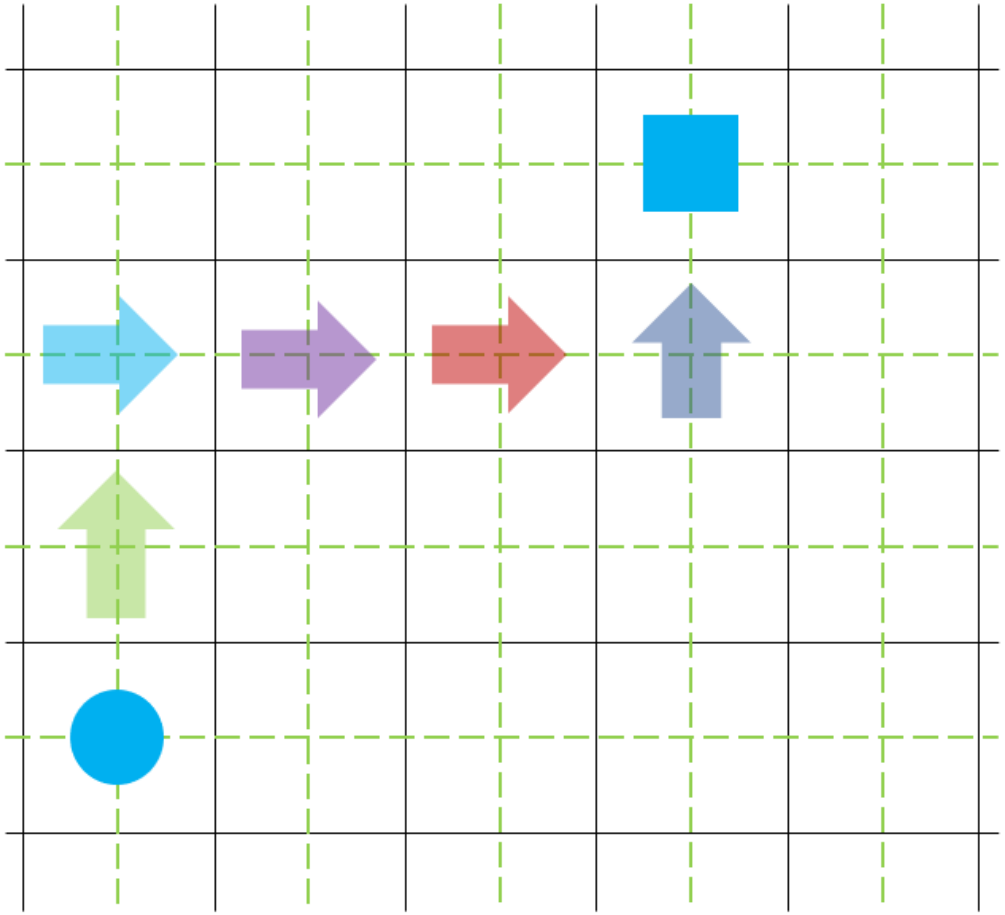


Baseline Algorithm (A-star, of one robot)

- Plan the entire path at a time

- 传统planning方法诸如A*、M*等在不同环境中难以灵活平衡exploration与exploition；
- 随着问题规模提升，centralized planner的**探索空间快速增长**，计算耗时难以控制；
- 而decentralized planner规划路径时不考虑其他agent与动态环境，因此经常需要在无法执行规划结果时进行replan，在多智体场景下replan的开销非常高昂

Path Finding (Dynamic Solution)



RL-based Algorithm (of one robot)

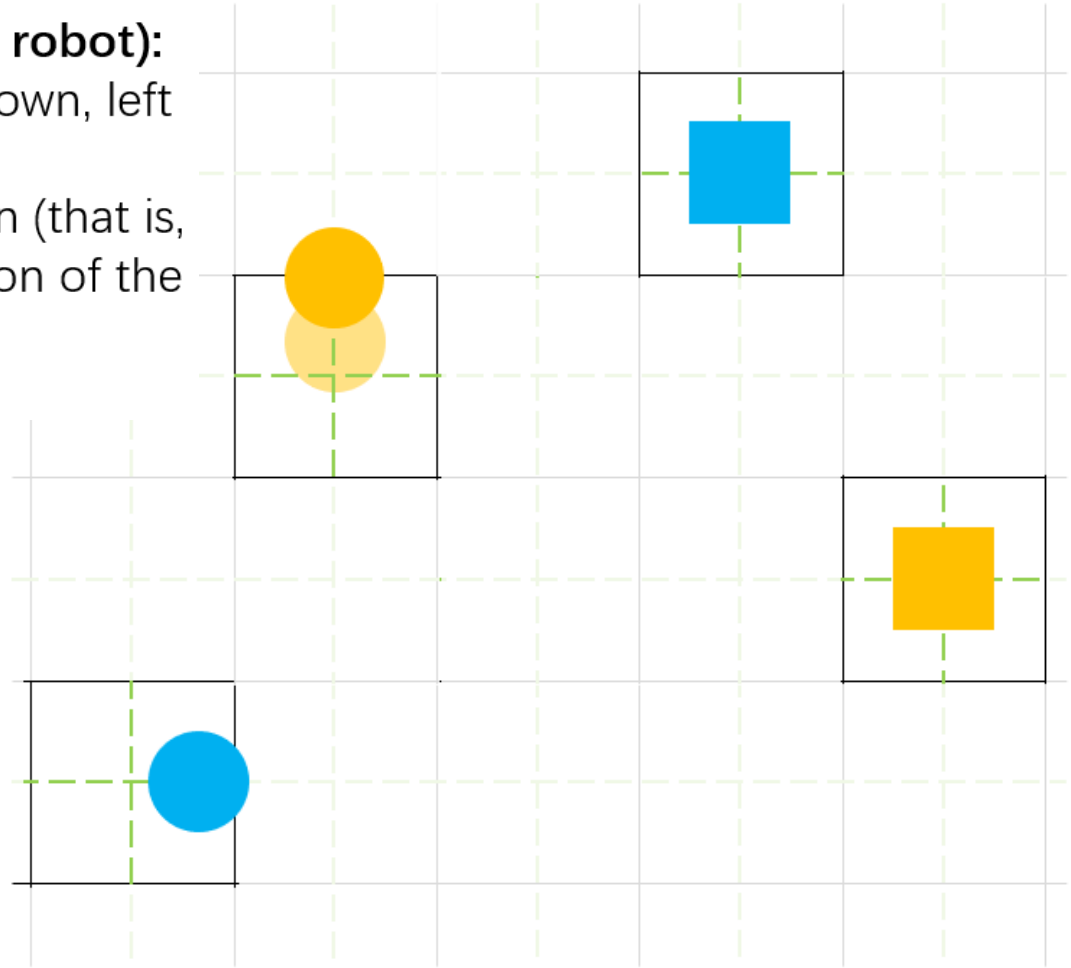
- different **colors** represent **decisions** made at different timesteps

Designing Observation & Action

Action space (of one robot):

- non-moving: up, down, left right, standstill
- moving: no decision (that is, maintain the decision of the previous timestep)

PS:
There is 50% for a robot to arrival next position each timestep.



State space (of one robot):

- the cell in which it is currently located (cell id)
- the cell where other robots are currently located
- the cell in which the target is currently located
- The cell where other robot targets are currently located
- Other robots heading (**just for moving robot**)

Adapting Asynchronous & Reward Design

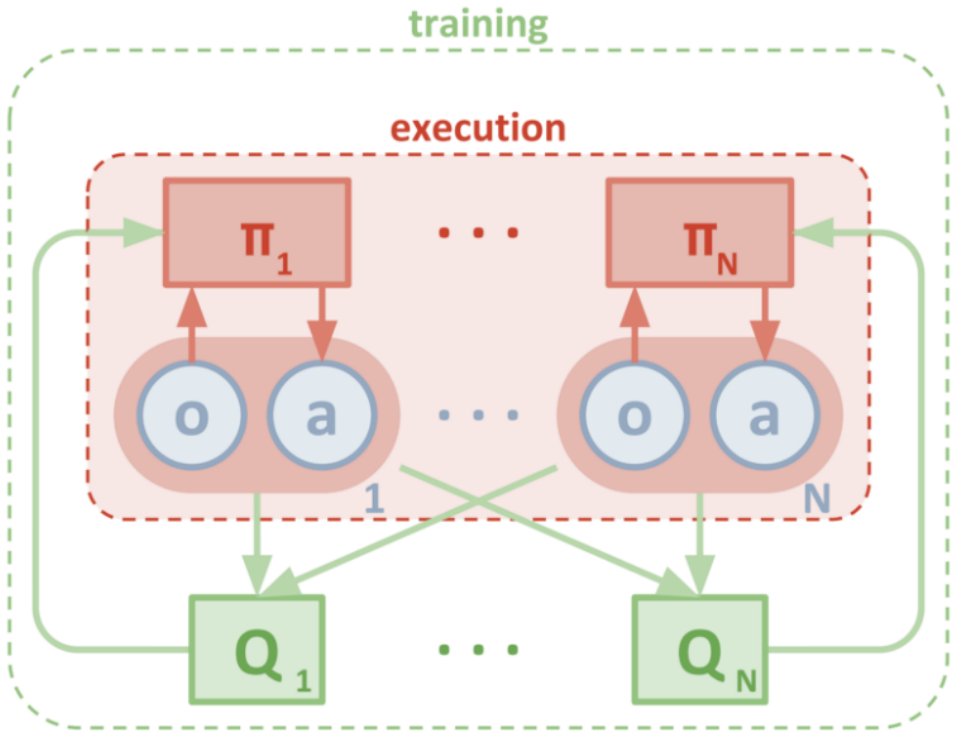
- In order to adapt **synchronous algorithm** into the **asynchronous movement**, we define **discrete decision** within each timestep.
  - Setting *is_ready* to indicate whether a robot is moving or not.
  - *True* for moving, *False for complete.*

- For each agent:
  - Observation includes:
    - ✓ Distance between each robot and its target
    - ✓ Distance between each robot and others
    - ✓ Previous action of all robots
    - ✓ Current position and target position of all robots
    - ✓ Movement indication of a robot

- Reward for an agent setting:
  - Having confliction -> -10
  - Single movement per timestep -> -1
  - Arriving at its target -> +1000
  - While moving:
    - Keeping current action -> -1
    - Changing action -> -10

MAAC solving MAPF

- Multi-Actor-Attention-Critic(MAAC)
  - Centralized learning / Decentralized executing
  - Each actor matches a critic
  - Critics share an attention network



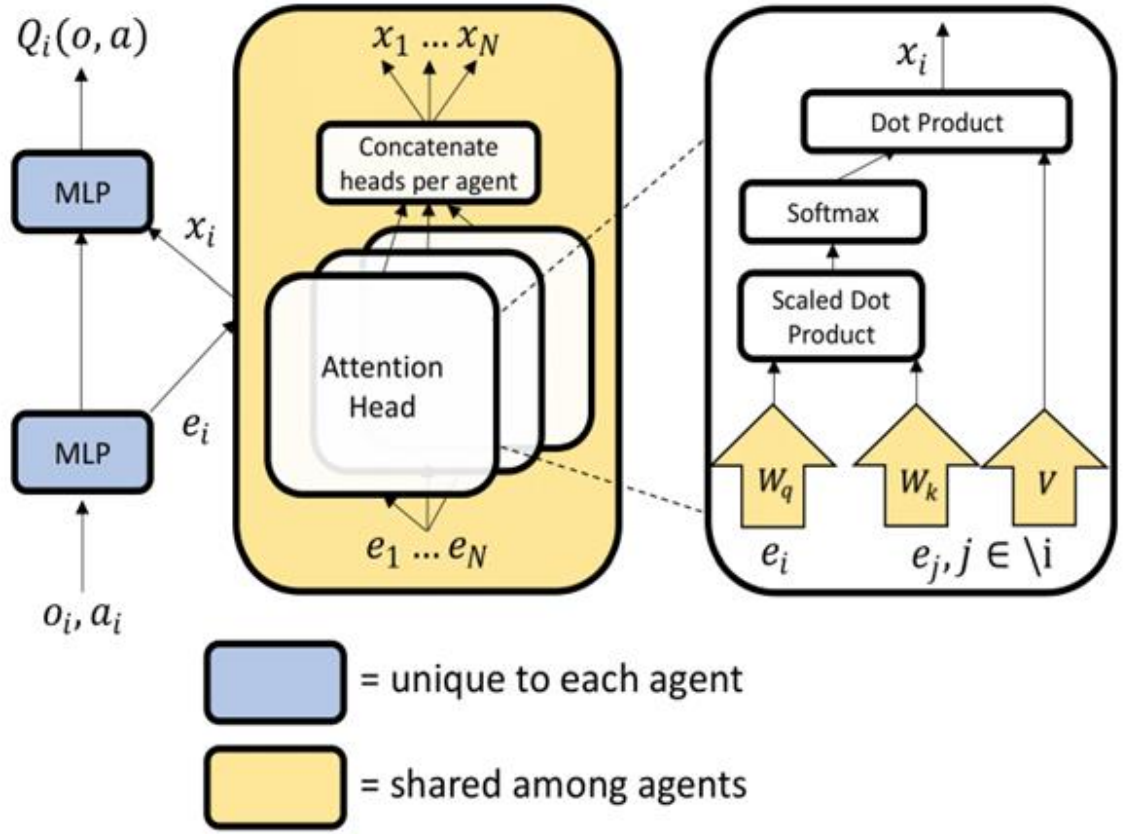
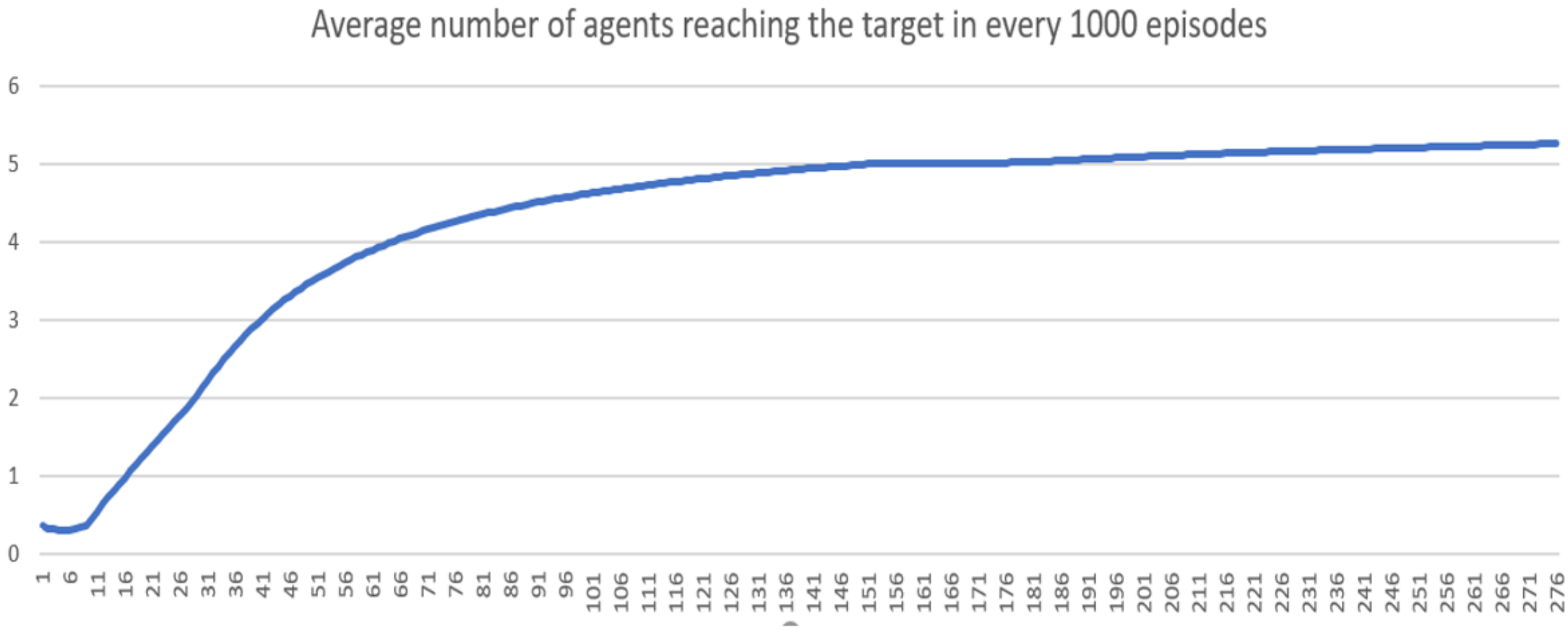


Figure 1. Calculating $Q_i^{\psi}(o,a)$ with attention for agent i. Each agent encodes its observations and actions, sends it to the central attention mechanism, and receives a weighted sum of other agents encodings (each tranformed by the matrix V)

[1] Iqbal, Shariq, and Fei Sha. "**Actor-attention-critic for multi-agent reinforcement learning**." *International Conference on Machine Learning*. PMLR, 2019.

Result (env: 8×8 world size， 6 agents)



Result (env: 8×8 world size， 6 agents)

分析与思考

- Centralized training的MARL（如MAAC）容易导致部分agent采取passive policy，最终体现为**到达率（success rate）低下**；

- 将传统planning方法与decentralized RL方法相**结合**，是一个比较合理的思路，其中Imitation Learning是一个比较有价值的方向。

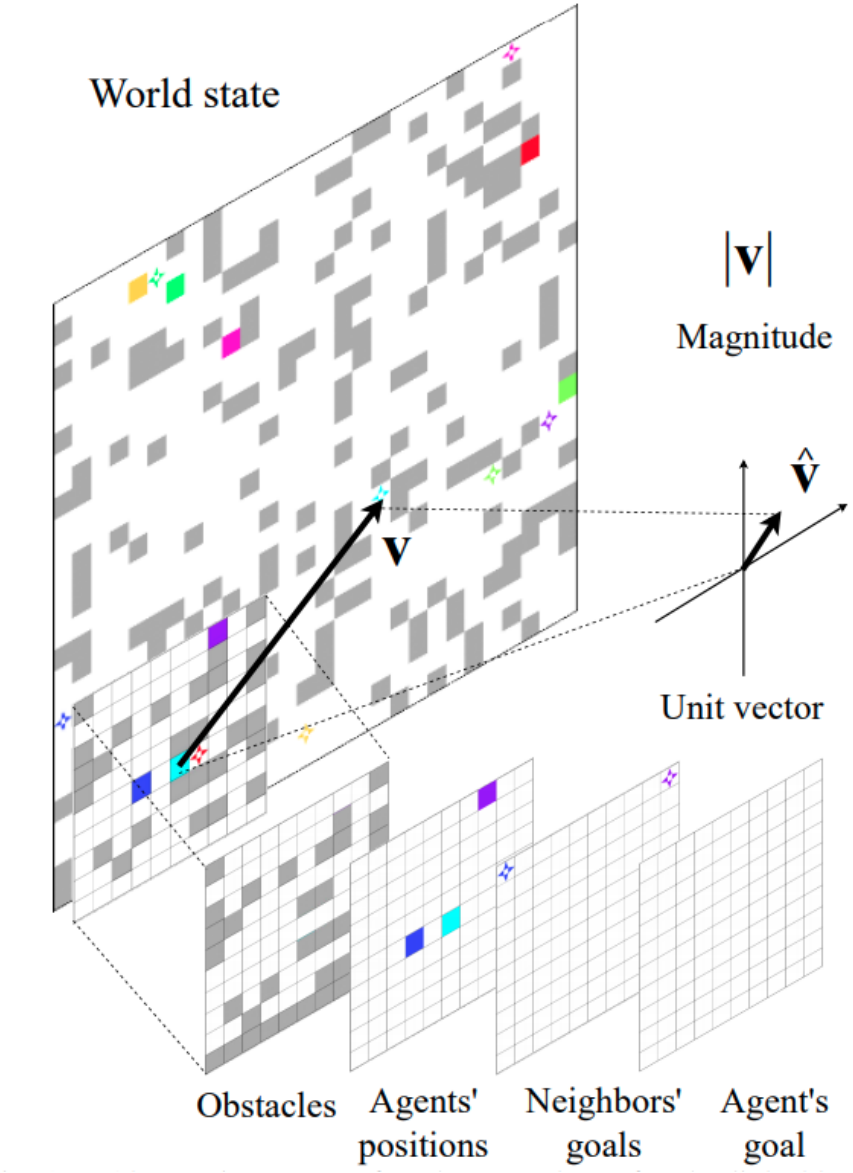
Related Work

- Imitation Learning & Reinforcement Learning
  - ***PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning***
  - *Learning Functionally Decomposed Hierarchies for Continuous Control Tasks with Path Planning*

- Reinforcement Learning **+**
  - *Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding*
    - *+ multi-step ahead tree search*
  - *MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments*
    - *+ evolutionary training approach*

PRIMAL

- **PRIMAL** is a novel framework for MAPF that:
  - Combining reinforcement learning and imitation learning
  - Training fully-decentralized but collaborative policies



SIMPLE REWARD STRUCTURE.

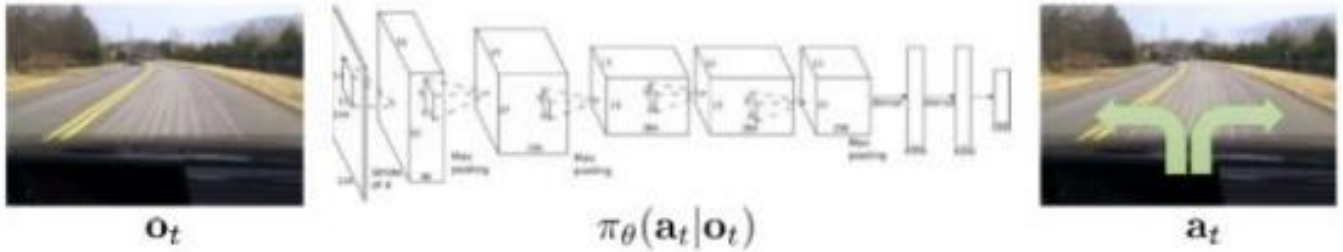
Action	Reward
Move [N/E/S/W]	-0.3
Agent Collision	-2.0
No Movement (on/off goal)	0.0 / -0.5
Finish Episode	+20.0

[2] Sartoretti, Guillaume, et al. "**PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning**." IEEE Robotics and Automation Letters 4.3 (2019): 2378-2385.

Behavioral Cloning

Behavioral Cloning，行为克隆，其实本质上就是一个标准的监督学习过程，如下图所示：

Imitation Learning



Policy Gradient

具体地说，策略梯度算法将策略建模成为$\pi_\theta(s,a)$，表示在s状态下选择a动作的概率，其中θ为参数。并且将负回报函数作为损失函数，应用梯度下降法将期望奖励最大化。定义为

$$J(\theta) = \sum_s d(s) \sum_a \pi_\theta(s,a)\mathcal{R}(s,a) \tag{1}$$

这样，(1)式对参数θ求梯度得到

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\mathcal{R}_{s,a} \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\mathcal{R}(s,a)\right]\end{aligned} \tag{2}$$

式子(2)的期望通过均值代替得到

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum \nabla_\theta \log \pi_\theta(s,a)\mathcal{R}(s,a) \tag{3}$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

Actor-Critic

从式子(3)来看蒙特卡洛策略梯度算法在策略梯度更新的过程中，考虑的是即时奖励v_t，而即时奖励具有较大噪声，为了得到更稳定的表现，可以使用长期回报来替代即时奖励。具体如式(4):

$$\nabla_\theta J(\theta) = \sum_{s \in S} d(s) \sum_{a \in A} \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a) G_{s,a} \tag{4}$$

其中$G_{s,a} = \sum \lambda^n \mathcal{R}_n$定义为(s,a)的长期回报, 根据Q函数的定义$Q(s, a) = \mathbb{E}[G_{s,a} | s, a]$,于是式子(4)使用长期回报期望$Q(s, a)$直接替代长期回报得到式(5)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(s, a) Q(s, a)] \tag{5}$$

于是根据(5)式我们可以得到$\Delta\theta = \nabla_\theta \log \pi_\theta(s, a) Q(s, a)$，用这种方式更新参数的就是Actor-Critic算法，简称AC算法。其中Critic就是$Q(s, a)$，本质上就是梯度权值，也可以说是评价梯度的重要性。

A2C & A3C

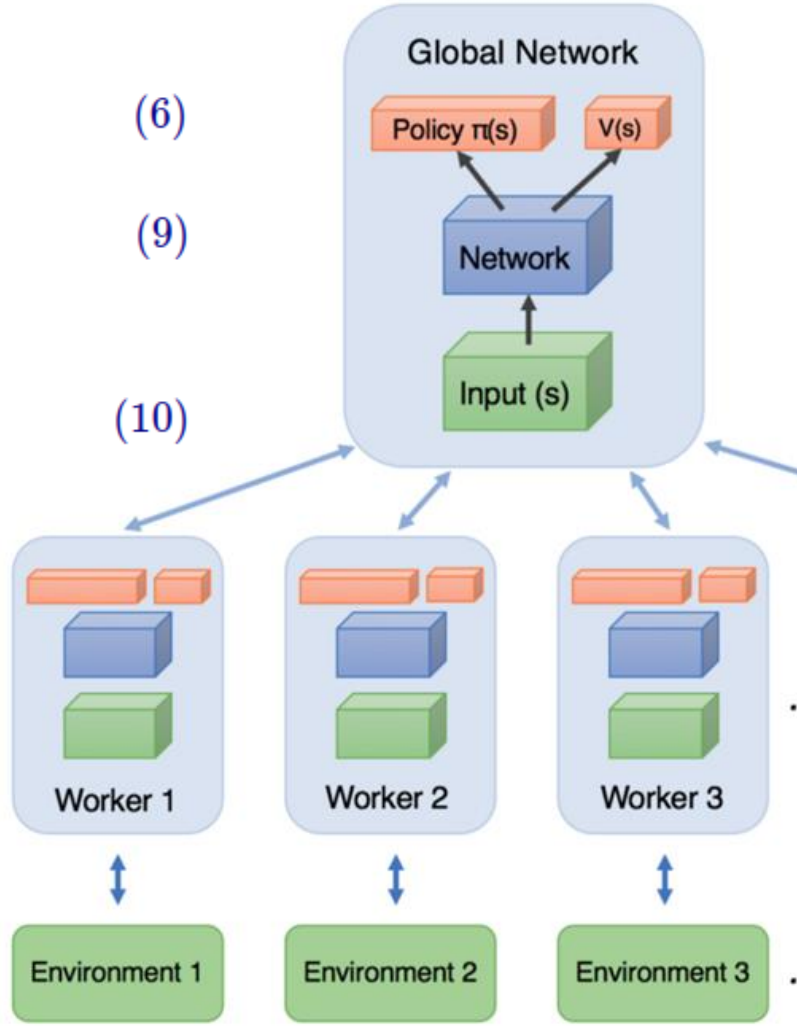
AC算法使用的Q函数是一个随机初始化的函数，需要在交互中学习逼近真正的$\hat{Q}$，这意味着我们在梯度更新中引入了噪声，或者说方差。为了解决这个问题，A2C引入了Baseline的概念。具体地说是通过在(5)式中引入一个Baseline函数$\mathcal{B}$得到(6)式子

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left\{\nabla_\theta \log \pi_\theta(s,a)\left[Q(s,a) - \mathcal{B}\right]\right\} \tag{6}$$

$$\mathcal{B}(s) = V(s) \tag{9}$$

最后，令$A(s,a) = Q(s,a) - V(s)$为优势函数(动作a相对平均表现的优势)，可以得到A2C算法的梯度公式

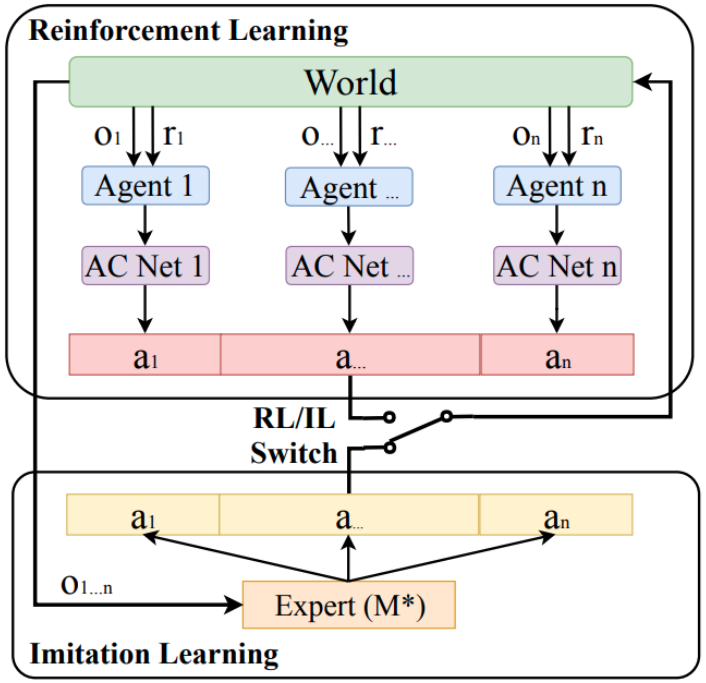
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A(s,a)\right] \tag{10}$$

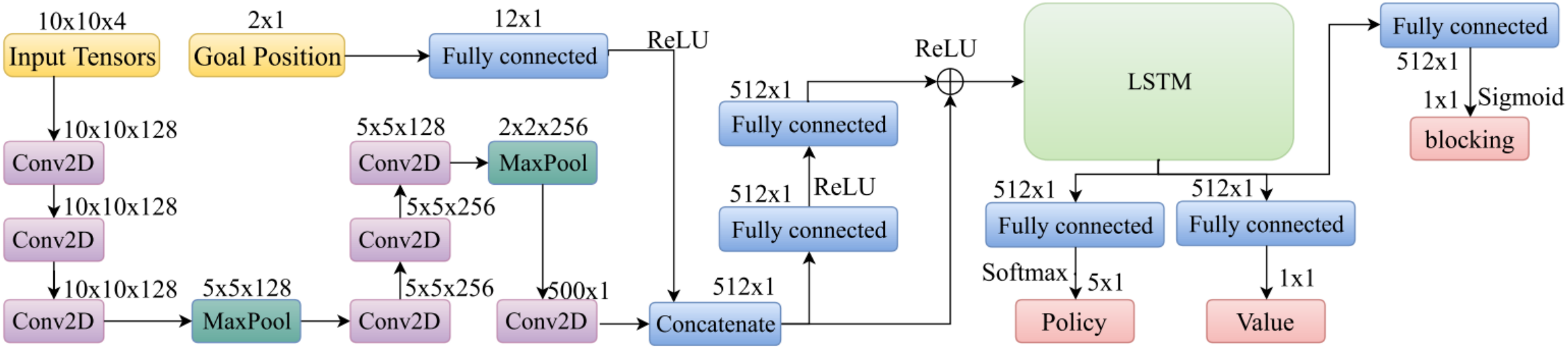


PRIMAL

- Training with ***blocking panelty***
- Selecting **RL(A3C)** or **IL(Behavior Cloning)** per episode

- **结果表明：**Decentralized method能够学会较好的个体policy，且因为policy共享，不存在agent之间的性能差异，目前在较大规模的任务中相较传统planning方法有**更好的到达率，但在step length方面有较大的差距**；

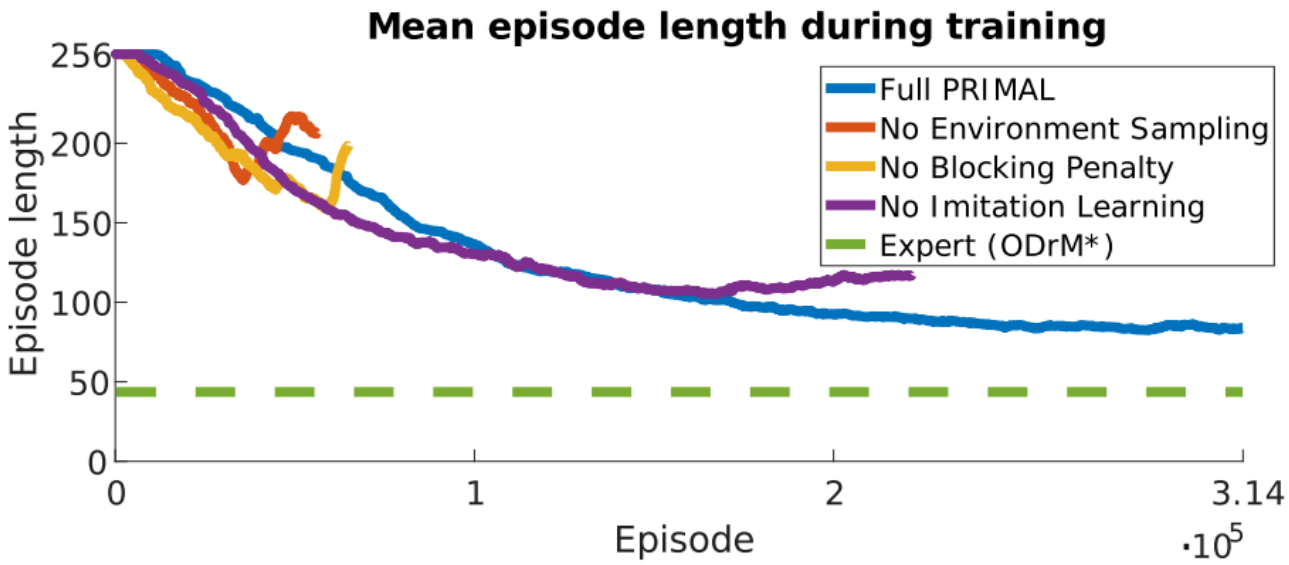




[2] Sartoretti, Guillaume, et al. "**PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning**." IEEE Robotics and Automation Letters 4.3 (2019): 2378-2385.

Ablation Study

- 为了检验PRIMAL中模仿学习部分对于policy训练的影响
  - 完整的PRIMAL（IM0.5——有0.5的概率采取模仿学习）
  - PRIMAL（IM0.1）
  - PRIMAL w/o IM

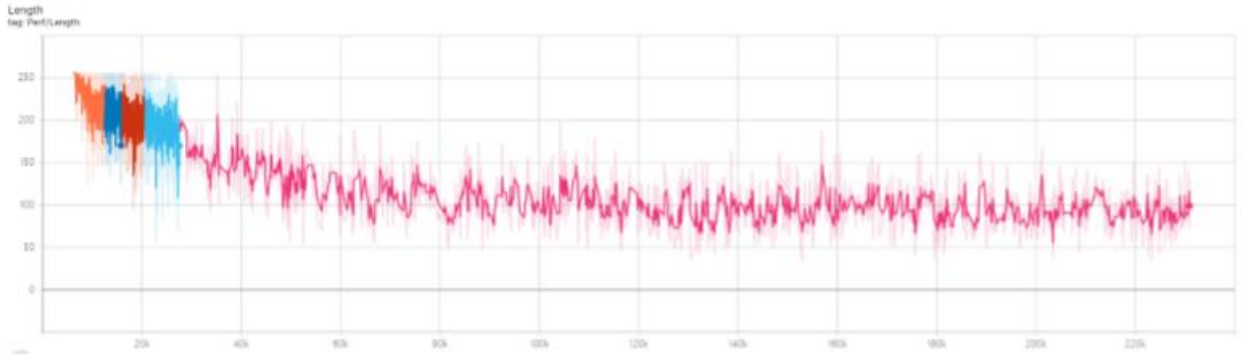


[2] Sartoretti, Guillaume, et al. "**PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning**." IEEE Robotics and Automation Letters 4.3 (2019): 2378-2385.

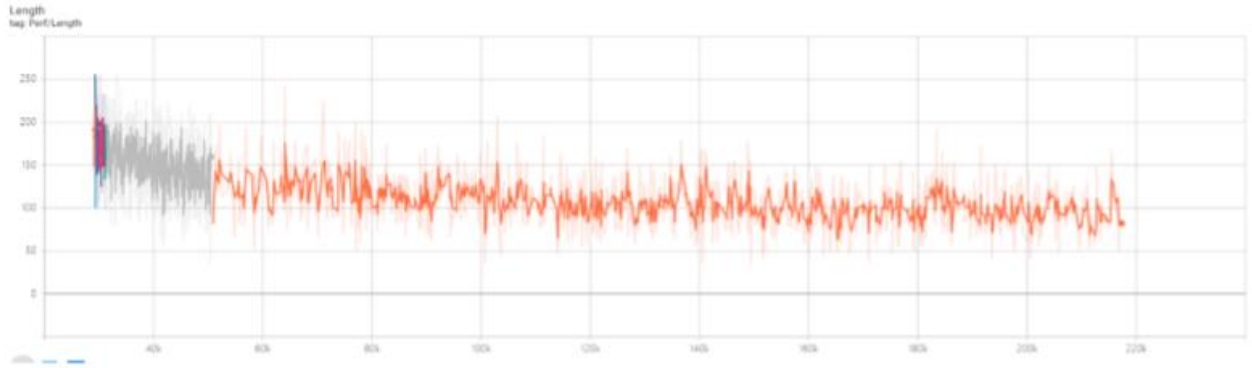
Result(Path Length)

- 分析:
  - Primal采取的behavior cloning方法仅考虑当前的观测信息，并没有包含诸如轨迹等时序信息——单步观测信息不足
  - 尽管planning方法能够解决long horizon的pathfinding问题，但是对于RL的policy而言，难以提供这样具有远见性的信息——在多智体系统中，这种问题在动态环境中更加严重
  - Goal-conditioned policy在解决观测视野外的pathfinding时难以保证其policy的efficiency——导致step length更长

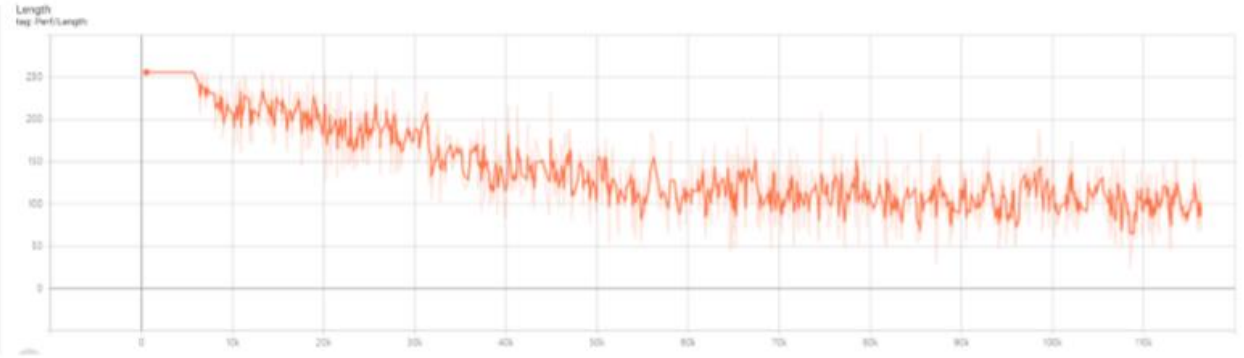
PRIMAL w/o IM



PRIMAL with IM(0.5)



PRIMAL with IM(0.1)



Related Work

- Imitation Learning & Reinforcement Learning
  - *PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning*
  - *Learning Functionally Decomposed Hierarchies for Continuous Control Tasks with Path Planning*

- Reinforcement Learning **+**
  - ***Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding***
    - *+ multi-step ahead tree search*
  - *MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments*
    - *+ evolutionary training approach*

MARL with Multi-step Ahead Tree Search

- **MDP Definition:**

From a decentralized point of view, MDP is defined as:

State $s_t^k \in \mathcal{S}$: The state of AGV k at time step t considers information in the 5×5 neighboring grids of the AGV k, each grid consists of the positions of obstacles and other AGVs, the goal information of current AGV and other observable AGVs.

Action $a_t^k \in \mathcal{A}$: Action space of each AGV is divided into two parts: move in four cardinal directions or stay still.

Reward Function $r_t^k \in \mathcal{R} \leftarrow \mathcal{S} \times \mathcal{A}$: The one step reward is designed as -0.2 and -0.4 when an AGV moves towards or away from its target, -0.5 for staying still, -20 for colliding with obstacles or the other AGVs, +40 for arriving at the target grid.

[3] Zhang, Yi, et al. "**Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding**." Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. 2020.

Proximal Policy Optimization

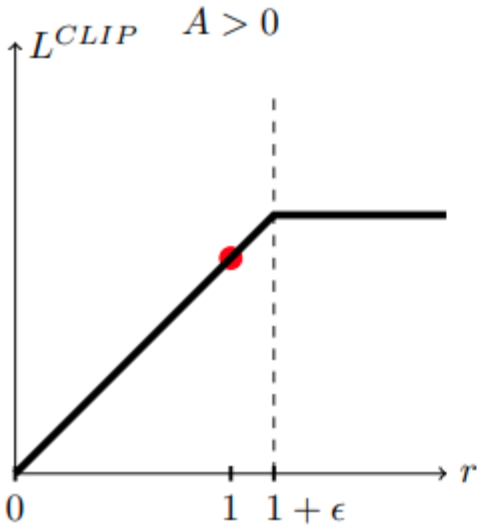
Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

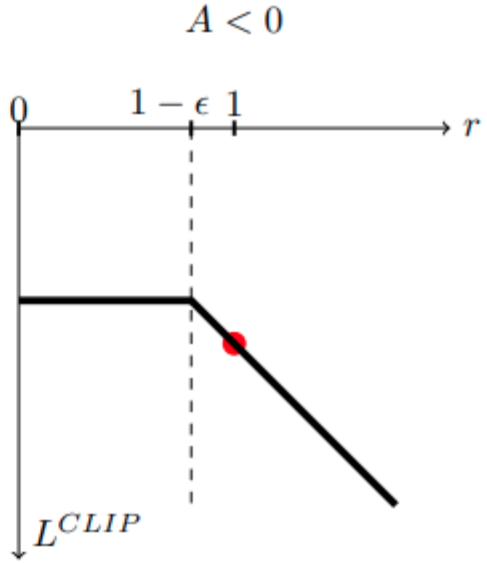
$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \tag{6}$$

The superscript CPI refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{7}$$





MARL with Multi-step Ahead Tree Search

- **MARL framework:**

Value Network. The state-value function is learned by minimizing the following loss function derived from Bellman equation:

$$L_{\theta_v} = \left(V_{\theta_v}\left(s_t^k\right) - V_{target}\left(s_{t+1}^k;\pi\right)\right)^2 \tag{1}$$

$$V_{target}\left(s_{t+1}^k;\pi\right) = r_t^k + V_{\theta_v'}\left(s_{t+1}^k\right) \tag{2}$$

Policy Network. In this paper, we use the same objective in the actor-critic algorithm from Proximal Policy Optimization [11]:

$$L_\theta = \mathbb{E}_t\left[\min\left(r_t\left(\theta\right), clip\left(r_t\left(\theta\right), 1-\epsilon, 1+\epsilon\right)\right)A_t\right] \tag{3}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

$$A_t = -V_{\theta_v}\left(s_t\right) + \sum_{\tau=t}^{T-1} r_\tau\, \gamma^{\tau-t} + \gamma^{T-t} V_{\theta_v}\left(s_T\right) \tag{4}$$

where θ, θ_v denotes the parameters of policy network and value network.

- **Searching:**

$$UCB_l = \frac{Q_l}{N_l} + C_p \frac{p(s_t, a_t)}{1+N_l}$$

C_p表示探索率

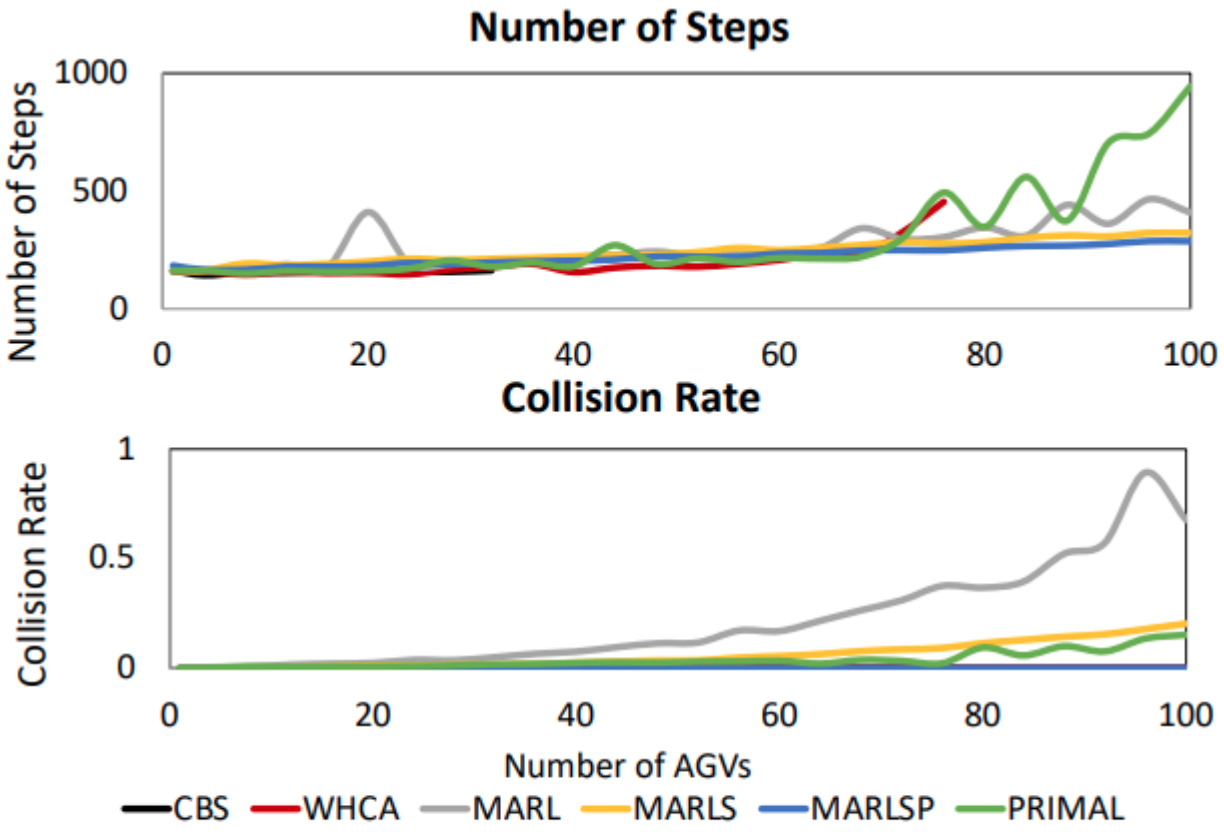
Algorithm 1: Multi-agent Training with Searching

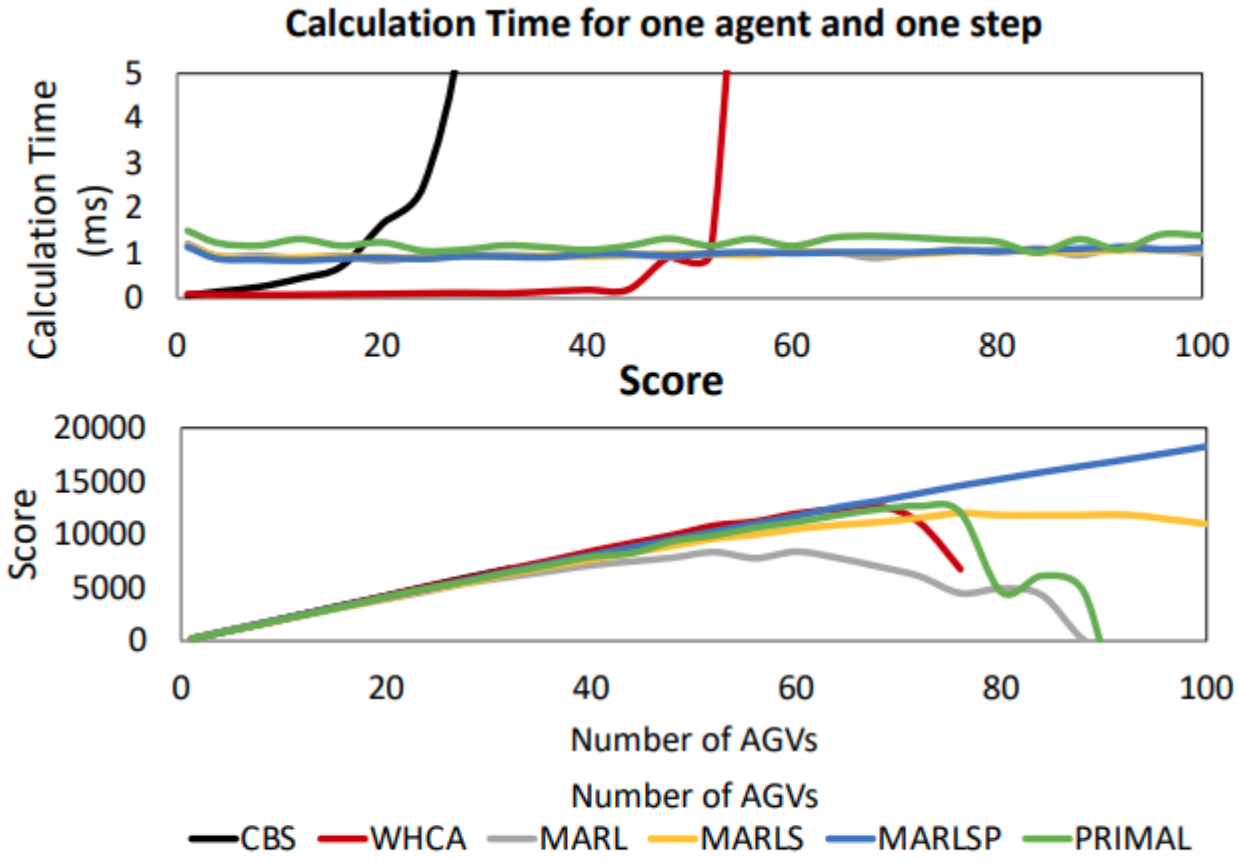
for $m = 1$ to $N_{episode}$ **do**
  Reset environment and get initial state
  Stage 1: Sampling
  while $t < T$ **do**
    for k=1 to K **do**
      Calculate probability weights $\pi(a_t^k|s_t^k)$
      Create root node v_0 with $Q_0 = 0, N_0 = 0$
      while $n < N$ **do**
        if any node $v_l \in V$ is not fully-expanded **then**
          Expand $v_0 \rightarrow v_l$ by choosing an action sequence $\{a_t, a_{t+1}, \ldots a_{t+\tau}\}$ according to policy π
          Estimate $s_{t+\tau}$ and $r_{t+\tau}$ by simulating actions
          Add new child v_l to V with
        else
          Sample a node v_l with the maximum UCB
          Backward propagation of parent and ancestor nodes by $Q_l = Q_l + q_l$, $N_l = N_l + 1$
        end if
      end while
      Choose v_0 with the largest Q_l/N_l and its action a_t^k
      Execute $\mathbf{a}_t^k$ and observe reward $\mathbf{r}_t^k$, next state $\mathbf{s}_{t+1}^k$
    end for
    Store the transitions $(s_t^k, a_t^k, r_t^k, s_{t+1}^k)$ into M
  end while
  Stage 2: Learning
  Sample a batch of experience: s_t^k, $V_{target}(s_{t+1}^k;\pi)$
  Update by minimizing the value loss Eq.(1) over the batch
  Update as $\theta \leftarrow \theta + \nabla_\theta L_\theta$ according to Eq.(3)
end for

MARL with Multi-step Ahead Tree Search

多智体人工智能实验室
Multi-Agent Artificial Intelligence Laboratory

- **Comparison results for AGV path finding in the 30 × 30 grid world:**





[3] Zhang, Yi, et al. "**Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding**." Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. 2020.

Related Work

- Imitation Learning & Reinforcement Learning
  - *PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning*
  - *Learning Functionally Decomposed Hierarchies for Continuous Control Tasks with Path Planning*

- Reinforcement Learning **+**
  - *Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding*
    - *+ multi-step ahead tree search*
  - ***MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments***
    - *+ evolutionary training approach*

MARL with Evolutionary Training

- An image-based representation which improves agents' robustness to handle different types of dynamic obstacles
- Decomposing a difficult long-range planning problem into multiple easier waypoint-conditioned planning tasks with the help of mature global planners
- An evolutionary multi-agent reinforcement learning approach that gradually eliminate low-performance policies during training to increase training efficiency and performance

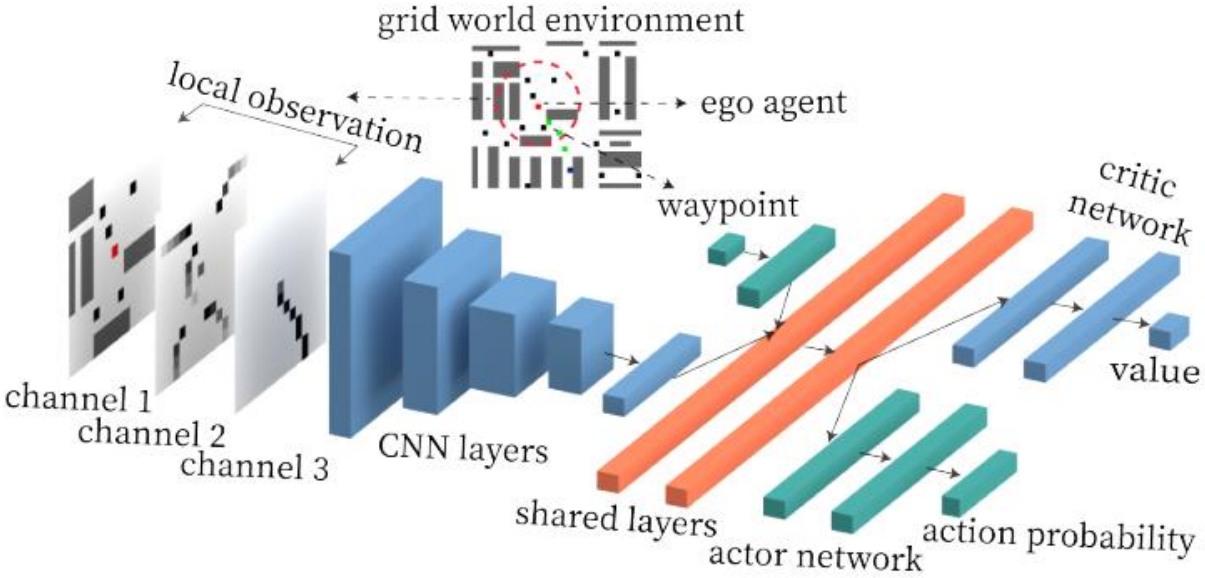


TABLE I: Reward Design

Reward	Value
step penalty r_s	-0.1 (move) or -0.5 (wait)
collision penalty r_c	-5
oscillation penalty r_o	-0.3
off-route penalty r_f	$-\min_{\boldsymbol{p}\in\mathcal{S}} \|\|\boldsymbol{p}_a - \boldsymbol{p}\|\|_2$
goal-reaching reward r_g	30

[4] Liu, Zuxin, et al. "MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments." arXiv e-prints (2020): arXiv-2007.

MARL with Evolutionary Training

- Algorithm & Result

TABLE II: Comparison of success rate over different experiment settings

Environment Setting			Success Rate				
map size	agent	dynamic obstacle	MAPPER	MAPPER w/o traj	MAPPER w/o guid	PRIMAL*	LRA*
20x20	15	10	**1.0**	0.971	0.877	0.964	0.996
20x20	35	30	**1.0**	0.961	0.836	0.980	0.999
20x20	45	30	**0.999**	0.854	0.607	0.971	0.997
60x65	70	100	**1.0**	0.256	0.516	0.352	**1.0**
60x65	130	140	**1.0**	0.473	0.221	0.404	0.992
120x130	150	40	**0.997**	0.324	0.211	0.389	0.994

Algorithm 1 Multi-Agent Evolutionary Training Approach

Require: Agents number N; discount factor γ; evolution interval K; evolution rate η;

Initialize agents' model weights $\Theta = \{\Theta_1, ..., \Theta_N\}$
repeat
  Set accumulated reward $R_1^{(k)}, ..., R_N^{(k)} = 0$
  // update model parameters via A2C algorithm
  for $k = 1, ..., K$ **do**
    for each agent i **do**
      Executing the current policy π_{Θ_i} for T timesteps, collecting action, observation and reward $\{a_i^t, \boldsymbol{o}_i^t, r_i^t\}$, where $t \in [0, T]$
      Compute return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$
      Estimate advantage $\hat{A}_i = R - V^{\pi_{\Theta_i}}(\boldsymbol{o}_i)$
      Compute gradients $\nabla_{\Theta_i} J = \mathbb{E}[\nabla_{\Theta_i} \log \pi_{\Theta_i} \hat{A}_i]$
      Update Θ_i based on gradients $\nabla_{\Theta_i} J$
    end for
    $R_i^{(k)} = R_i^{(k)} + R_i$
  end for
  Normalize accumulated reward to get $\bar{R}_1^{(k)}, ..., \bar{R}_N^{(k)}$
  Find maximum reward $\bar{R}_j^{(k)}$ with agent index j
  // Evolutionary selection
  for each agent i **do**
    Sample m from uniform distribution between $[0, 1]$
    Compute evolution probability $p_i = 1 - \frac{\exp(\eta \bar{R}_i^{(k)})}{\exp(\eta \bar{R}_j^{(k)})}$
    if $m < p_i$ **then**
      $\Theta_i \leftarrow \Theta_j$
    end if
  end for
until converged

[4] Liu, Zuxin, et al. "MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments." arXiv e-prints (2020): arXiv-2007.

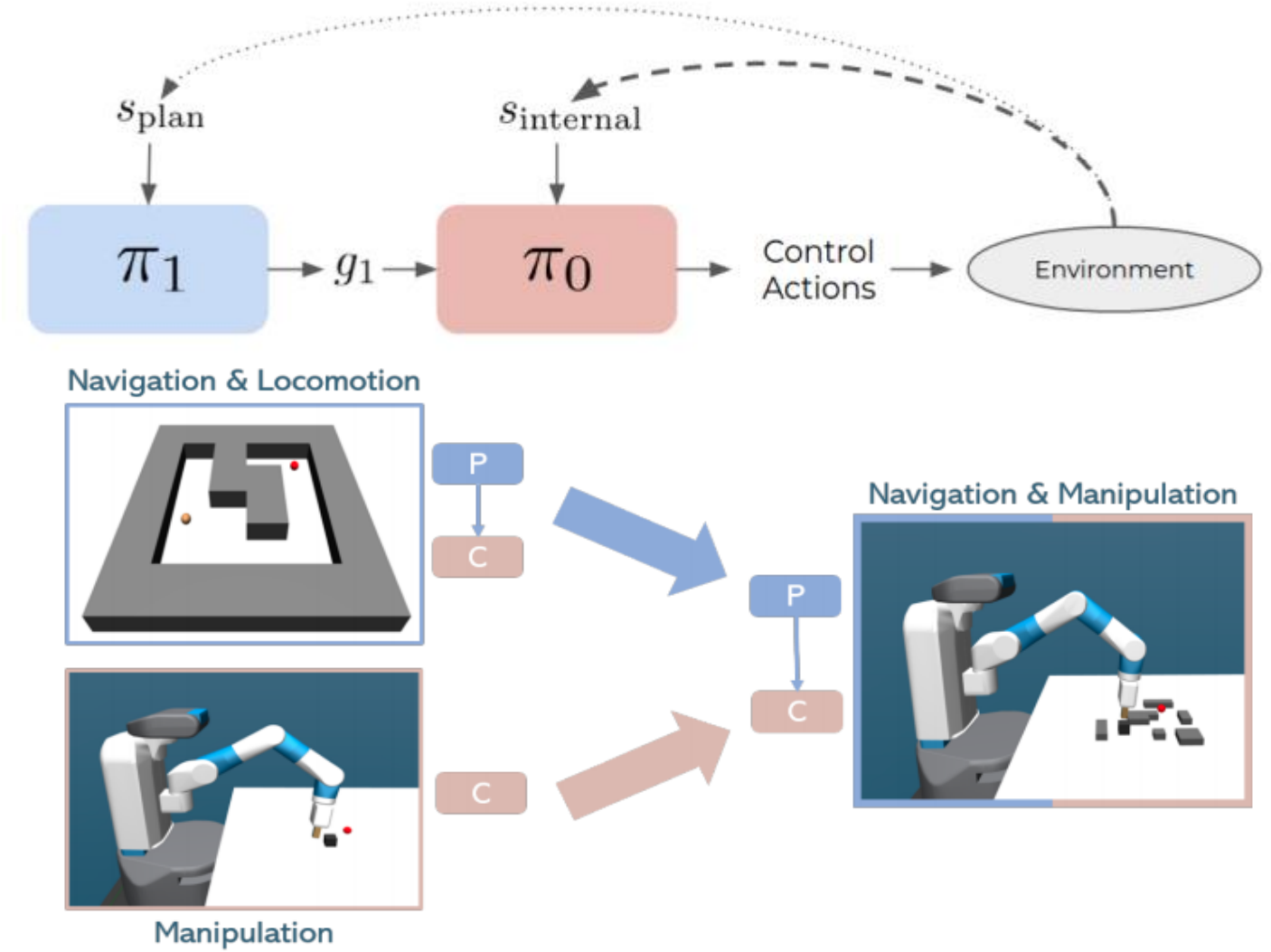
Related Work

- Imitation Learning & Reinforcement Learning
  - *PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning*
  - ***Learning Functionally Decomposed Hierarchies for Continuous Control Tasks with Path Planning***

- Reinforcement Learning **+**
  - *Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding*
    - *+ multi-step ahead tree search*
  - *MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments*
    - *+ evolutionary training approach*

Hierarchical Decompositional Reinforcement Learning

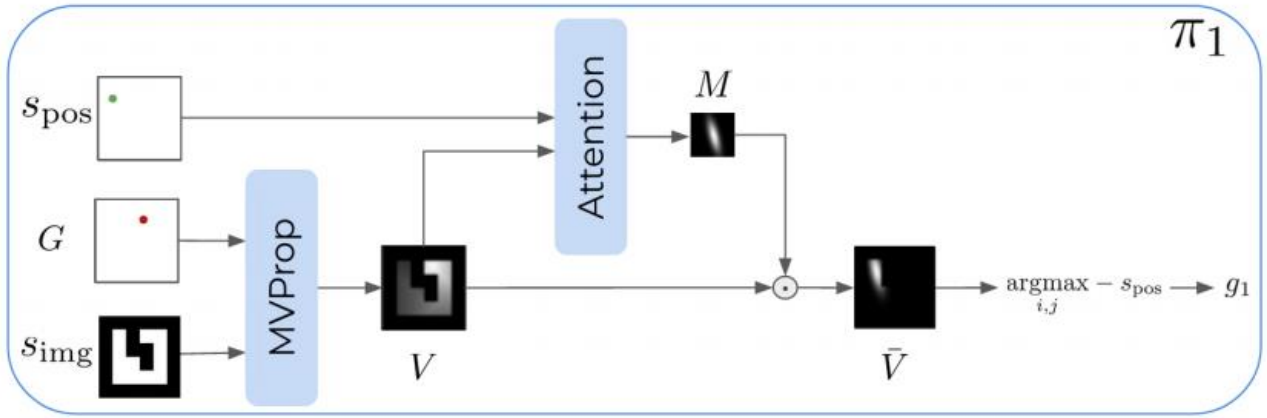
- Decomposing the planning policy of single agent into:
  - Planning Layer π_1
  - Control Layer π_0



[5] Jendele, Lukas, et al. "**Learning Functionally Decomposed Hierarchies for Continuous Control Tasks**." arXiv preprint arXiv:2002.05954 (2020).

Hierarchical Decompositional Reinforcement Learning

- Planning Layer(DQN)



- Control Layer(DDPG)
  - Learning a goal-conditioned policy

The Q-function (critic) and the policy π (actor) are approximated by using neural networks with parameters θ^Q and θ^π. The objective for θ^Q minimizes the loss:

$$L(\theta^Q) = \mathbb{E}_{\mathcal{M}}\left[\left(Q(s_t, g_t, a_t; \theta^Q) - y_t\right)^2\right], \text{where} \quad (1)$$

$$y_t = r(s_t, g_t, a_t) + \gamma Q(s_{t+1}, g_{t+1}, a_{t+1}; \theta^Q).$$

The policy parameters θ^π are trained to maximize the Q-value:

$$L(\theta^\pi) = \mathbb{E}_\pi\left[Q(s_t, g_t, a_t; \theta^Q) | s_t, g_t, a_t = \pi(s_t, g_t; \theta^\pi)\right] \quad (2)$$

[5] Jendele, Lukas, et al. "**Learning Functionally Decomposed Hierarchies for Continuous Control Tasks**." arXiv preprint arXiv:2002.05954 (2020).

改进思路——解决模仿学习低效的问题

- Step1:为observation添加历史轨迹信息
- **Step 2:将原本end-to-end的goal-conditioned policy拆分为hierarchical框架：**
  - **上层planning layer负责给出当前agent要到达的下一个waypoint；**
  - **下层的control layer负责学习一个goal-conditioned policy，到达下一个waypoint。**
- Step 3:改进policy的模仿学习method——将behavior cloning替换为goalGAIL
- Step 4:将模仿的算法替换为其他decentralized传统planning算法（odrm* 为centralized，训练耗时大，且难以求解问题规模较大的问题）

Thank you

- End