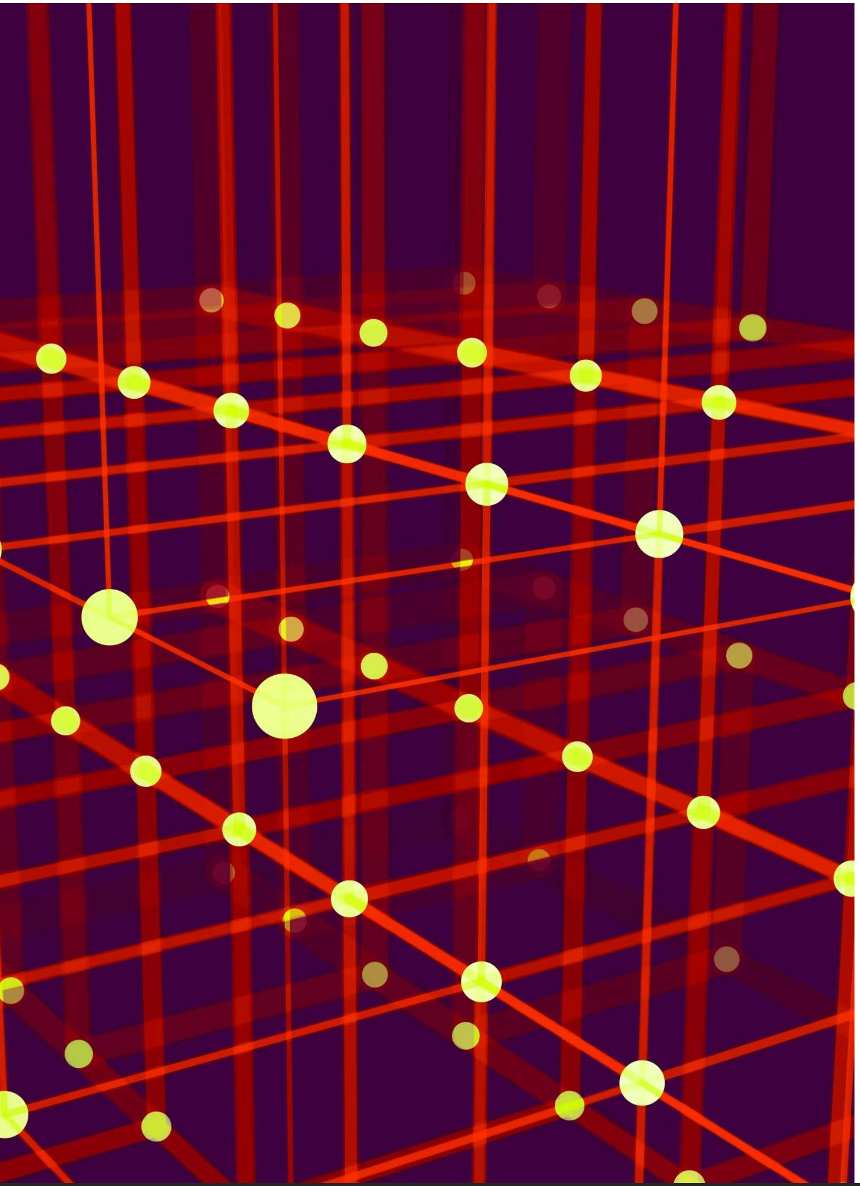
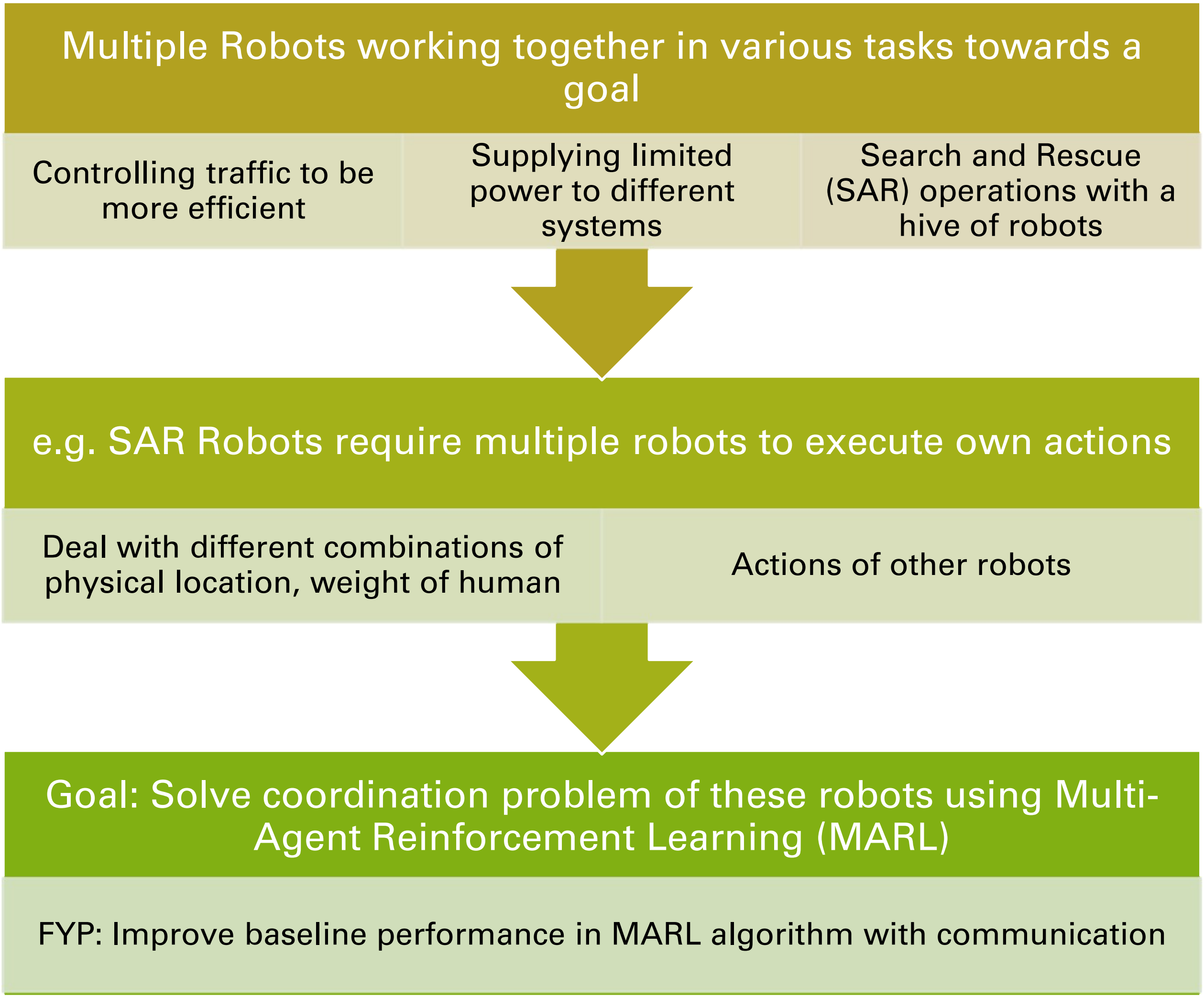


Multi-agent strategy learning in reinforcement learning for coordinated problem solving

WIRA AZMOON AHMAD

SUPERVISOR: DR AKSHAY NARAYAN

Motivation



Multiple Robots working together in various tasks towards a goal

Controlling traffic to be more efficient

Supplying limited power to different systems

Search and Rescue (SAR) operations with a hive of robots

e.g. SAR Robots require multiple robots to execute own actions

Deal with different combinations of physical location, weight of human

Actions of other robots

Goal: Solve coordination problem of these robots using Multi-Agent Reinforcement Learning (MARL)

FYP: Improve baseline performance in MARL algorithm with communication

Reinforcement Learning

Reinforcement Learning (RL) aims to learn actions to take

Supported by reward system informing how good actions are

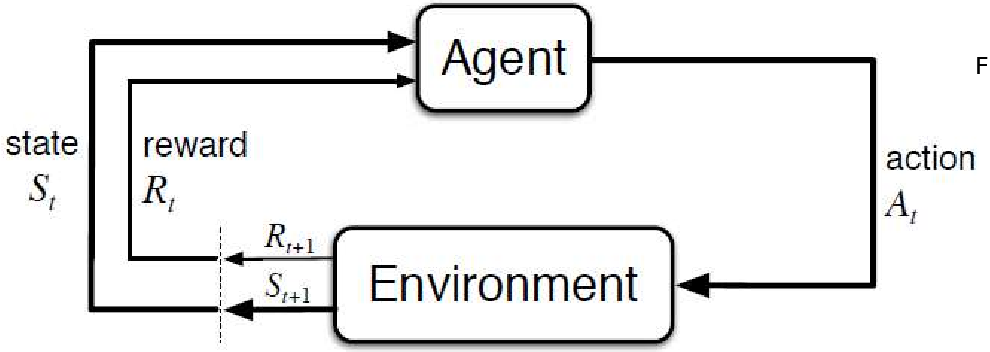


Figure from Sutton & Barto (2018)

Multi-Agent RL involves multiple agents trying to achieve goal of system

Single-Agent RL

State Space	S	
Action Space	A	
Reward Function	$R(s_t, a_t, s_{t+1}) \in \mathbb{R}$	where $s_t, s_{t+1} \in S, a_t \in A$
Transition Model	$P(s_t, a_t, s_{t+1}) \sim \Pr(s_{t+1} \mid s_t, a_t)$	

Markov Decision Process (MDP) framework

Markov assumption – Next state only dependent on current state and action

Goal: Obtain optimal policy $\pi: S \rightarrow A$ or $\pi(s_t) \sim \Pr(a \mid s_t)$

Value Functions

Value at each state

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t|\, a_t \sim \pi(s_t), s_0 = s]$$

Q-value at each state-action pair

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[R_t|\, a_t \sim \pi(s_t), s_0 = s, a_0 = a]$$

with the discounted reward R_t, using a discount factor $\gamma \in [0,1)$

$$R_t = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$$

Optimal Policy

Suppose we have optimal policy π^*. Then

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s, a)$$

$$\pi^*(s) = \operatorname*{argmax}_a Q^{\pi^*}(s, a)$$

Value Approximation

Model $V(s)$ or $Q(s,a)$, then policy can be greedily obtained

$$\hat{\pi}(s) = \underset{a}{\operatorname{argmax}}\, Q^{\hat{\pi}}(s,a)$$

Deep RL now commonly used to approximate functions, such as Deep Q-Networks (DQN) (Mnih, et al., 2015)

$$L(\theta) = \mathbb{E}_{s,a\sim\mathcal{D}}\left[\left(y - Q(s,a;\theta)\right)^2\right]$$

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$$

Policy Approximation

Directly parameterize policy

Define expected value of policy π as

$$J(\theta) = \mathbb{E}_{\tau}[R]$$

Steps taken in its gradient (Sutton, McAllester, Singh, & Mansour, 1999)

$$\nabla J(\theta) = \mathbb{E}_{s,a\sim\mathcal{D}}[\nabla_{\theta} \log \pi_{\theta}(a|s)\, Q^{\pi}(s,a)]$$

Multi-Agent RL

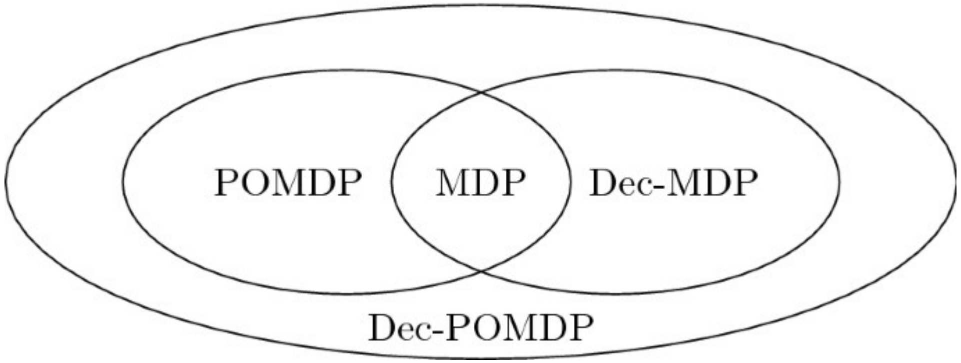
Many agents in single environment

Common to model environment as partially observable

Decentralized Partially Observable MDP (Dec-POMDP)

- Decentralized actions and observations per agent
- Observations giving a belief over the state

n agents



Multi-Agent RL

Number of agents	n	1
State Space	S	S
Action Space	$A = A_1 \times \cdots \times A_n$	A
Reward Function	$R_i(s, \boldsymbol{a}) \in \mathbb{R}$	$R(s, a) \in \mathbb{R}$
Transition Model	$P(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t+1}}) \sim \Pr(\boldsymbol{s_{t+1}} \vert \boldsymbol{s_t}, \boldsymbol{a_t})$	$P(s_t, a_t, s_{t+1}) \sim \Pr(s_{t+1} \vert s_t, a_t)$
Observations	$O = O_1 \times \cdots \times O_n$	

Each agent i learns policy $\pi_i(a_i|o_i) \sim \Pr(a_i|o_i)$

DRQN

Capable systems to support Deep Learning

Deep Q-Networks (DQN) model complicated Value Functions

- Approximate $Q(s, a)$
- Assumes accurate model of state s
- But $Q(s, a) \neq Q(o, a)$

Deep Recurrent Q-Networks (DRQN) better model states of partially observable environments (Hausknecht & Stone, 2015)

MARL Challenges

Non-stationarity

- $\Pr(s' | s, a_i, \pi_1, \ldots, \pi_n) \neq \Pr(s' | s, a_i, \pi_1', \ldots, \pi_n')$ if any $\pi_i \neq \pi_i'$

Noise and Variance

- unstable training due to rewards depending on other agents’ actions

Literature Review

Category	Description	Examples
Independent Learners	Other agents' actions are part of environment	IQL, IA2C, IPPO
Fully Observable Critic	Global critic to guide local agent actors	MADDPG, MAA2C, MAPPO
Value Function Factorization	Determine share of value function for each agent	VDN, QMIX
Consensus Learn to Communicate	Factor in communication costs	Diff-DAC DIAL, SAF

Oroojlooy & Hajinezhad (2019)

Algorithms

Evaluating algorithms for baselines

Centralized Training, Decentralized Execution (CTDE)

Train with all information available centrally

Execute actions independently

Independent Learners

Treat other agents' actions as part of environment

Each agent trained independently

Problem: *Non-stationarity*

$$\Pr(s' | s, a_i, \pi_1, \ldots, \pi_n) \neq \Pr(s' | s, a_i, \pi_1', \ldots, \pi_n')$$

if any $\pi_i \neq \pi_i'$

Markov assumption violated

Can still achieve good performance like IQL in Atari games (Mnih, et al., 2015)

Independent Learner Algorithms

IA2C (Mnih, et al., 2016)

- Objective Function $\nabla J(\theta) = \mathbb{E}_{s,a\sim\mathcal{D}}[\nabla_{\theta'} \log \pi(a|s;\theta')\, A(s,a;\theta,\theta_v)]$,
- Advantage Function $A(s_t, a_t; \theta, \theta_v) = R_t - V(s_t; \theta_v)$
- Independent actor $\pi_i(a_i|s_i;\theta_i)$ and critic networks $V_i(s_i;\theta_{vi})$

IPPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017)

- Objective Function $L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$
- Independent actor $\pi_i(a_i|s_i;\theta_i)$ and critic networks $V_i(s_i;\theta_{vi})$

Fully Observable Critic

Critic that sees all observations and actions of all agents

Given all actions, next state independent of policy

$$\Pr(s'|s, a_1, \dots, a_n, \pi_1, \dots, \pi_n) = \Pr(s'|s, a_1, \dots, a_n)$$

Actor-critic models

Typically,

- 1 actor per agent π_i
- 1 centralized critic V^π or Q^π

Fully Observable Critic Algorithms

MAA2C (Mnih, et al., 2016)

- Like IA2C
- Joint state-value function as centralized global critic $V(\boldsymbol{s};\theta_v)$

MAPPO (Yu, et al., 2021)

- Like IPPO
- Joint state-value function as centralized global critic $V(\boldsymbol{s};\theta_v)$

MADDPG (Lowe, et al., 2017)

- Objective Function $\nabla_{\mathrm{\theta_i}} J(\theta_i) = \mathbb{E}_{\boldsymbol{o},\boldsymbol{a}\sim\mathcal{D}}\left[\nabla_{\theta_i}\pi_i(a_i|o_i)\nabla_{a_i}Q_i^{\pi}(\boldsymbol{o},\boldsymbol{a}) \mid a_i = \pi_i(o_i)\right]$
- Loss Function $L(\theta_i) = \mathbb{E}_{\boldsymbol{o},\boldsymbol{a},\boldsymbol{r},\boldsymbol{o}'}\left[\left(y - Q_i(\boldsymbol{o},\boldsymbol{a};\theta_i)\right)^2\right]$
- Each agent i has actor $\pi_i(o_i;\theta_i)$ with centralized critic Q_i^{π}

Value Function Factorization

Problem: Some agents *lazy* and do not learn good policy

- An agent having a good policy may discourage exploration from other agents

Get share of reward from each agent

E.g. VDN (Sunehag, et al., 2018),

$$Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{n} Q_i(\tau_i, a_i; \theta_i)$$

where $\boldsymbol{\tau} \in \mathcal{T} \equiv (\mathcal{O} \times \mathcal{A})^*$ is a combined observation-action history

Value Function Factorization Algorithm

QMIX (Rashid, et al., 2018)

- Extend VDN by considering $\operatorname*{argmax}_{\boldsymbol{a}} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \operatorname*{argmax}_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \operatorname*{argmax}_{a_n} Q_n(\tau_n, a_n) \end{pmatrix}$

- Loosen constraint to $\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \forall i \in [n]$

Literature Review

Category	Description	Examples
Independent Learners	Other agents’ actions are part of environment	IQL, IA2C, IPPO
Fully Observable Critic	Global critic to guide local agent actors	MADDPG, MAA2C, MAPPO
Value Function Factorization	Determine share of value function for each agent	VDN, QMIX
Consensus Learn to Communicate	Factor in communication costs	Diff-DAC DIAL, SAF

Oroojlooy & Hajinezhad (2019)

Learn To Communicate

Communication-action / Message passing, on top of action

MARL with communication = “Comm-MARL”

Learn what to send

Use message to make better decisions

Typically have messages from message space, $\boldsymbol{m} \in \mathcal{M}$

Learn To Communicate Algorithms

DIAL (Foerster, Assael, de Freitas, & Whiteson, 2016)

- Q-network models $Q\left(o_t^i, h_{t-1}^i, m_{t-1}^{-i}, a_t^i, m_t^i\right)$
- Takes in hidden state h_{t-1}^i
- Messages m_{t-1}^{-i}, m_t^i

SAF (Liu et al., 2023)

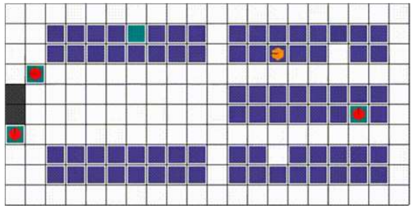
- Shared Knowledge Source (KS), and cross-attention (CA) (Chen, Fan, & Panda, 2021)
- Generate messages from observation, $\boldsymbol{m}_{\boldsymbol{t}}' = g_\theta(\boldsymbol{o}_{\boldsymbol{t}})$
- Write into KS, limited slots $L \leq n$ agents, $F_t \leftarrow f(\boldsymbol{m}_{\boldsymbol{t}}')$
- Read from KS, $\boldsymbol{m}_{\boldsymbol{t}} = h(F_t)$
- Sent to centralized critic as $\boldsymbol{s}_{\boldsymbol{t}}^{\boldsymbol{SAF}} = g_\phi(\boldsymbol{s}_{\boldsymbol{t}}, \boldsymbol{m}_{\boldsymbol{t}})$

Literature Review

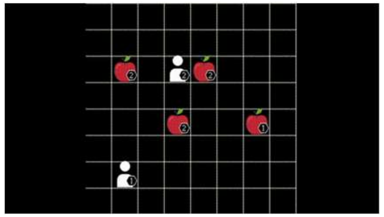
Category	Description	Examples
Independent Learners	Other agents' actions are part of environment	IQL, IA2C, IPPO
Fully Observable Critic	Global critic to guide local agent actors	MADDPG, MAA2C, MAPPO
Value Function Factorization	Determine share of value function for each agent	VDN, QMIX
Consensus Learn to Communicate	Factor in communication costs	Diff-DAC DIAL, SAF

Oroojlooy & Hajinezhad (2019)

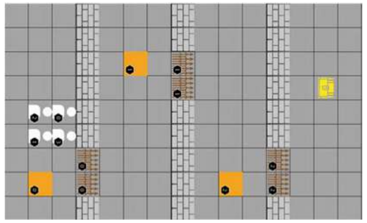
Environments



RWARE



LBF



PP

Commonly used to evaluate RL algorithms

EPyMARL framework (Papoudakis, Christianos, Schäfer, & Albrecht, 2021)

Environment	Description	Actions	Rewards
Robotic Warehouse (RWARE)	Grid-world warehouse to deliver shelves to workstations	$\{Turn\ Left, Turn\ Right, Forward, Load, Unload\}$	1 iff successfully delivered
Level-Based Foraging (LBF)	Agents collect levelled food items	$\{None, North, South, East, West, Load\}$	Each time food is loaded
PressurePlate (PP)	Reach treasure chest	$\{Up, Down, Left, Right, No{-}op\}$	Independent reward based on assigned plate

Environment Task Names

Environment	Environment Task Name	Name Format
RWARE	rware-tiny-{Xag}-v1	- tiny is the map size chosen, corresponding to one row, and three columns of groups of shelves. - Each group of shelves consists of 16 shelves arranged in an 8×2 grid. - Xag refers to the number of agents (i.e. 4ag means 4 agents) in the environment. - Each agent can only observe a 3×3 grid centered around themselves.
LBF	Foraging-{grid_size}x{grid_size}-{n_agents}p-{food}-v2	- grid_size is the size of both the horizontal and vertical lengths of the environment. - n_agents is the number of agents. - food is the number of food items scattered in the environment.
PressurePlate	pressureplate-linear-{n}p-v0	- n refers to the number of agents.

Benchmark

Benchmark from EPyMARL (Papoudakis, Christianos, Schäfer, & Albrecht, 2021)

Maximum returns with 95% confidence intervals over 5 seeds

Tasks\Algs.	IPPO	MAPPO	MAA2C	QMIX
RWARE Tiny 4p	31.82 ± 10.71	**49.42 ± 1.22**	32.50 ± 9.79	0.30 ± 0.19
LBF 8x8-2p-2f-c	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.96 ± 0.07

Heterogeneity and Coordination

Liu et al. (2023) introduce heterogeneity and coordination

Heterogeneity: Quantitative measure of the variation of environment dynamics

- If environment has K transition functions, K is the heterogeneity

Coordination: Quantitative measure of coordination required among agents to solve tasks

- If environment requires minimally c agents to achieve any subgoal, c is the coordination

Environments mentioned have low heterogeneity (i.e. homogenous) and low coordination

- LBF may need $c > 1$ agent for some foods
- Otherwise, $K = 1, c = 1$

Parameter Sharing

Deep MARL algorithms may use parameter sharing

All agents use the same parameters

Each actor and critic network receive identity of agent to learn different behaviour

- E.g. actor $\pi_i(a_i|s_i;\theta_i)$ and critic networks $V_i(s_i;\theta_{vi})$
- Now common $\theta_i = \theta$ and $\theta_{vi} = \theta_v$

Shown to improve performance of MARL algorithms

- (Papoudakis, Christianos, Schäfer, & Albrecht, 2021)

Experiment Setup

Python 3.8, PyTorch 1.13, OpenAI Gym 0.21.0

	MAPPO	MAA2C	QMIX
Learning rate α	0.0005		
Discount factor γ	0.99		
Time steps t Tested every $t = 50000$	$t_{max} = 20050000$		$t_{max} = 2050000$
Reward Standardisation	False	True	

SAF implemented on MAPPO and MAA2C sets no. of KS slots $L = 2$

Graphs show test returns over training episodes

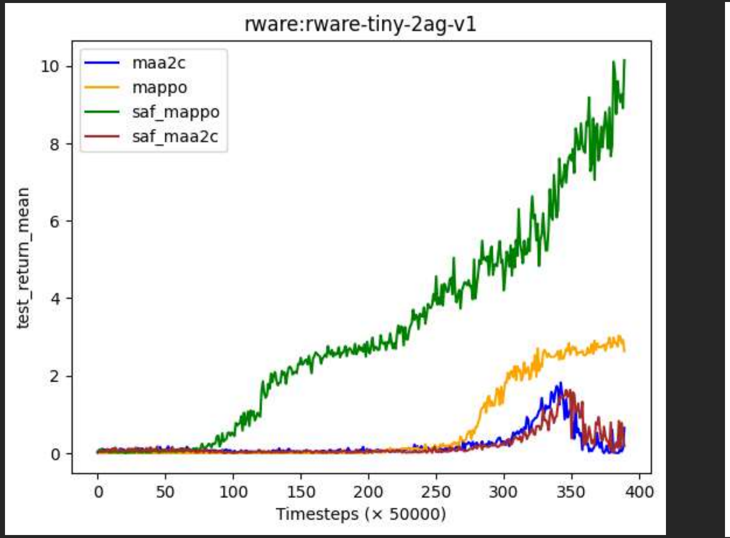
Results (RWARE)

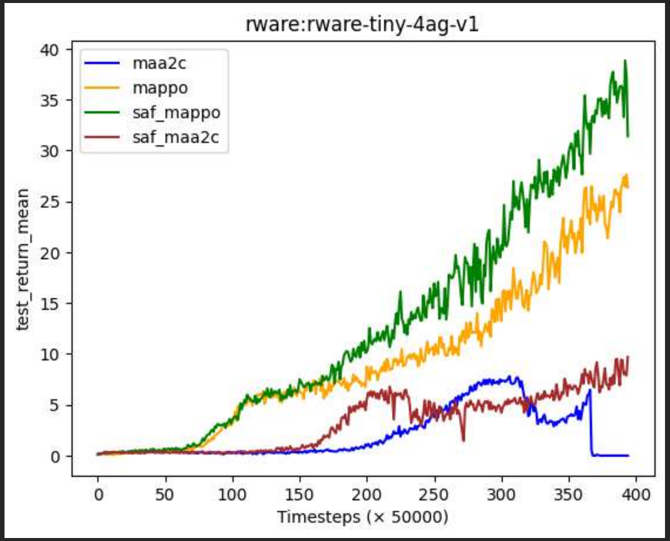
MAPPO performs best compared to MAA2C

SAF **improves** MAPPO

SAF marginally improves MAA2C

QMIX not tested



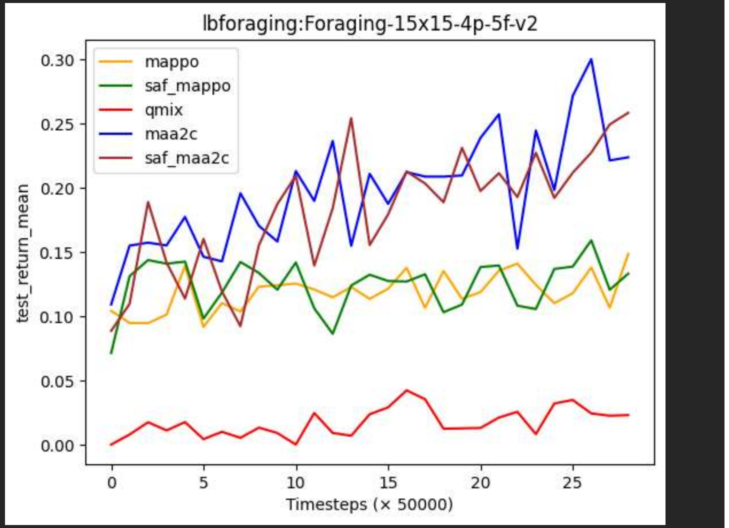


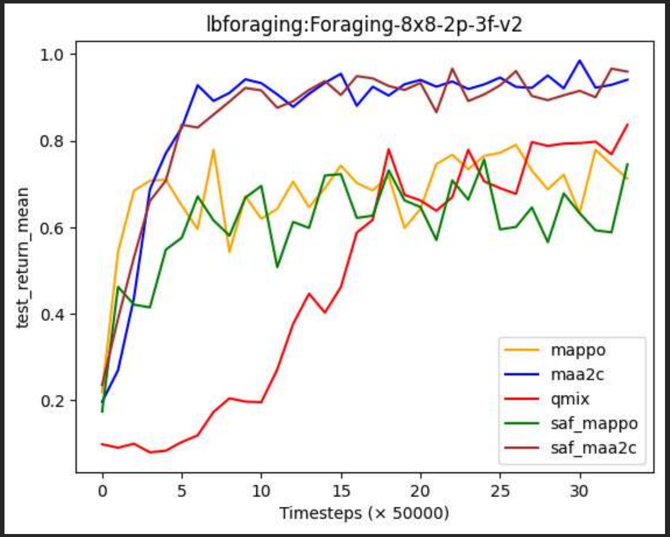
Results (LBF)

MAA2C performs best

SAF does not affect either algorithm

QMIX competitive on smaller map, but not bigger one



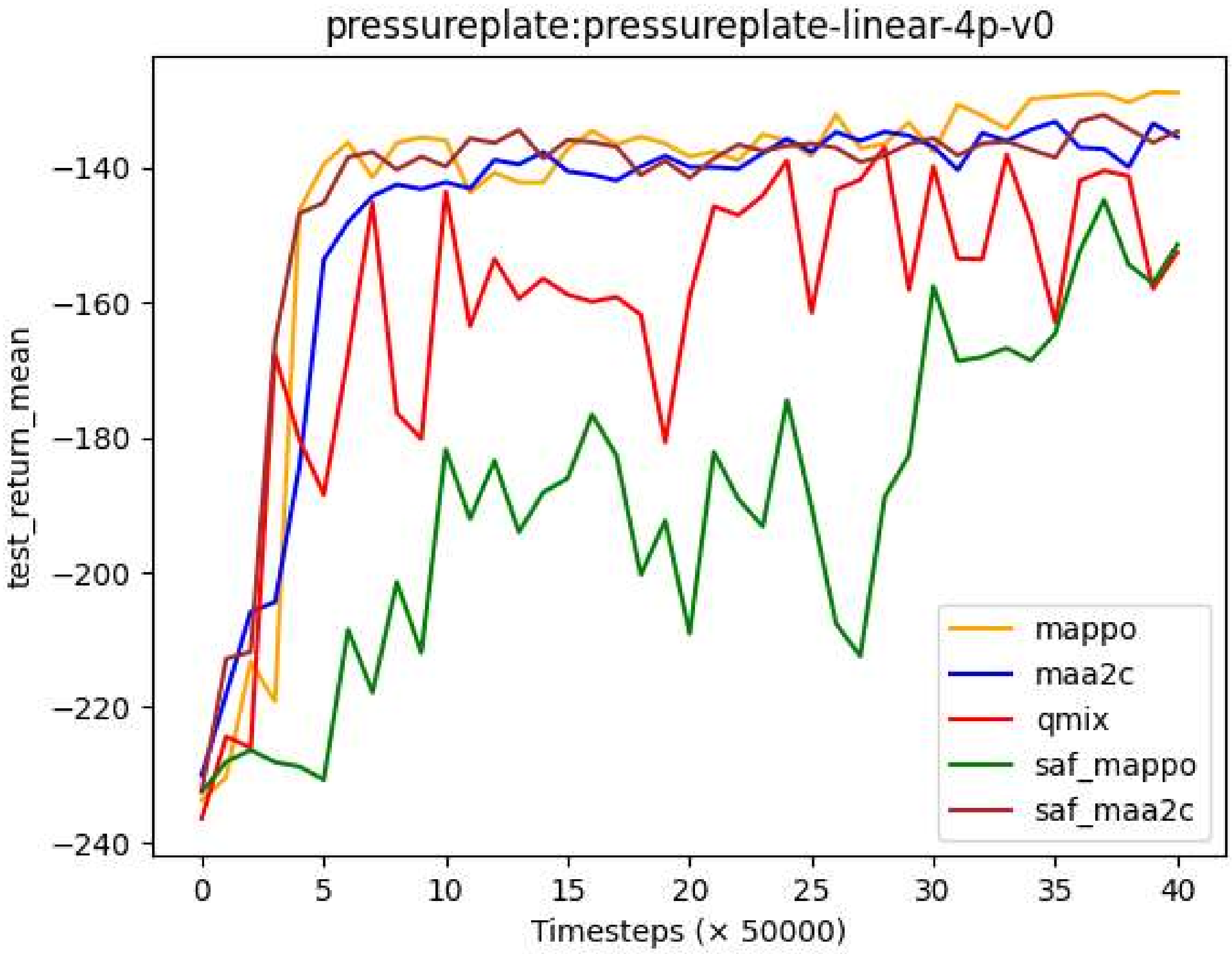


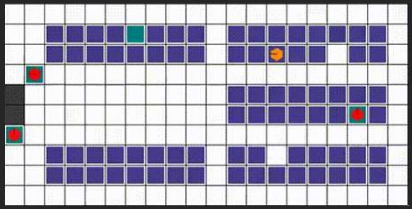
Results (PressurePlate)

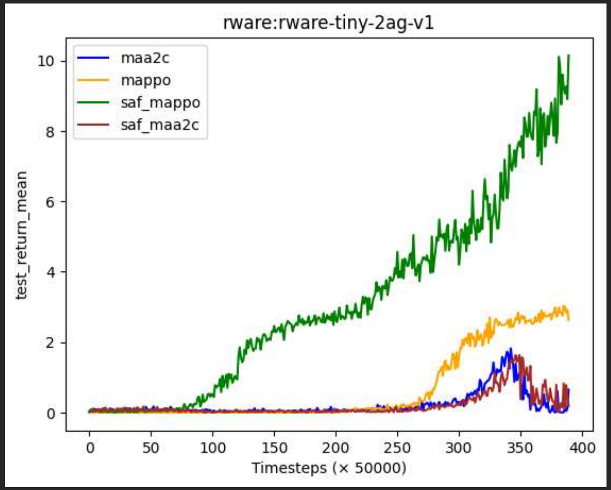
MAPPO performs best

SAF slightly decreases MAPPO performance

SAF does not affect MAA2C







Discussion (SAF on envs)

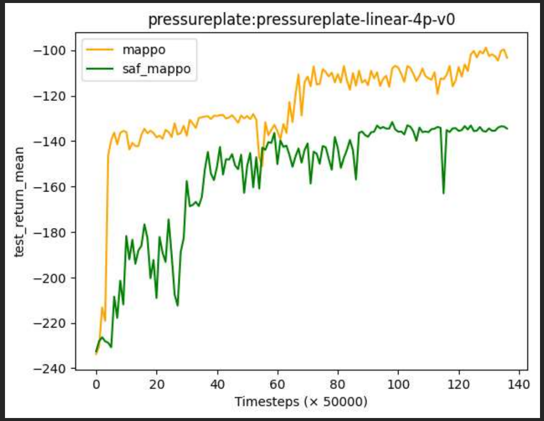
SAF can improve MAPPO in specific environment tasks

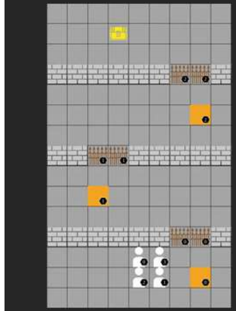
Partial observability of RWARE could make KS improve agents' movements

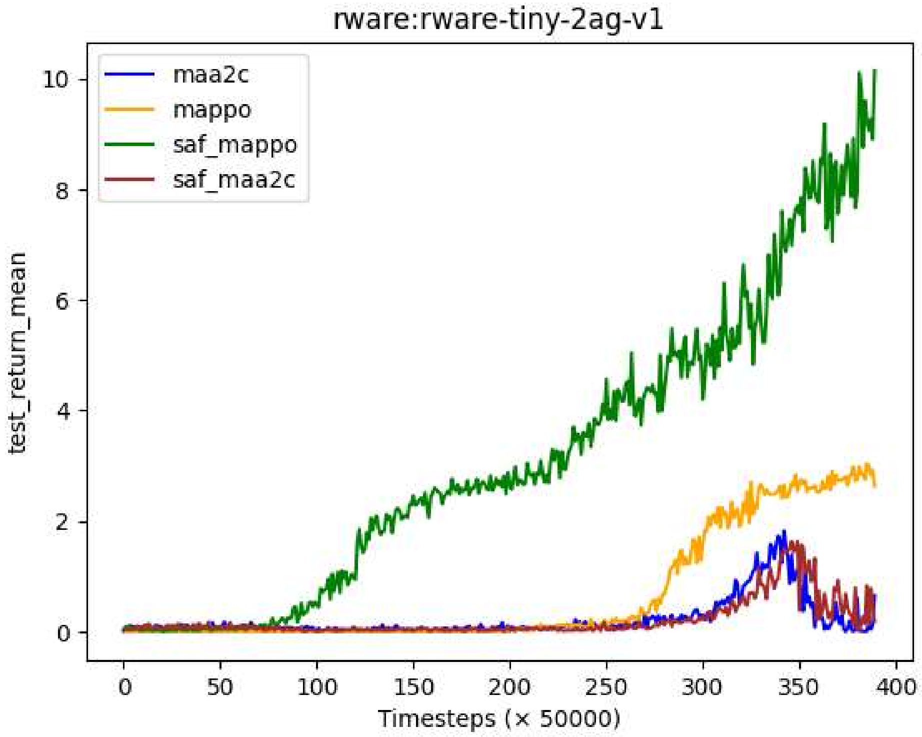
- 3×3 grid of observability

SAF worsens in others

PressurePlate has each agent almost independent in their task of reaching a plate







Discussion (SAF on algos)

SAF improves MAPPO greatly

But does not affect MAA2C much

Could attribute to MAPPO having better sample efficiency (Schulman et al., 2017)

- MAPPO can “learn more” from each sample

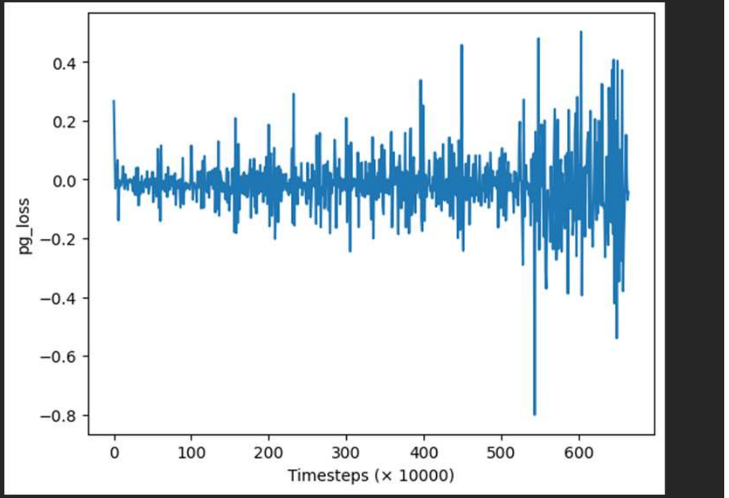
SAF can leverage with its KS

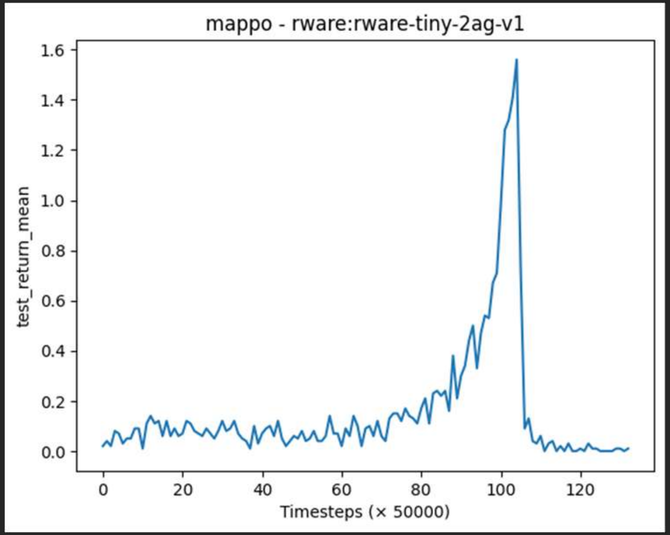
Challenges: Instability

Requires over 12 hours of training

Error while training misconfigured MAPPO on RWARE

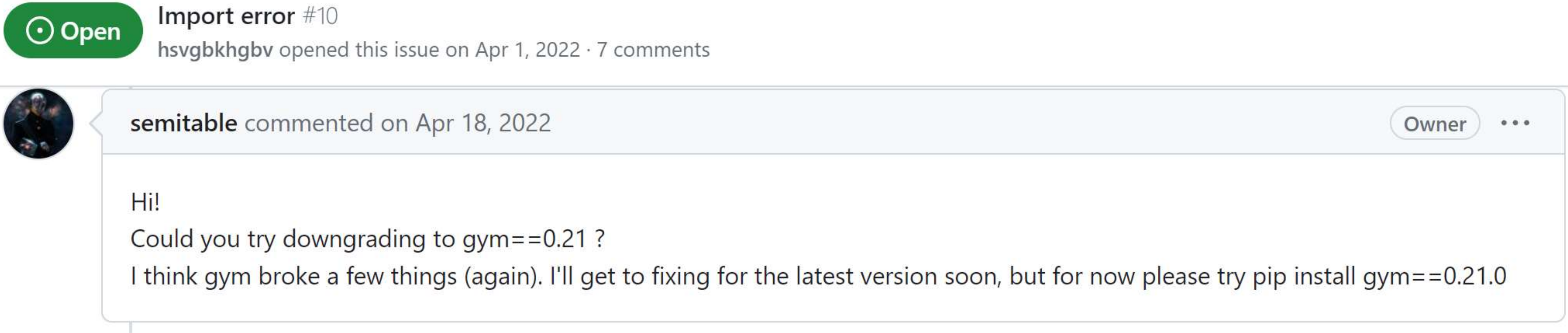
Showcases sensitivity of training RL





Challenges: Dependencies

Dependency issues must be handled properly to train



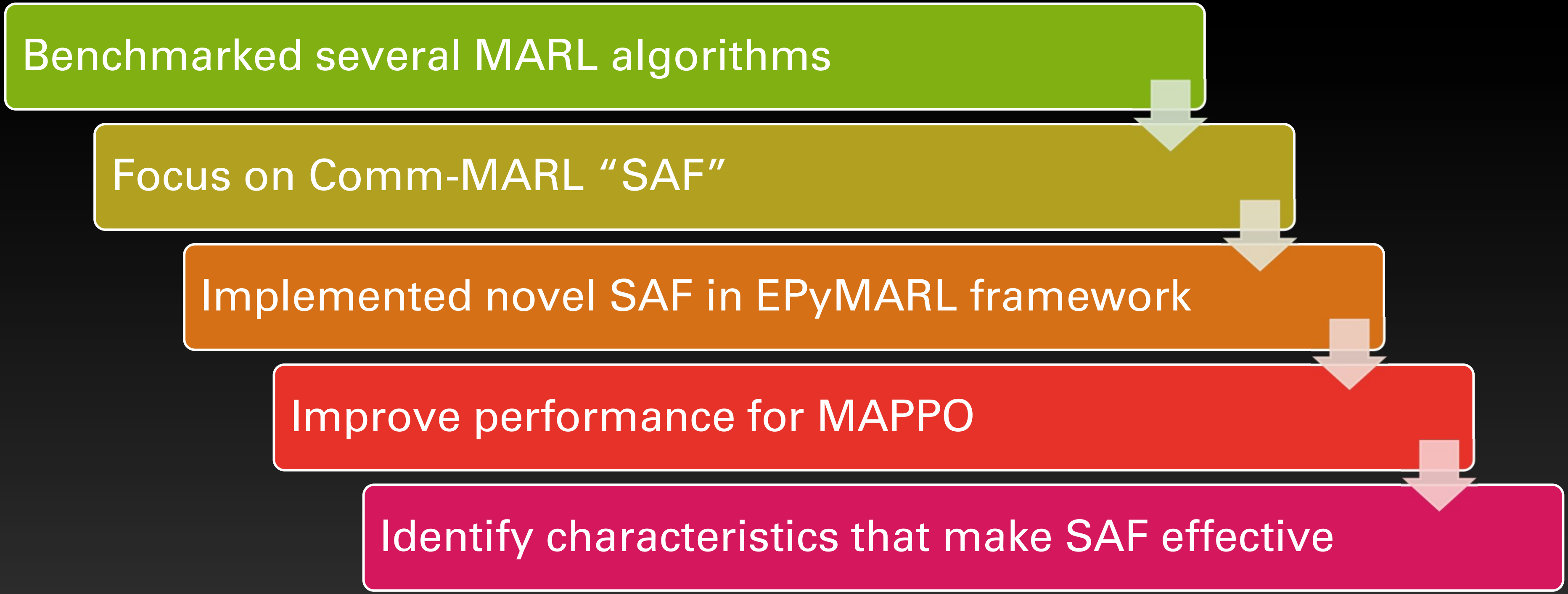
Import error #10

hsvgbkhgbv opened this issue on Apr 1, 2022 · 7 comments

semitable commented on Apr 18, 2022

Hi!
Could you try downgrading to gym==0.21 ?
I think gym broke a few things (again). I'll get to fixing for the latest version soon, but for now please try pip install gym==0.21.0

Conclusion



Benchmarked several MARL algorithms

Focus on Comm-MARL “SAF”

Implemented novel SAF in EPyMARL framework

Improve performance for MAPPO

Identify characteristics that make SAF effective

Future Work

Shown: Improving performance of baseline

1. Check generalization over much more tasks
2. Combine more ideas, like the combination of SAF and MAPPO
3. Configuring SAF hyperparameters
4. Adding SAF policy pool (not implemented here)
5. Benchmarking more Comm-MARL algorithms
6. Complexity analysis for efficiency of algorithms