TEXAS A&M UNIVERSITY CORPUS CHRISTI

Adaptive Robotic Intelligence for Complex Environment

복잡계 적응을 위한 로봇 지능 구조에 관한 연구

K-TAG 네트워킹 신기술 세미나
10월 17일(화) 오후 8시

Um Dugan, Ph.D.

Texas A&M University – Corpus Christi

Robots are working in a complex environment?

Robots are replacing human labor

Various decision makings required for tasks in a complex environment

Sensing, actuation, sorting, navigation, handling

Complex adaptive system approach for complex changing environments

- A **complex adaptive system** is a system that is *complex* in that it is a dynamic network of interactions, but the behavior of the ensemble may not be predictable according to the behavior of the components.
- It is a "complex macroscopic collection" of relatively "similar and partially connected micro-structures" formed in order to adapt to the changing environment and increase their survivability as a macro-structure.[1][2][4]
- Popular approaches : replicator dynamics (mathematical models for individual or groups to make decisions.).[5]
- Our approaches : Hierarchical Reinforcement Learning

What’s required for a complex adaptive system

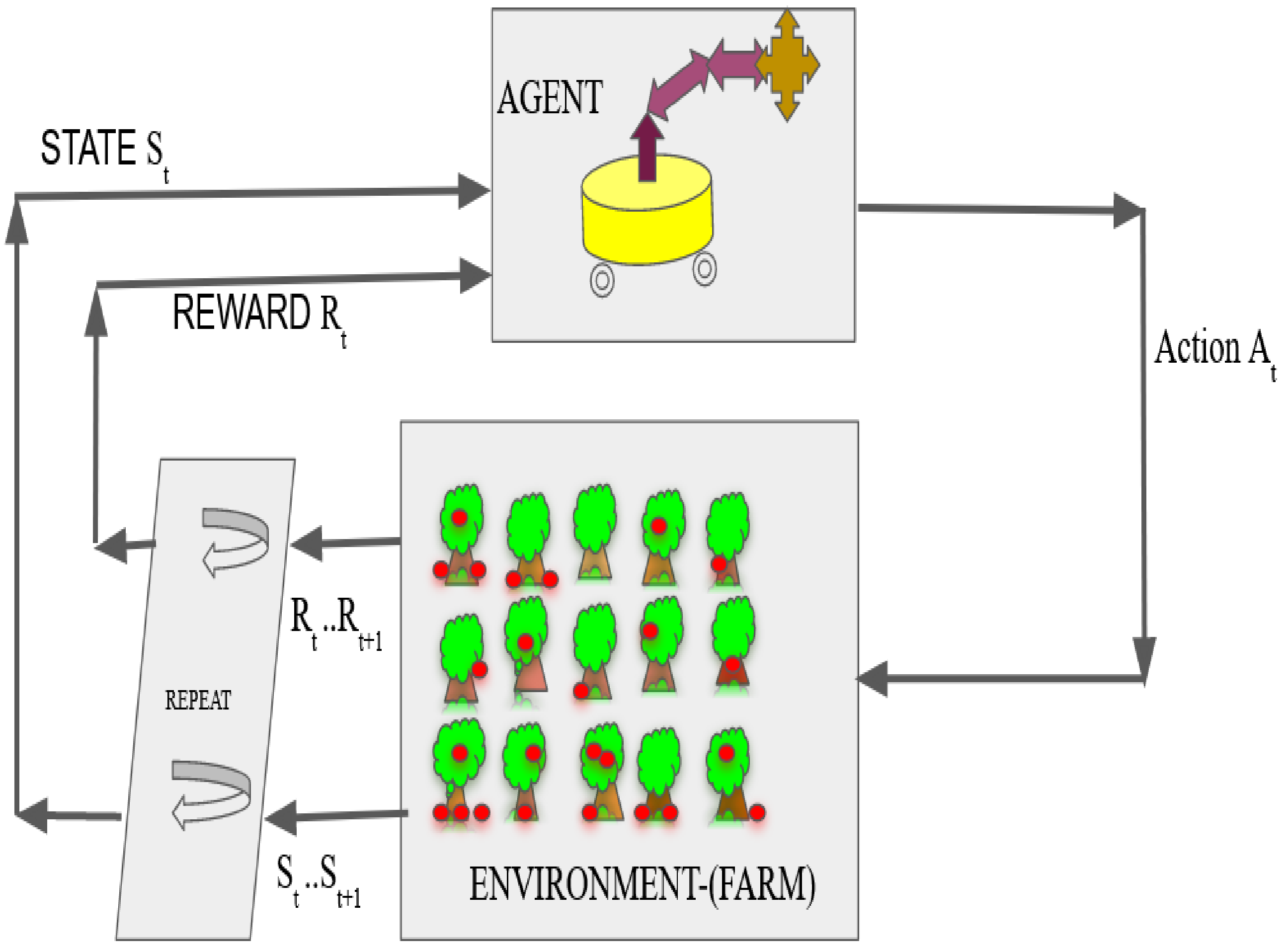
- **Ability to deal with complex event horizon tasks**
- **Ability to deal with conflicting goals**
- **Ability to train itself in unsupervised manner**
- **Ability to incorporate transfer learning**
- **Ability for adaptive evolution for further complex environments**

Complex and changing environment

Delivery robot

- Path planning
- Navigation
- Door open/close
- Collision avoidance

What is Reinforcement Learning?



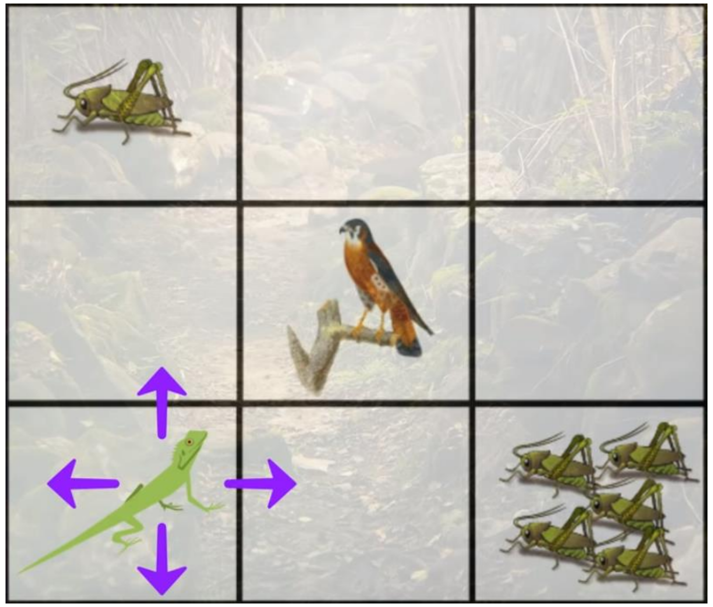
- Agent: An entity that takes actions (an algorithm). For example, A Mobile Robot / Drone.

- Action ***(a)***: Decisions taken by the agent in the environment.

- Farm Environment.

  - **State (*S*)**: *observations* from the environment.
  - **Reward**: An immediate response from the environment to an agent's behaviors.
  - **Policy**: Policy is a function→ maps states and actions, denoted by $\boldsymbol{\pi}$, $\boldsymbol{\pi}(S) = a$.

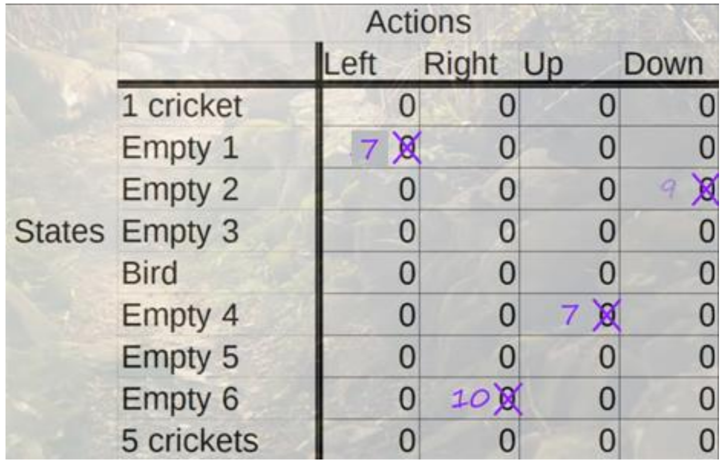
Bellman Equation



Expresses the value of a decision dilemma for a given state in terms of rewards obtained from the action and value of state resulting from that action taken.

$$V_\pi(s) = R(S, a) + \gamma V_\pi(S')$$

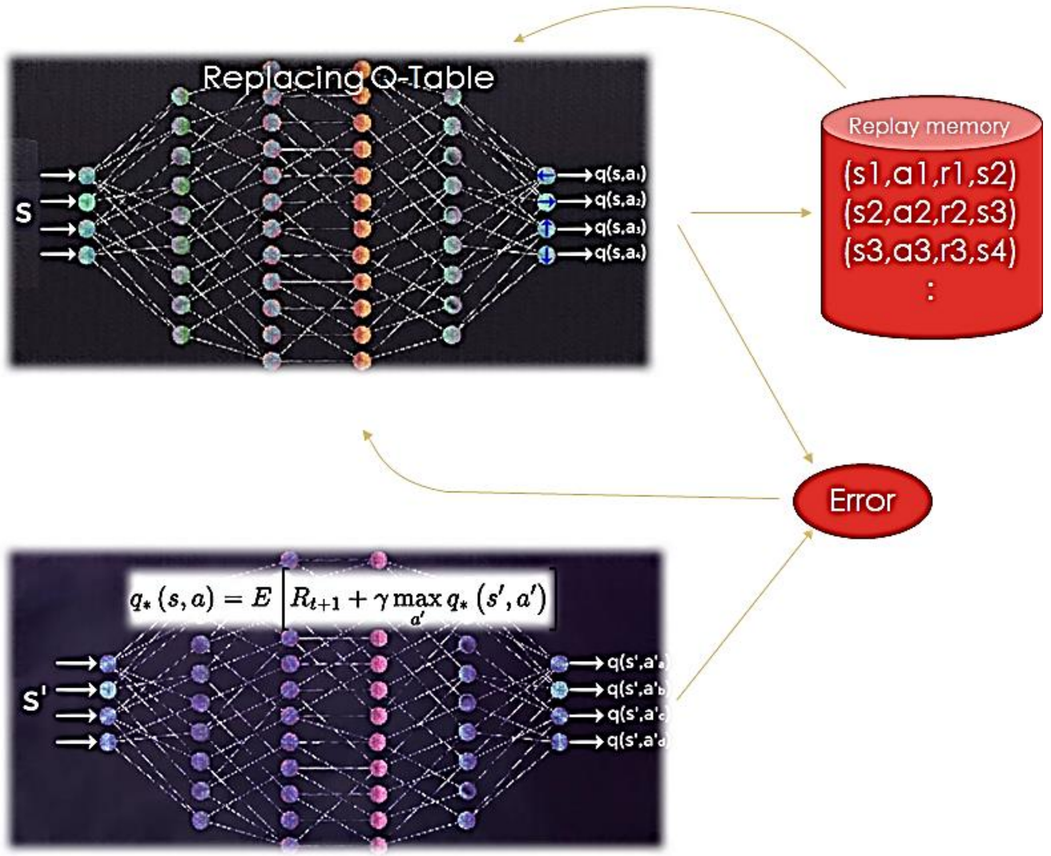
Computationally Bellman's optimality principle is very expensive to choose the most effective strategy.



		Actions			
		Left	Right	Up	Down
	1 cricket	0	0	0	0
	Empty 1	7 ~~0~~	0	0	0
	Empty 2	0	0	0	9 ~~0~~
States	Empty 3	0	0	0	0
	Bird	0	0	0	0
	Empty 4	0	0	7 ~~0~~	0
	Empty 5	0	0	0	0
	Empty 6	0	10 ~~0~~	0	0
	5 crickets	0	0	0	0

- Complex environment → environmental model is unknow
- Model Free Algorithm → good for complex environment

V and *Q* values of Reinforcement Learning



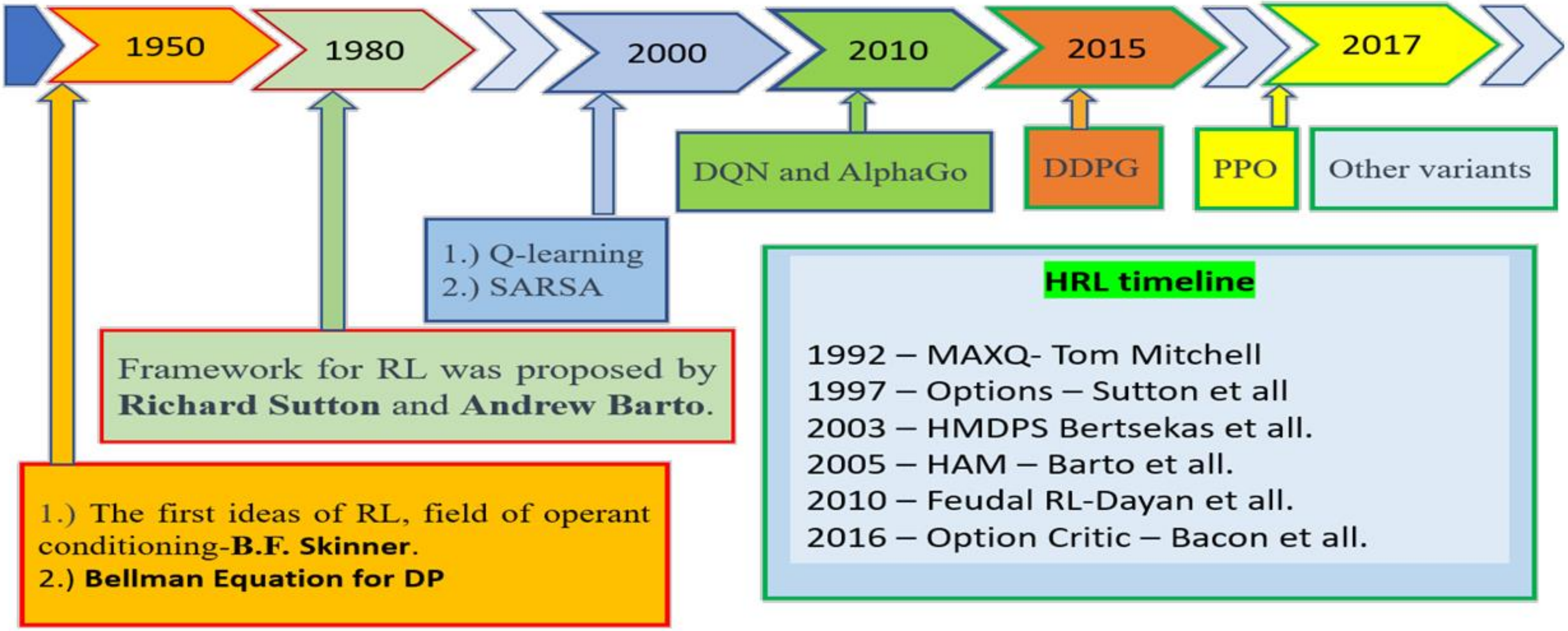
State Value Function or Value Function (*V*): Measuring Value of state as Reward.

$$V_\pi(S) = E[R|S_0 = S] = E[\sum \gamma^t r^t | S_0 = S]$$

***Q*-Value or Action-Value (*Q*):** Measuring Value of both state and action as reward.

$$Q_\pi(S,a) = E[R|S,a,\pi] = E[\sum \gamma^t r^t | S,a,\pi]$$

Brief time of Reinforcement Learning



Framework for RL was proposed by **Richard Sutton** and **Andrew Barto**.

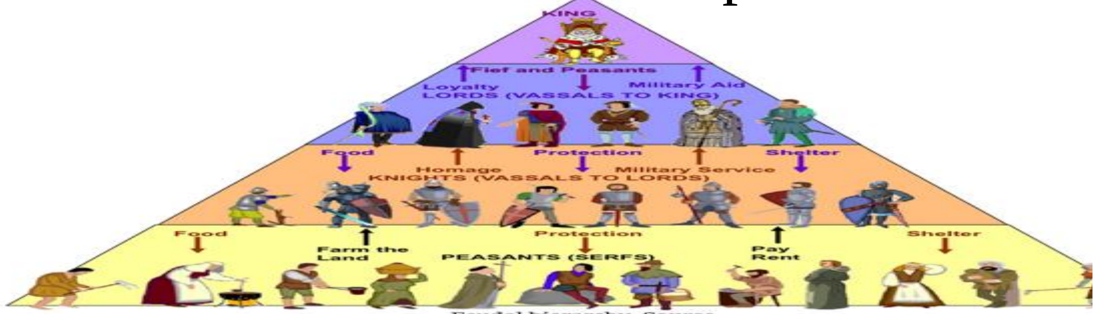
1.) The first ideas of RL, field of operant conditioning-**B.F. Skinner**.
2.) **Bellman Equation for DP**

HRL timeline

1992 – MAXQ- Tom Mitchell
1997 – Options – Sutton et all
2003 – HMDPS Bertsekas et all.
2005 – HAM – Barto et all.
2010 – Feudal RL-Dayan et all.
2016 – Option Critic – Bacon et all.

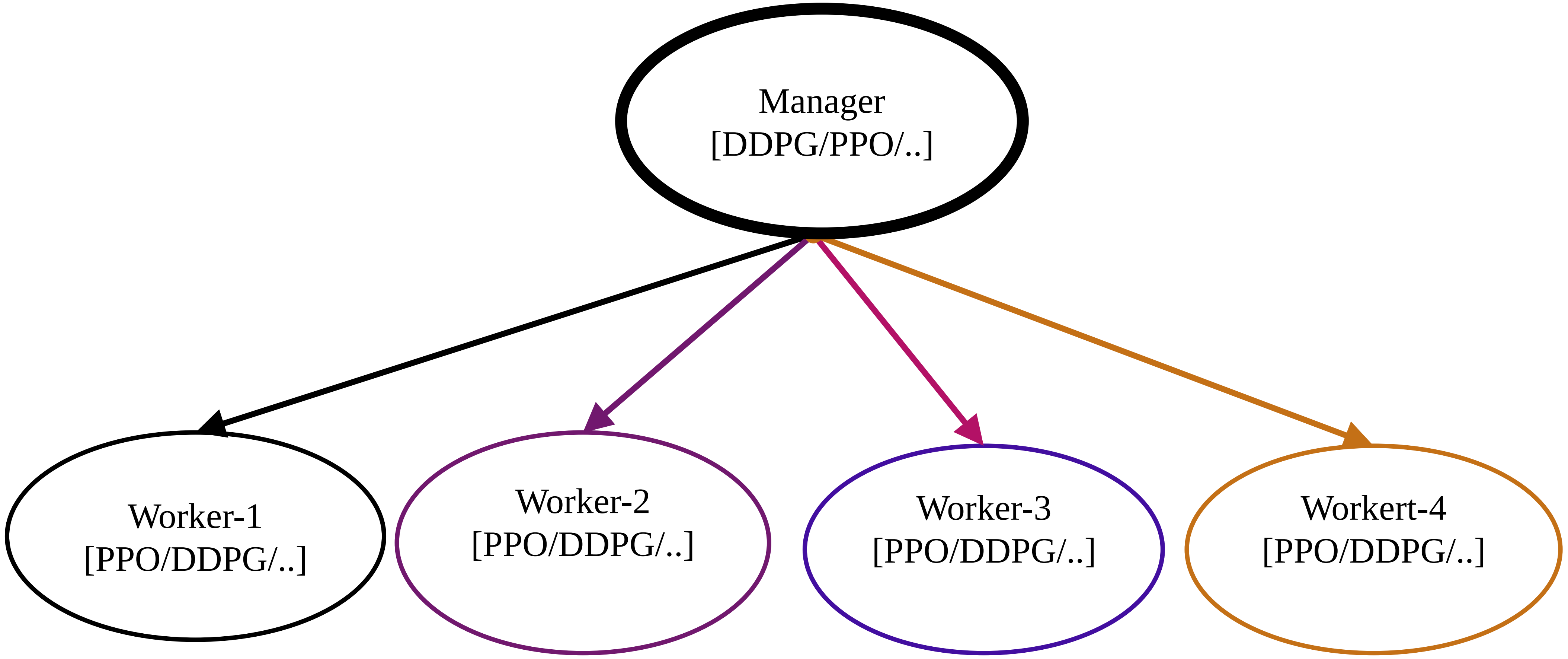
Hierarchical Reinforcement Learning (HRL)

- Basic RL algorithm’s unable to scale for high dimensions.
- The state and action abstractions are discovered to help an RL algorithm to scale.
- For example, instructions like “Command a Mobile Robot to move from a target to Destination” With the use of this abstraction, a real-time agent can create a strategy for breaking down a larger task into smaller, manageable pieces.
- The Hierarchical RL idea is that an agent can have abstract actions as part of its state space that aids in faster planning.

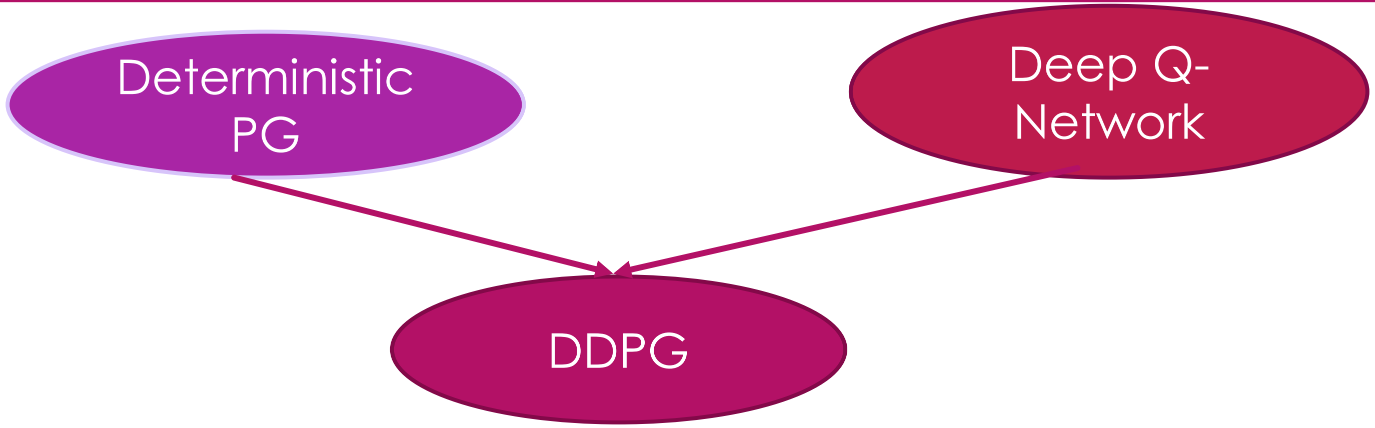


Feudal hierarchy. Source

Proposed Structure of HRL



Deep Deterministic Policy Gradient (DDPG)

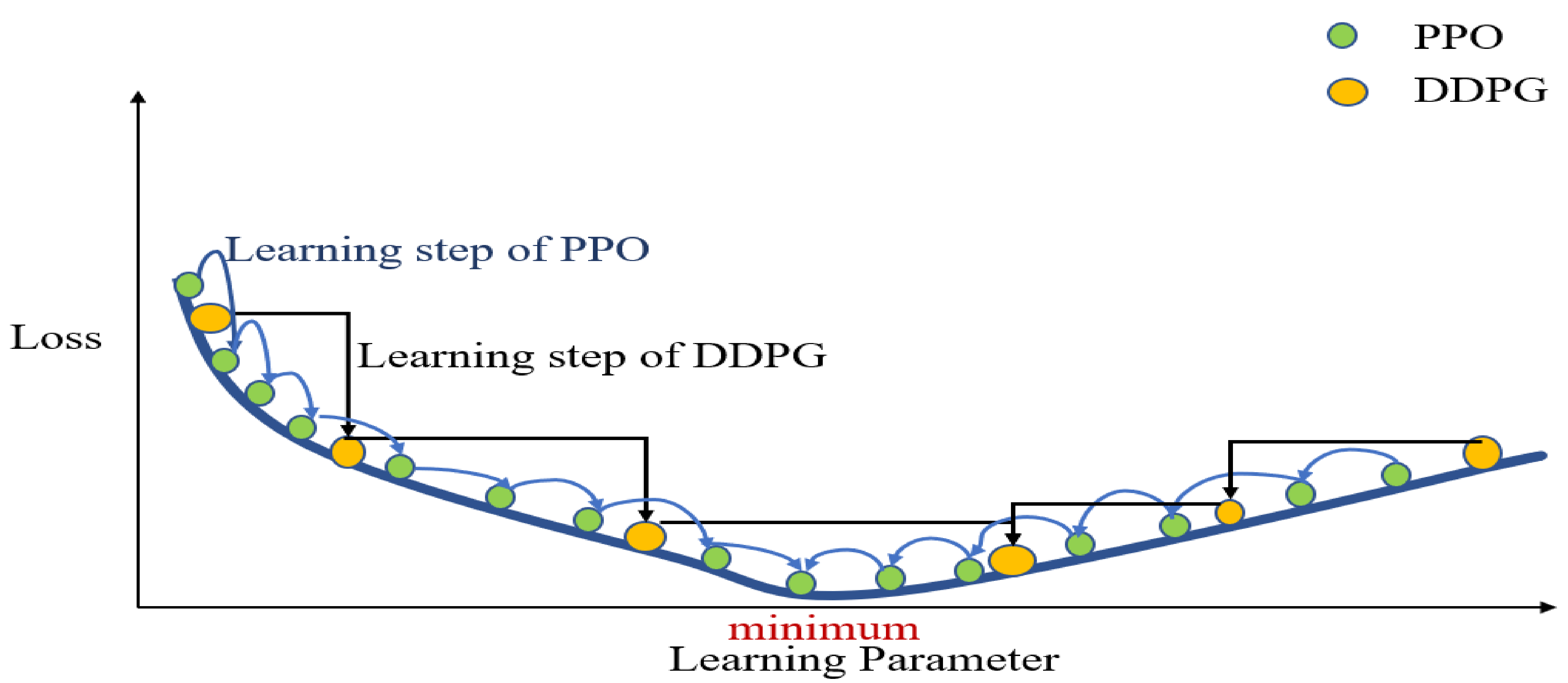


-
- The DQN works for discrete state space, whereas the DDPG extended for continuous state spaces .
- It has soft update the actor-critic networks in a way that they update the network parameters in a small time steps.
- **DPG**
  - DPG models the policy as a deterministic decision as a = **π**(S).
  - Since the probability distribution is stochastic in nature, we can think this deterministic policy as a special case of the stochastic approach.

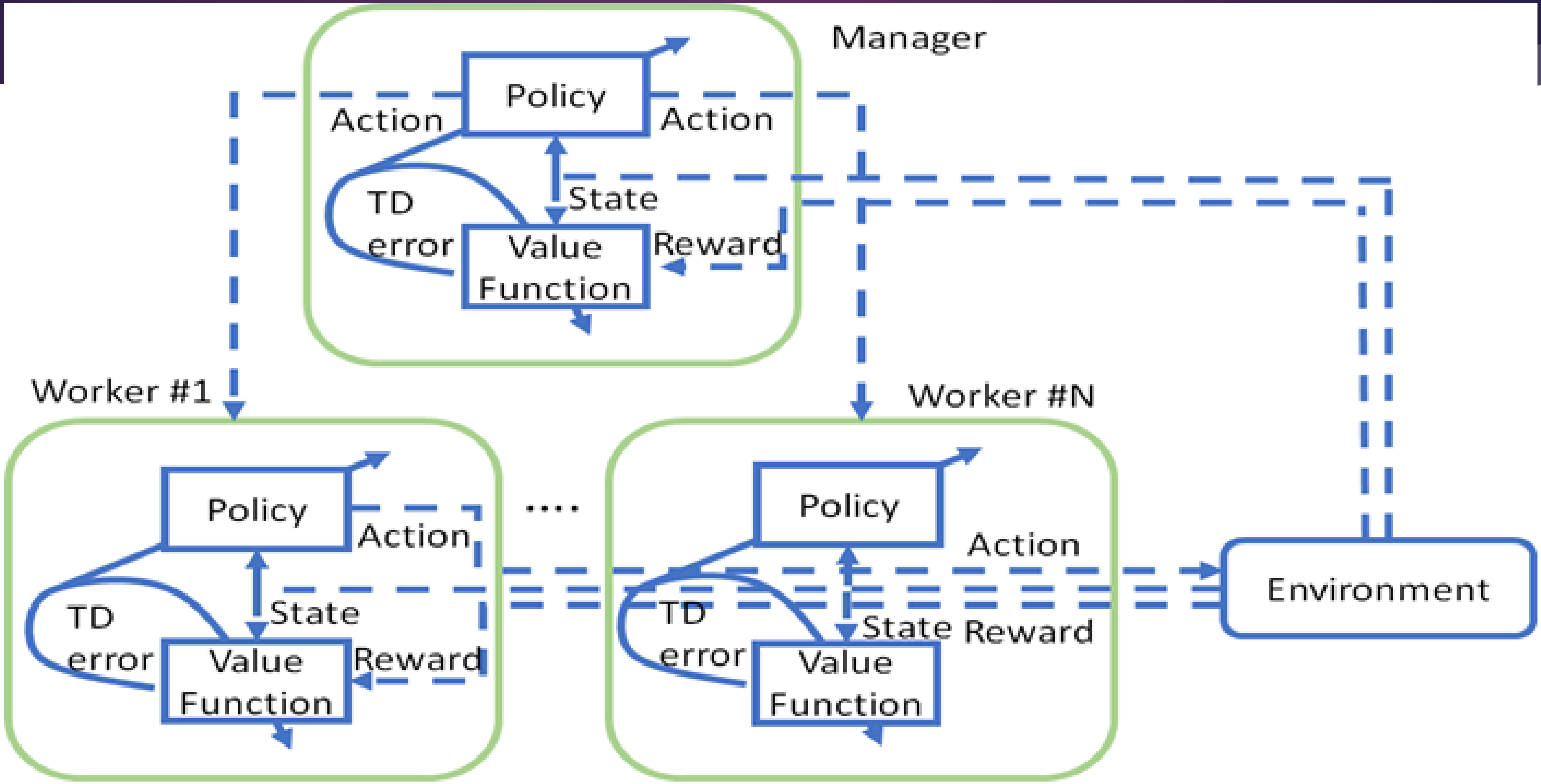
Proximal Policy Optimization (PPO)-On-policy

- For stable training, it avoids parameter updates that drastically change the policy in one step.
- By imposing a KL divergence constraint on the amount of the policy updates at each iteration, training stability improved a lot.
- Schulman et al., implemented using a clipped surrogate objective function.
- $L^{CLIP}(\theta) = E\left[\min(r(\theta)A_{\theta_{old}}(s,a)\,, clip(\gamma(\theta), 1-\varepsilon, 1+\varepsilon)A_{\theta old}(\mathrm{s},\mathrm{a}))\right]$
- θ = policy parameter, E = Denotes the empirical expectation over time steps, Probability Ratio, r = (P(θ_{new})/P(θ_{old})), A = estimated advantage value at the time t, ε = hyper parameter usually 0.1 or 0.2

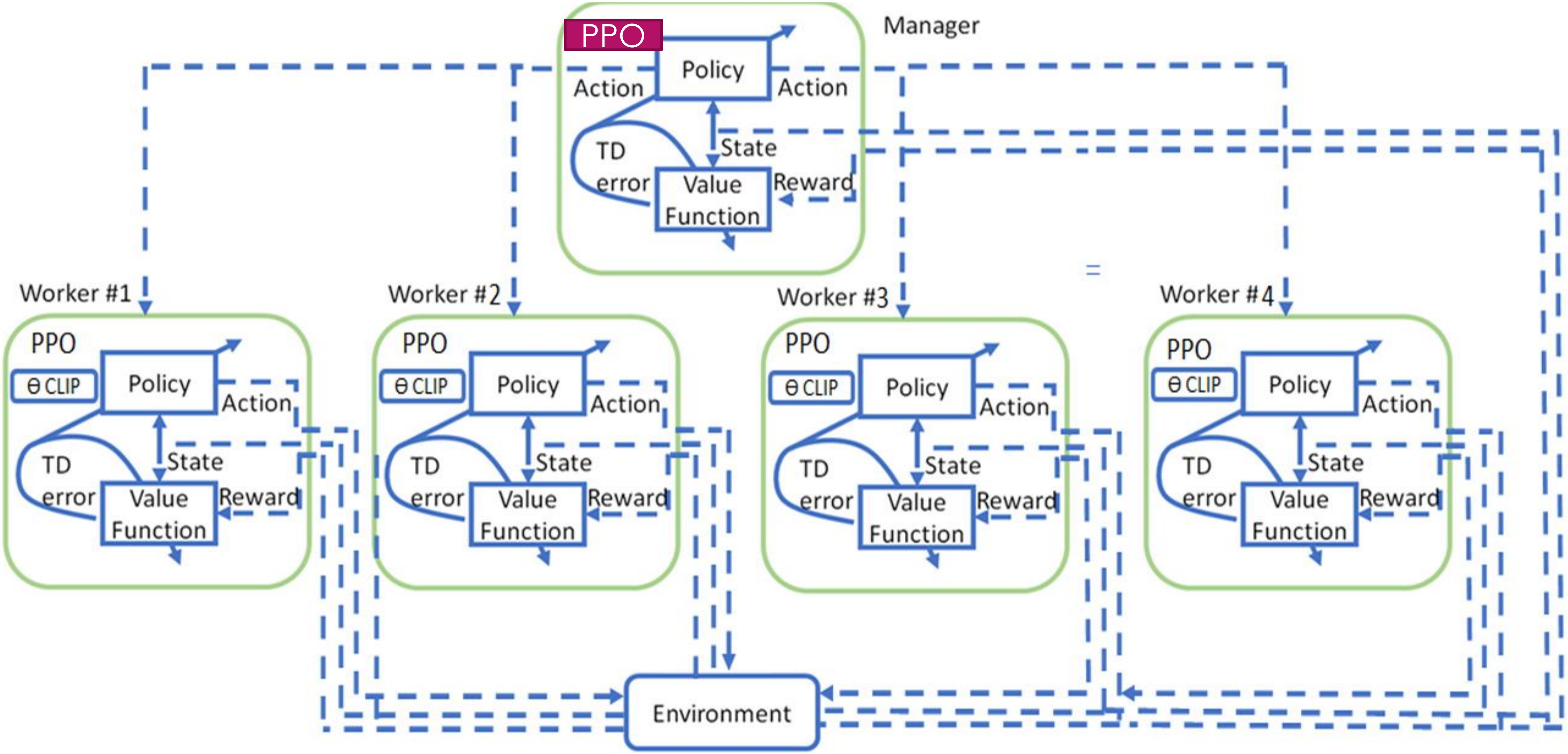
Learning Steps of PPO vs DDPG



HDDPG (HDDPG)


Manager
Policy
Action
Action
TD error
State
Value Function
Reward
Worker #1
Worker #N
Policy
Action
TD error
State
Value Function
Reward
....
Policy
Action
TD error
State
Reward
Value Function
Environment

HPPO


PPO
Manager
Policy
Action
Action
TD error
State
Value Function
Reward
Worker #1
Worker #2
Worker #3
Worker #4
PPO
θ CLIP
Policy
Action
TD error
State
Value Function
Reward
=
Environment

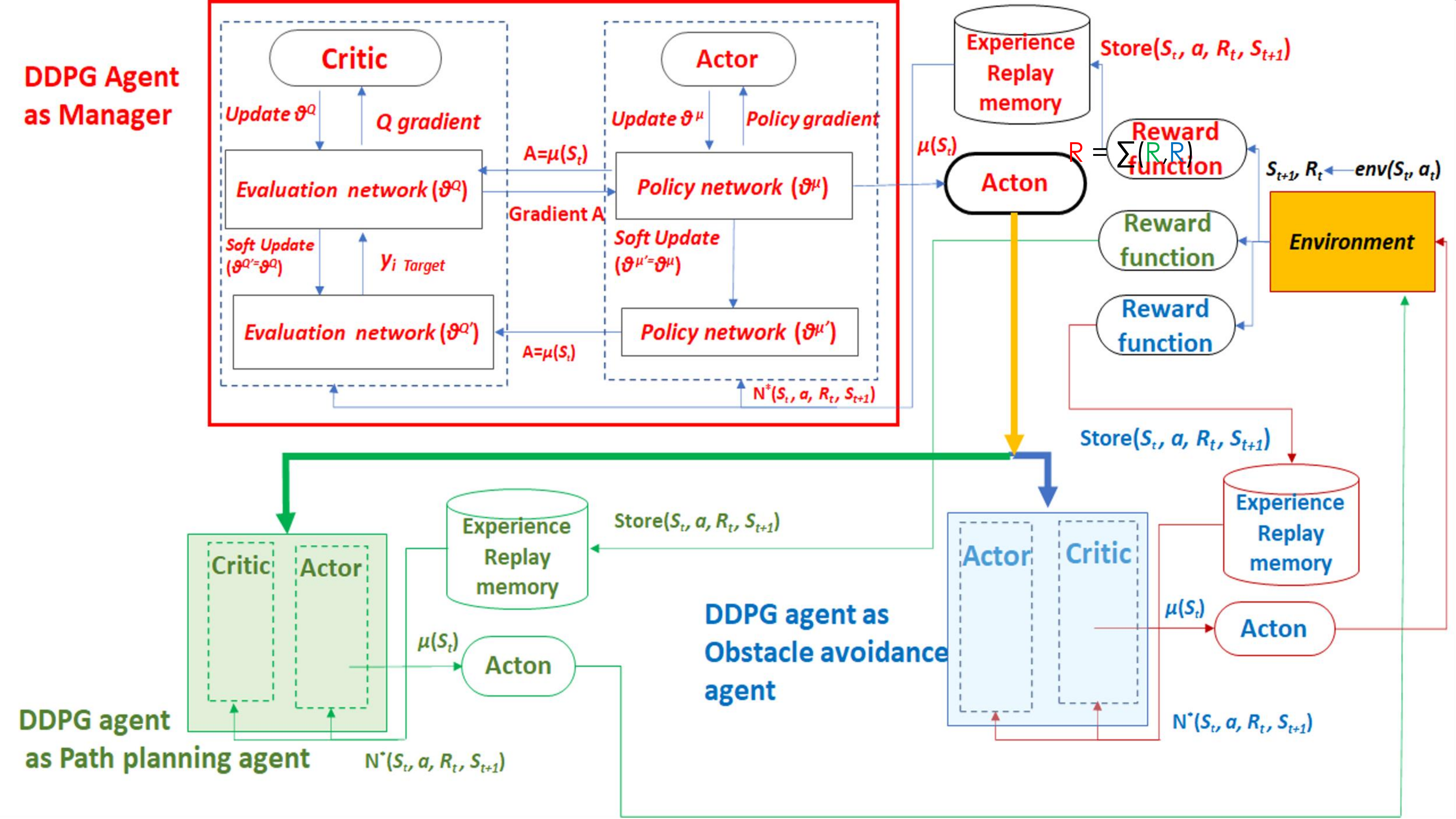
Hierarchical RL

- HRL algorithms have the potential to be useful in applications, particularly those where tasks can be decomposed into a hierarchy of subtasks (Complex Event Horizon Problems).

- Challenges of HRL
  - Complexity: Needs to learn both high level and low-level policy.
  - Difficulty in designing hierarchical structures manually (states/rewards).
  - Lack of generalization.

Transfer Learning

- Due to high complexity of state and action spaces→ learning sufficient interaction samples can be challenging.
- This might cause safety issues in industries where making the wrong decision could have catastrophic consequences, such as autonomous driving and health informatics.
- Transfer learning in the context of RL aims to learn an optimal policy $\pi*$ for a complex environment M_t, from a set of simple environment M_s.
- This approach is to learn a generalized policy from those simple environments, which can be quickly adapted to target environments.


DDPG Agent as Manager
Critic
Actor
Update ϑ^{Q}
Q gradient
Update ϑ^{μ}
Policy gradient
$A=\mu(S_t)$
Evaluation network (ϑ^{Q})
Policy network (ϑ^{μ})
Gradient A
Soft Update ($\vartheta^{Q'}=\vartheta^{Q}$)
y_i Target
Soft Update ($\vartheta^{\mu'}=\vartheta^{\mu}$)
Evaluation network ($\vartheta^{Q'}$)
Policy network ($\vartheta^{\mu'}$)
$A=\mu(S_t)$
$N^*(S_t, a, R_t, S_{t+1})$
Experience Replay memory
Store(S_t, a, R_t, S_{t+1})
$\mu(S_t)$
Acton
$R = \sum(R, R)$
Reward function
S_{t+1}, R_t
$env(S_t, a_t)$
Reward function
Environment
Reward function
Store(S_t, a, R_t, S_{t+1})
Experience Replay memory
Store(S_t, a, R_t, S_{t+1})
Critic
Actor
Experience Replay memory
Actor
Critic
DDPG agent as Obstacle avoidance agent
$\mu(S_t)$
Acton
$\mu(S_t)$
Acton
DDPG agent as Path planning agent
$N^*(S_t, a, R_t, S_{t+1})$
$N^*(S_t, a, R_t, S_{t+1})$

HDDPG Algorithm

```
1. Randomly initialize critic and actor networks for training Goal and Obstacle preferred tasks.
2.  Obtain the trained model for goal touch preferred task as worker_1.
3.  Obtain the trained model for Obstacle avoidance preferred task worker_2
                                        :
4.    Obtain the trained model for Obstacle avoidance preferred task worker_n.

5.   Randomly initialize critic and actor networks of Manager agent for managerial level tasks to deploy best worker action ‘a’ for the current state ‘s’ to
achieve maximum reward.
6.  Initialize target network
7.  Initialize replay buffer RB
8.  for maximum Episode do:
9.          Initialize a random process N for action exploration.
10.         Receive initial observation state s1.
11.         for maximum step do
12.             Retrieve action probability of manager from the manager policy
13.                 If in given action probability range:
14.                         Range 1: execute worker_1 action ‘a1’
15.                         Range 2: execute worker_2 action ‘a2’
                                        :
16.                         Range n: execute worker_n action ‘an’
17.                         Store reward rt and new state st+1 transitions in RB.
18.                 If the buffer is full, train the model
19.                     N transitions (si, ai, ri, si+1) are randomly sampled from RB.
20.                     Update all the networks and compute the loss function.
21.         end
22.  end
```

State-action value(Q-value) of the Manager

Therefore, the action of a manager will be dependent on that of each worker. That is,

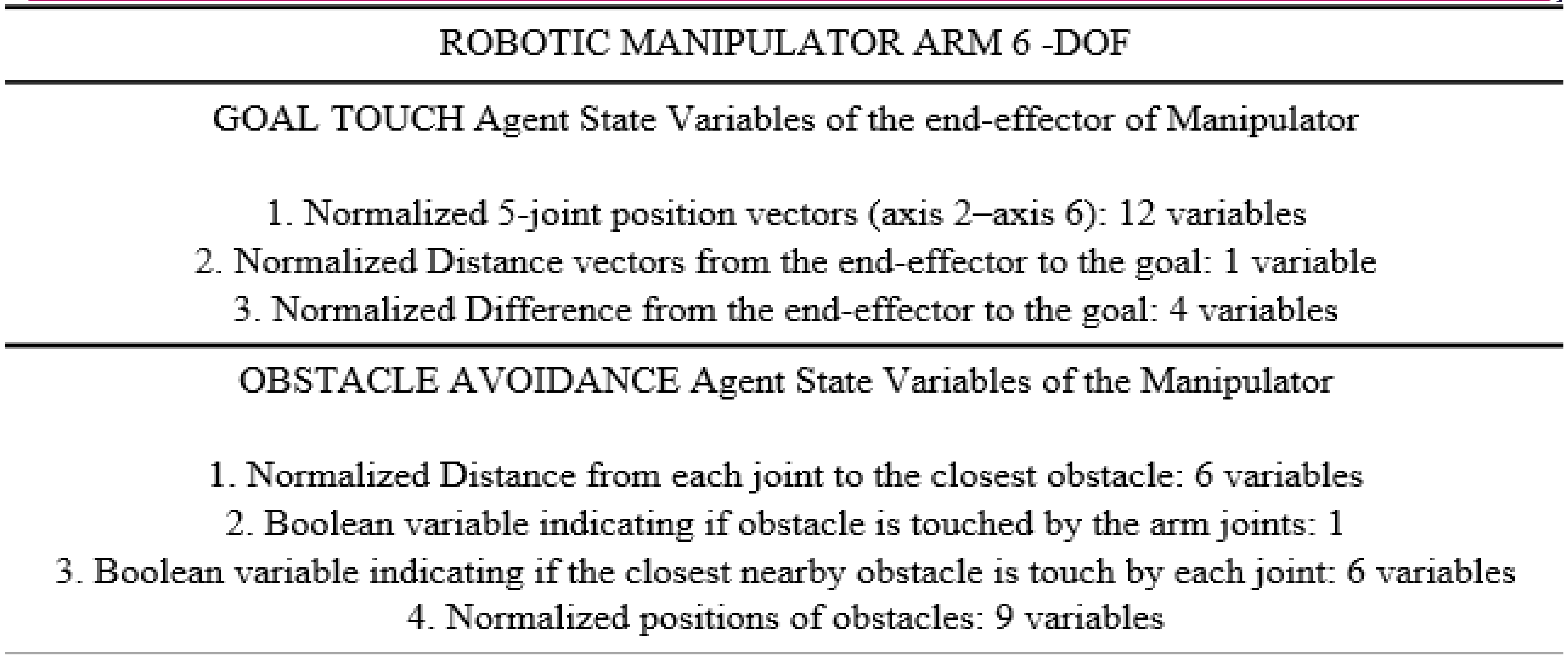
$$a|\pi = a \ \in \{a_1, a_2, a_3, ..a_n\},$$

$$Q^m_{k+1}(s,a) = R(s,a) + \Upsilon \cdot \sum_{s'} P(s'|s,a) \cdot \operatorname{argmax} Q^m_k(s',a')$$

where Q^m_k is the action-value function of a manager, is the trained policy of the manager, and

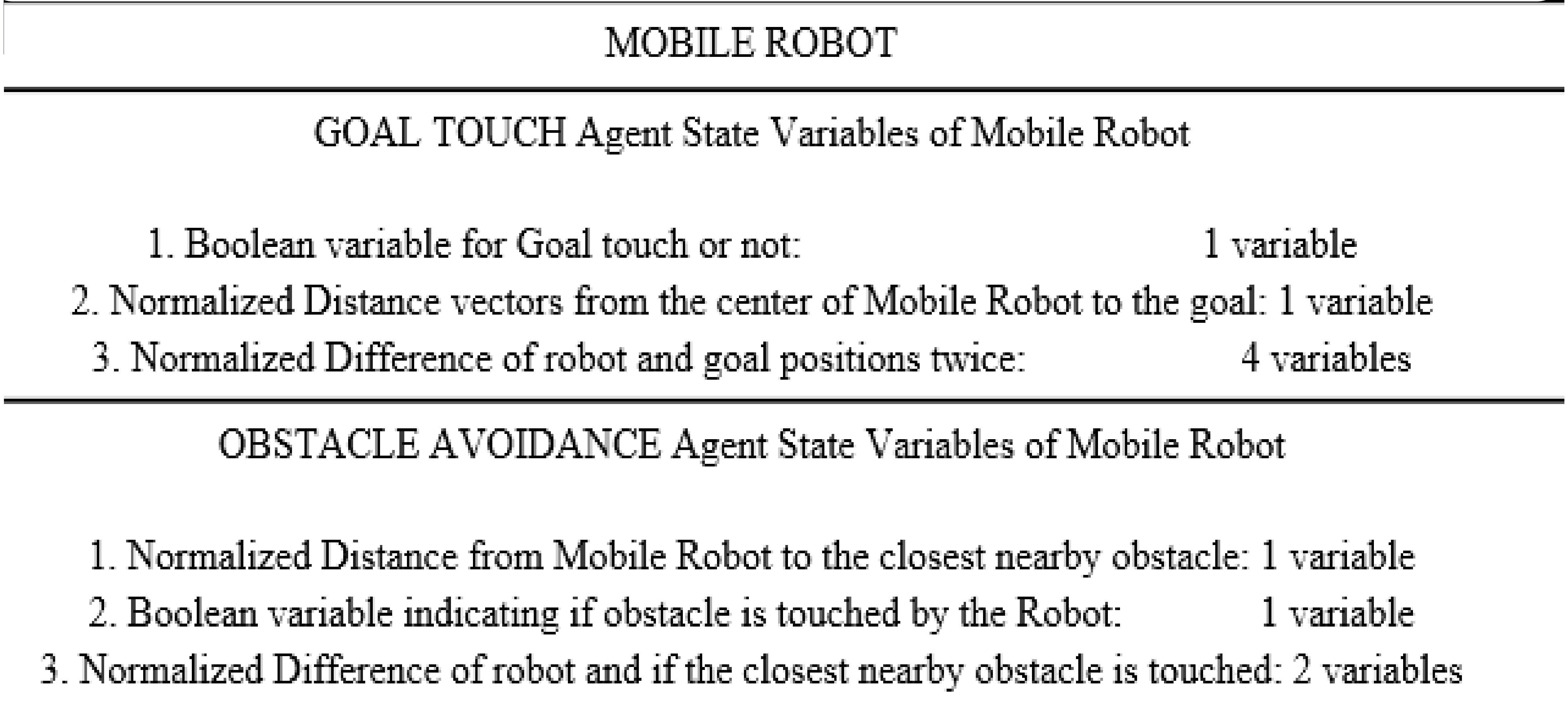
$a_1, a_2, a_3, ..a_n$ represent the action of each worker.

Total State Variables of the Environment (48)



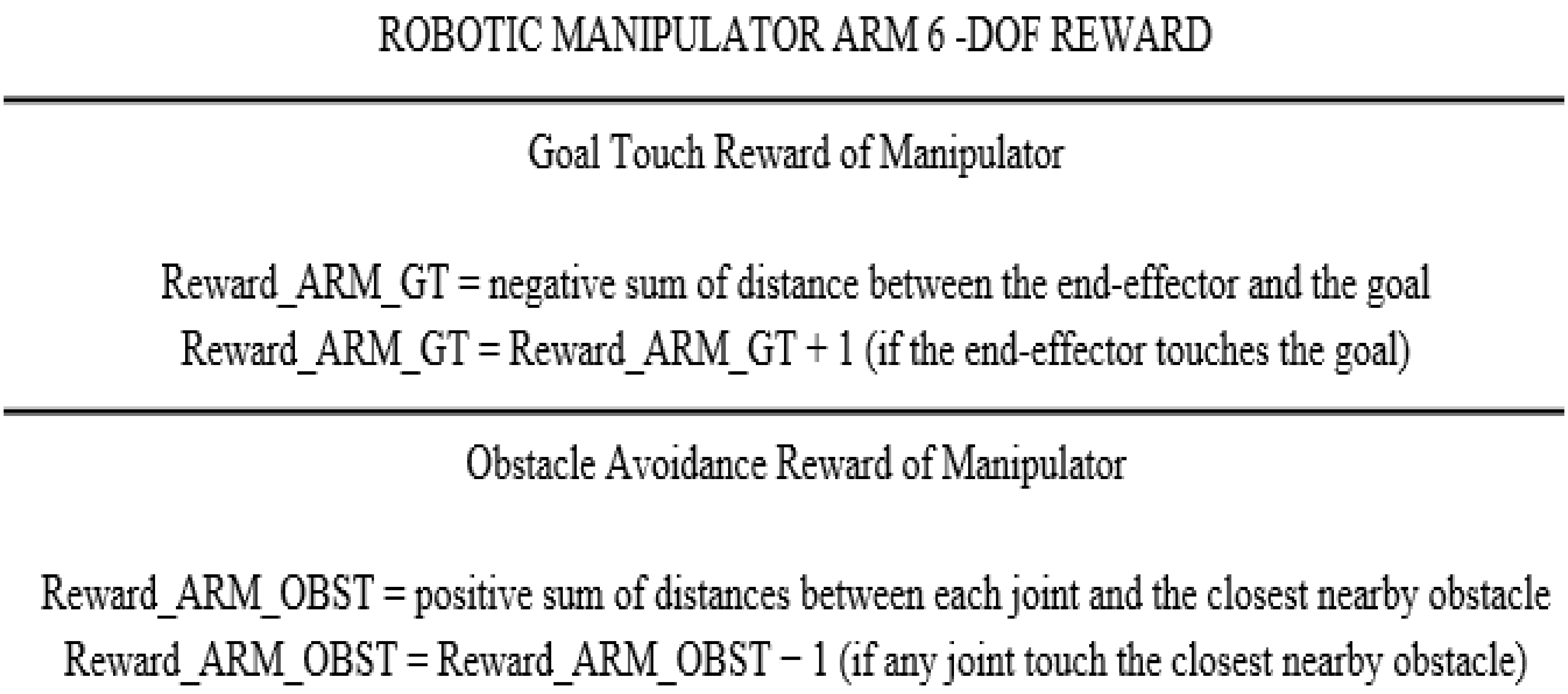
ROBOTIC MANIPULATOR ARM 6 -DOF
GOAL TOUCH Agent State Variables of the end-effector of Manipulator 1. Normalized 5-joint position vectors (axis 2–axis 6): 12 variables 2. Normalized Distance vectors from the end-effector to the goal: 1 variable 3. Normalized Difference from the end-effector to the goal: 4 variables
OBSTACLE AVOIDANCE Agent State Variables of the Manipulator 1. Normalized Distance from each joint to the closest obstacle: 6 variables 2. Boolean variable indicating if obstacle is touched by the arm joints: 1 3. Boolean variable indicating if the closest nearby obstacle is touch by each joint: 6 variables 4. Normalized positions of obstacles: 9 variables

Total State Variables of the Environment (48)



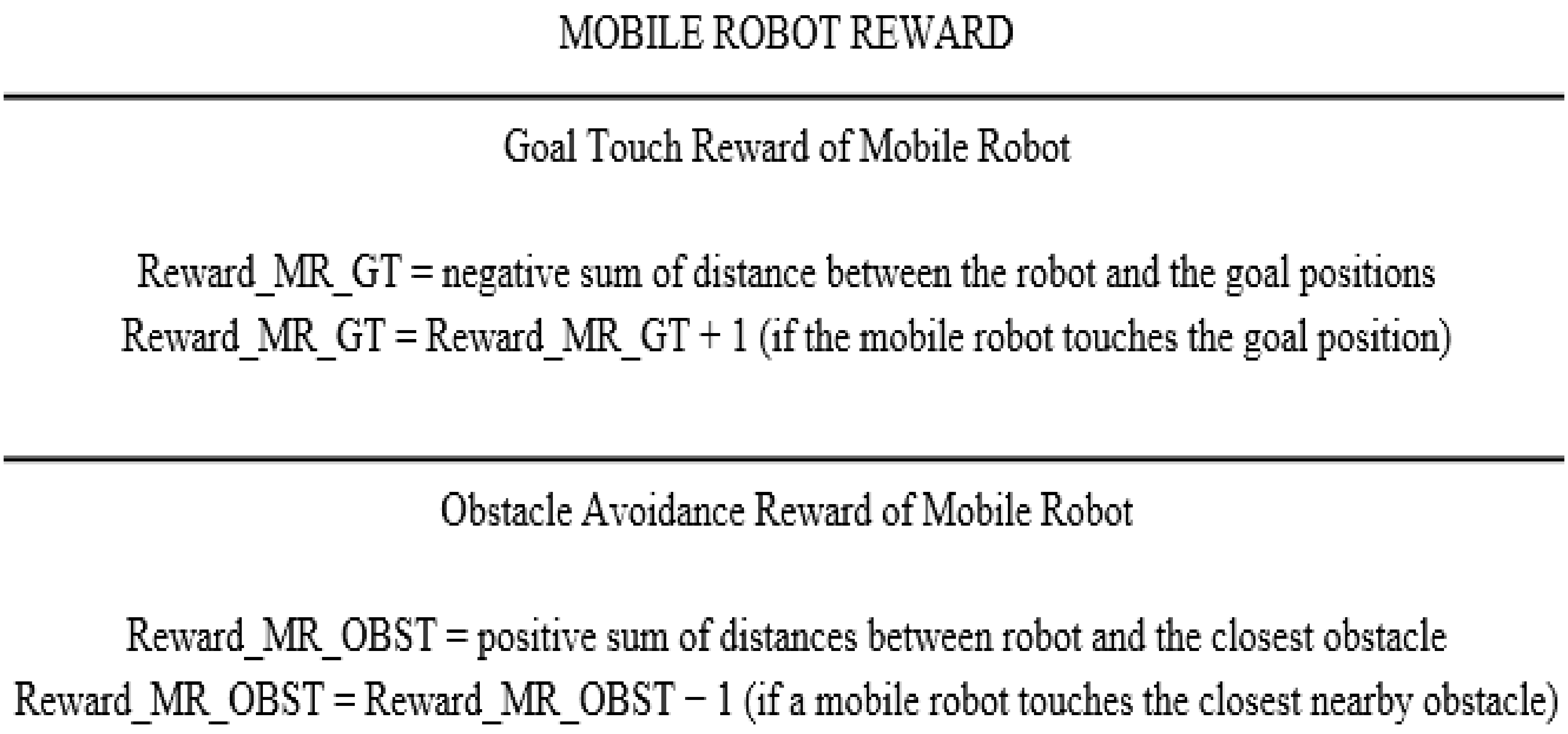
MOBILE ROBOT	
GOAL TOUCH Agent State Variables of Mobile Robot	
1. Boolean variable for Goal touch or not:	1 variable
2. Normalized Distance vectors from the center of Mobile Robot to the goal:	1 variable
3. Normalized Difference of robot and goal positions twice:	4 variables
OBSTACLE AVOIDANCE Agent State Variables of Mobile Robot	
1. Normalized Distance from Mobile Robot to the closest nearby obstacle:	1 variable
2. Boolean variable indicating if obstacle is touched by the Robot:	1 variable
3. Normalized Difference of robot and if the closest nearby obstacle is touched:	2 variables

Reward System Design



ROBOTIC MANIPULATOR ARM 6 -DOF REWARD
Goal Touch Reward of Manipulator Reward_ARM_GT = negative sum of distance between the end-effector and the goal Reward_ARM_GT = Reward_ARM_GT + 1 (if the end-effector touches the goal)
Obstacle Avoidance Reward of Manipulator Reward_ARM_OBST = positive sum of distances between each joint and the closest nearby obstacle Reward_ARM_OBST = Reward_ARM_OBST − 1 (if any joint touch the closest nearby obstacle)

Reward System Design



MOBILE ROBOT REWARD

Goal Touch Reward of Mobile Robot

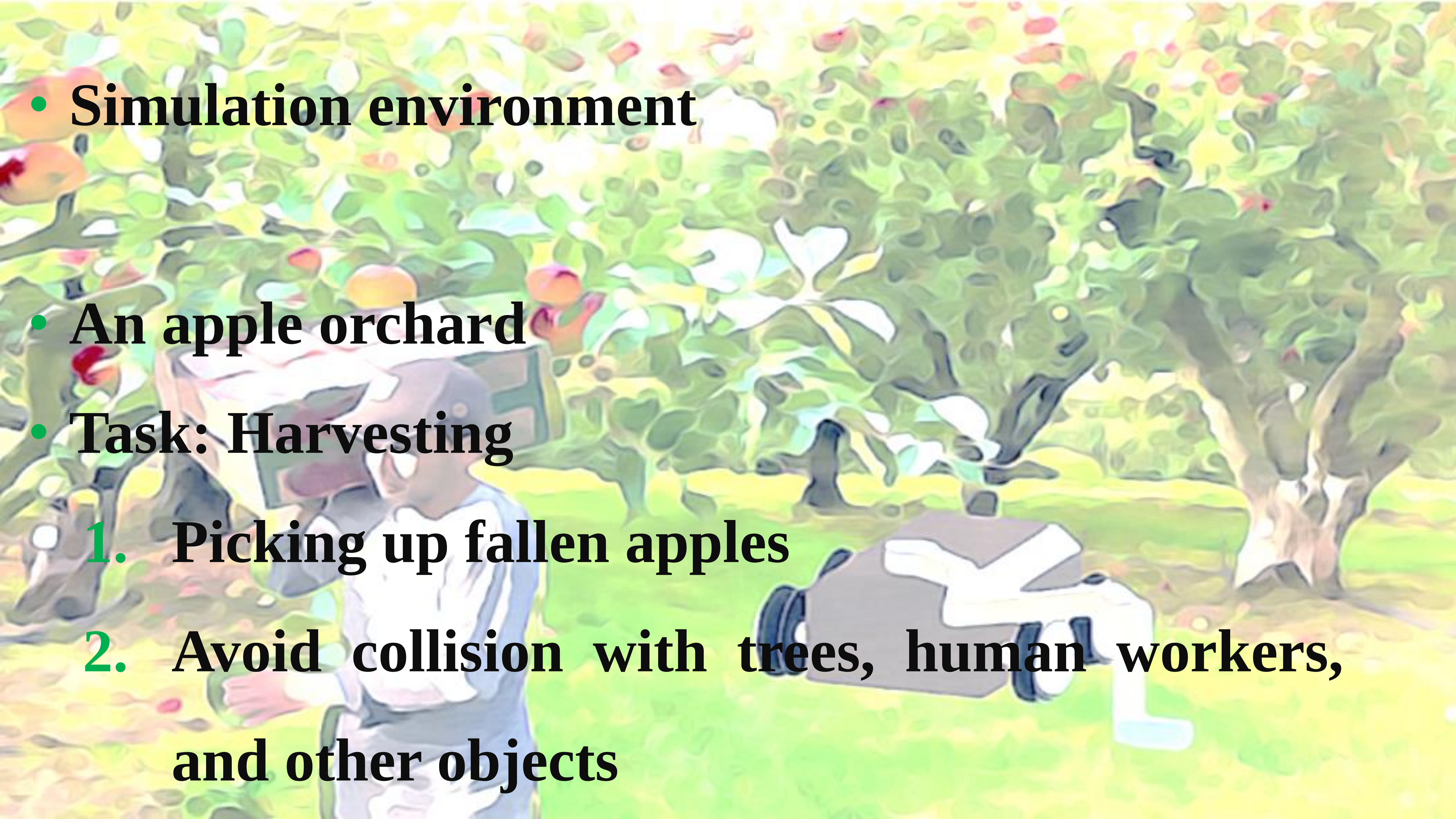
Reward_MR_GT = negative sum of distance between the robot and the goal positions

Reward_MR_GT = Reward_MR_GT + 1 (if the mobile robot touches the goal position)

Obstacle Avoidance Reward of Mobile Robot

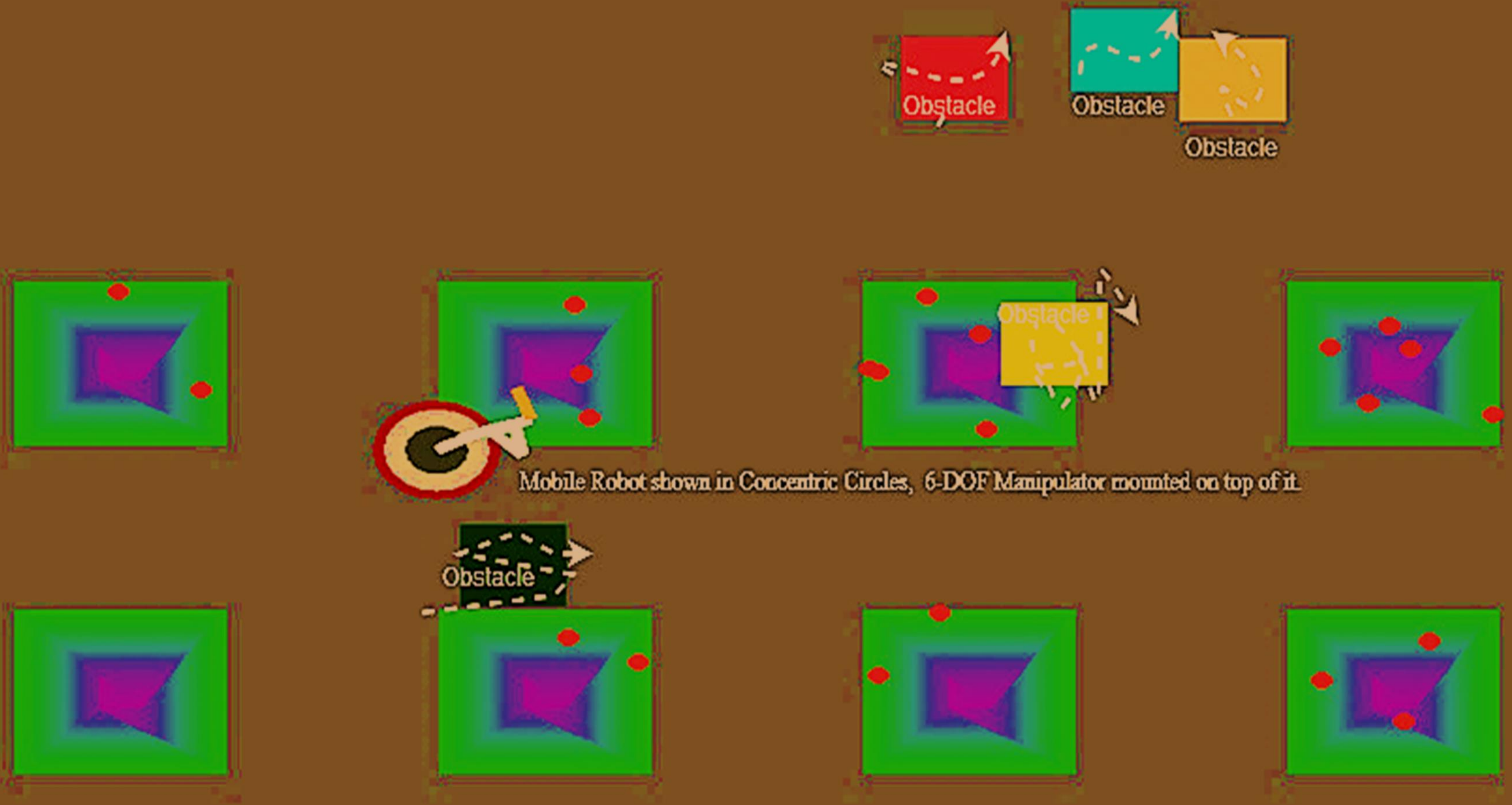
Reward_MR_OBST = positive sum of distances between robot and the closest obstacle

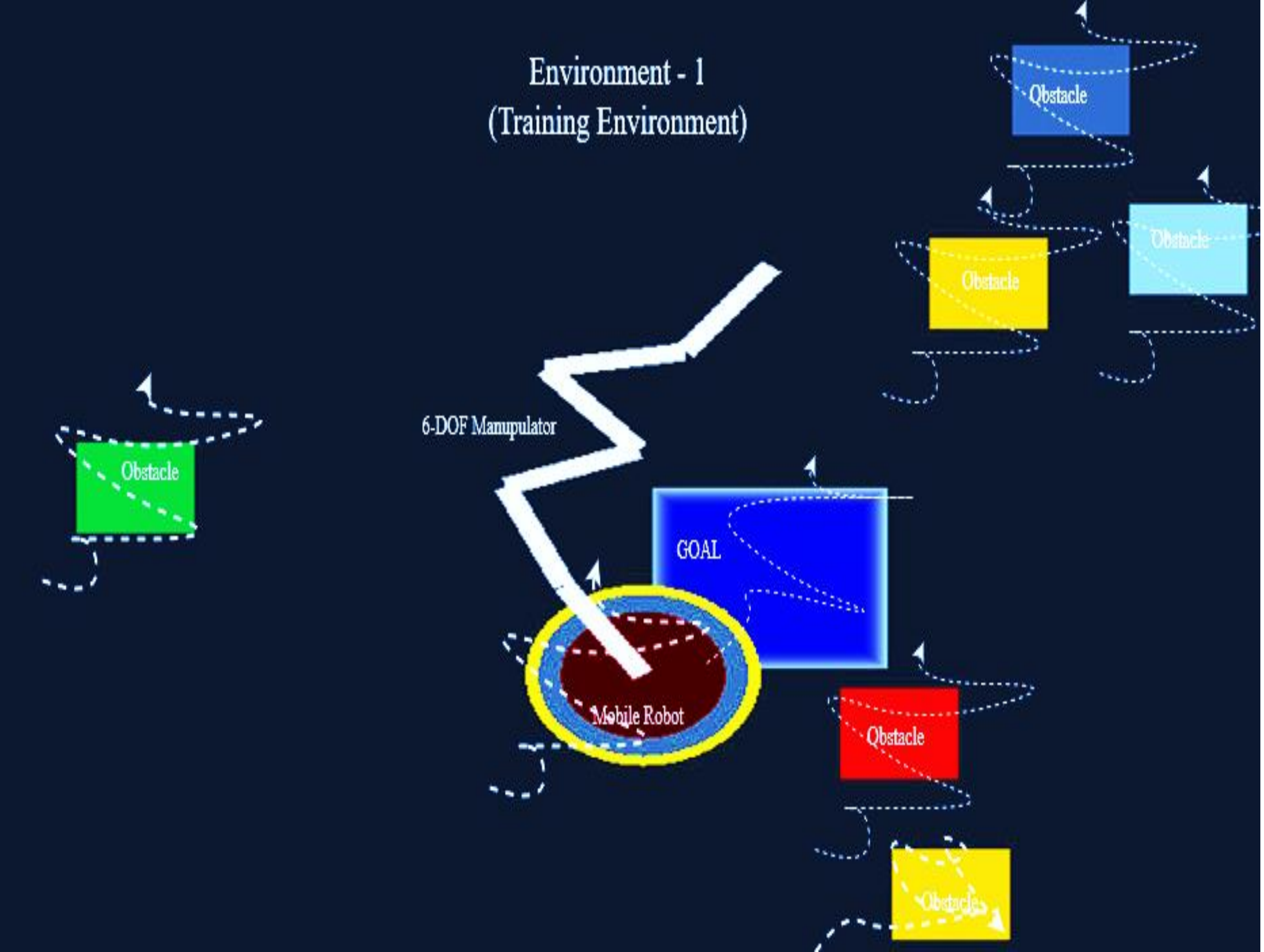
Reward_MR_OBST = Reward_MR_OBST − 1 (if a mobile robot touches the closest nearby obstacle)



- Simulation environment

- An apple orchard
- Task: Harvesting
  1. Picking up fallen apples
  2. Avoid collision with trees, human workers, and other objects

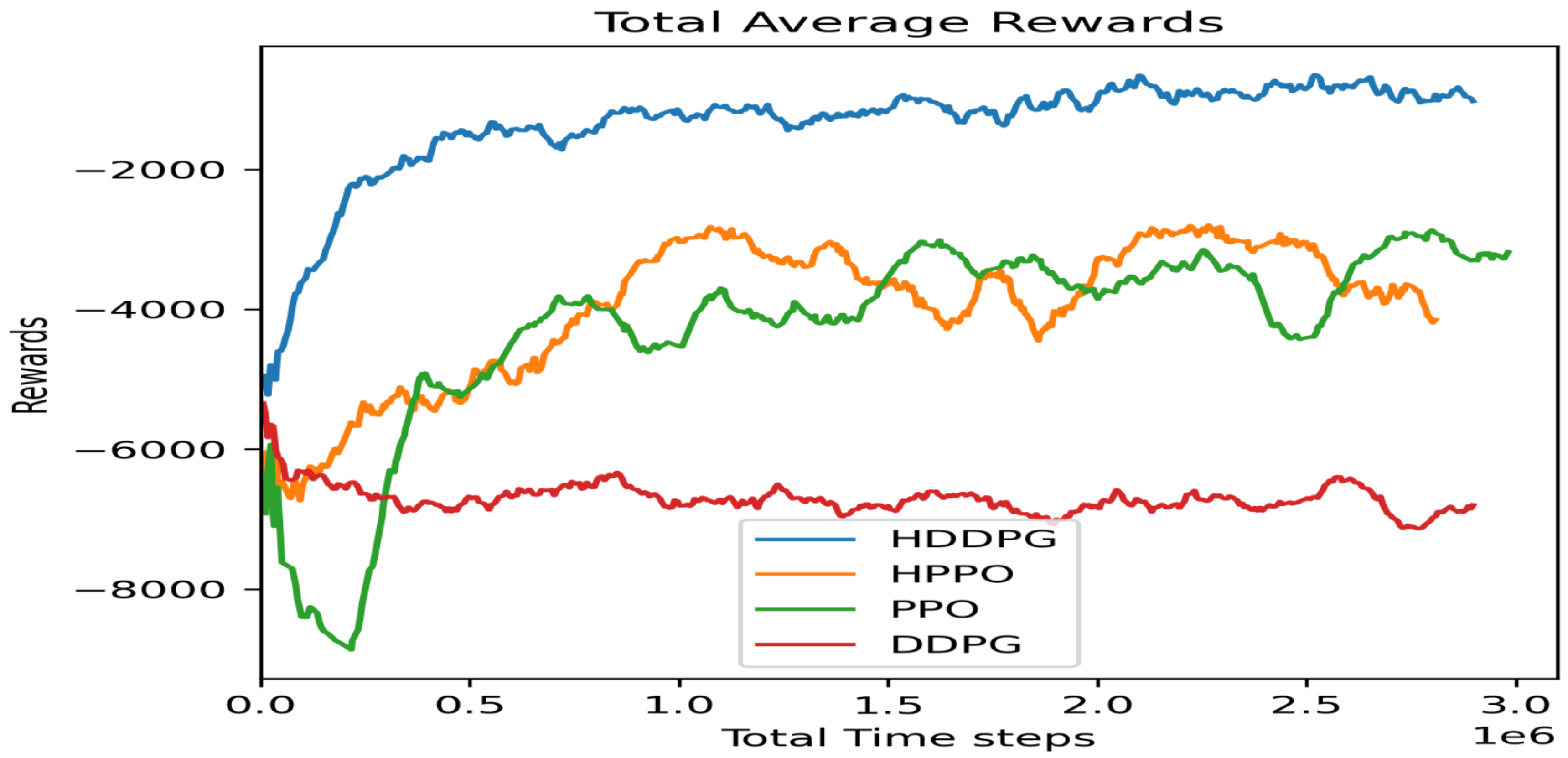

Obstacle
Obstacle
Obstacle
Obstacle
Mobile Robot shown in Concentric Circles, 6-DOF Manipulator mounted on top of it.
Obstacle



Training

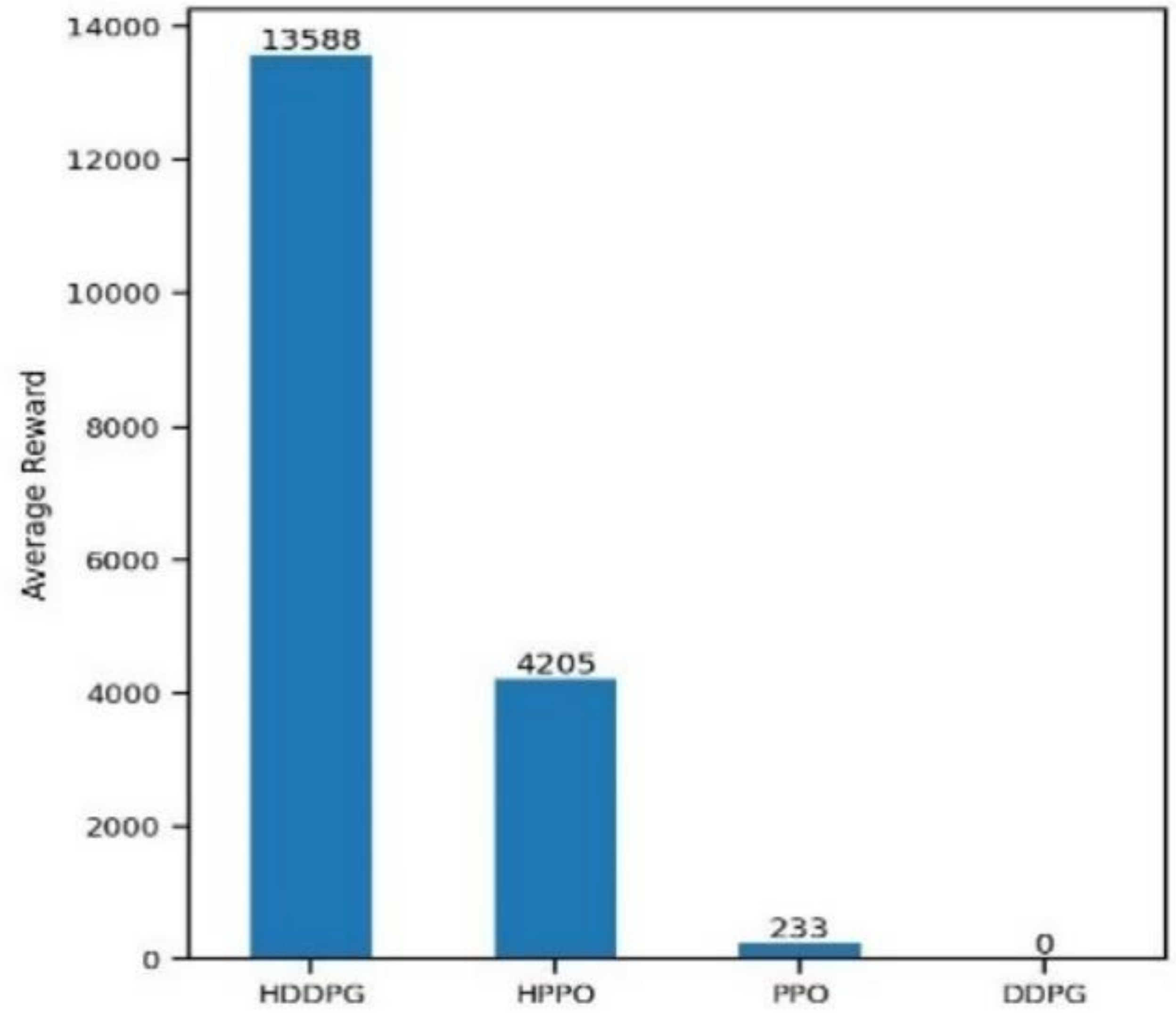
- **Sensitive skin equipped 6 DOF robotic manipulator + mobile platform**
- **Manipulator**
  - **Worker #1**
    - **Goal touching**
  - **Worker #2**
    - **Collision avoidance**
- **Mobile robot**
  - **Worker #3**
    - **Goal touching**
  - **Worker #4**
    - **Collision avoidance**

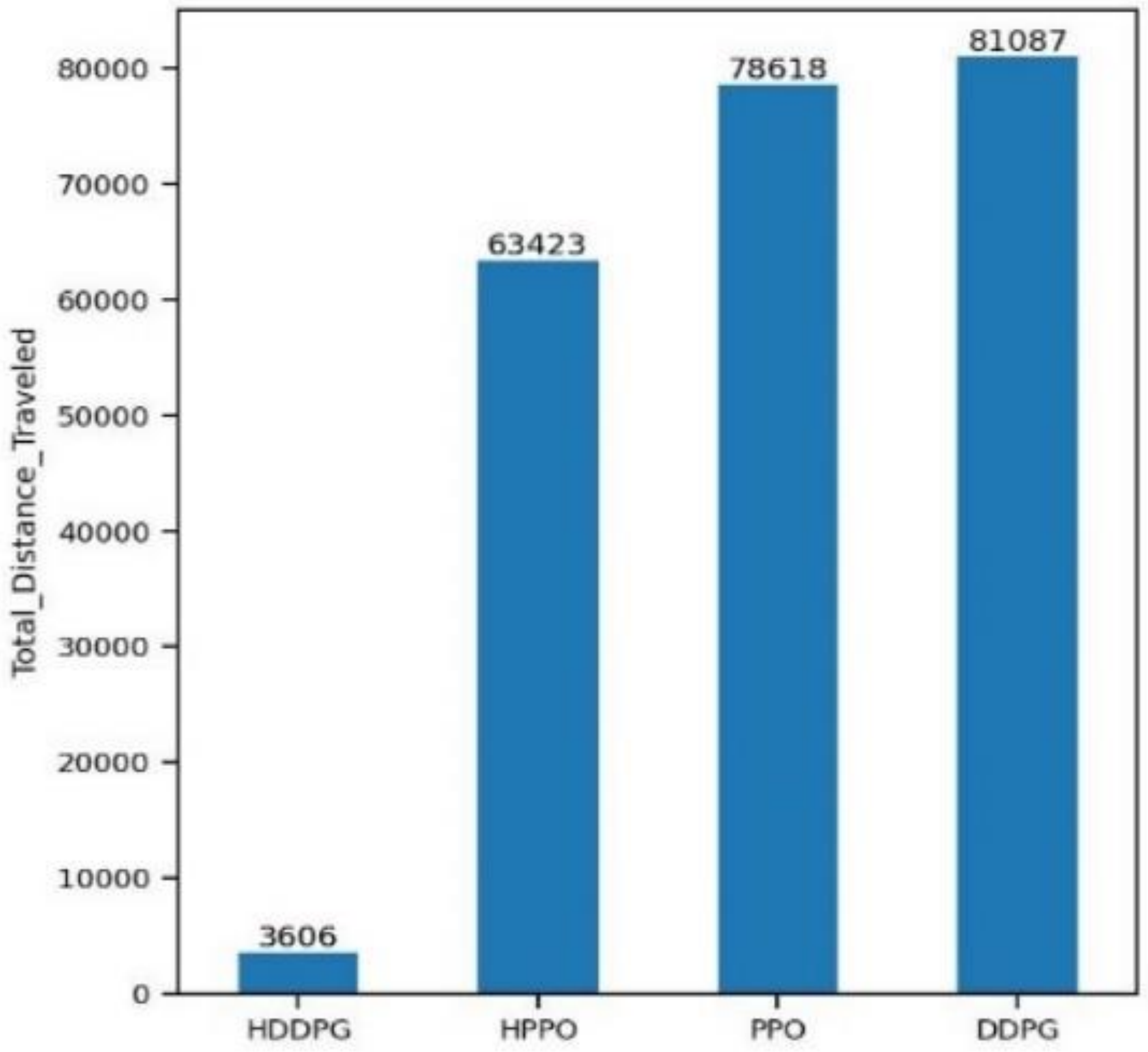
RESULTS



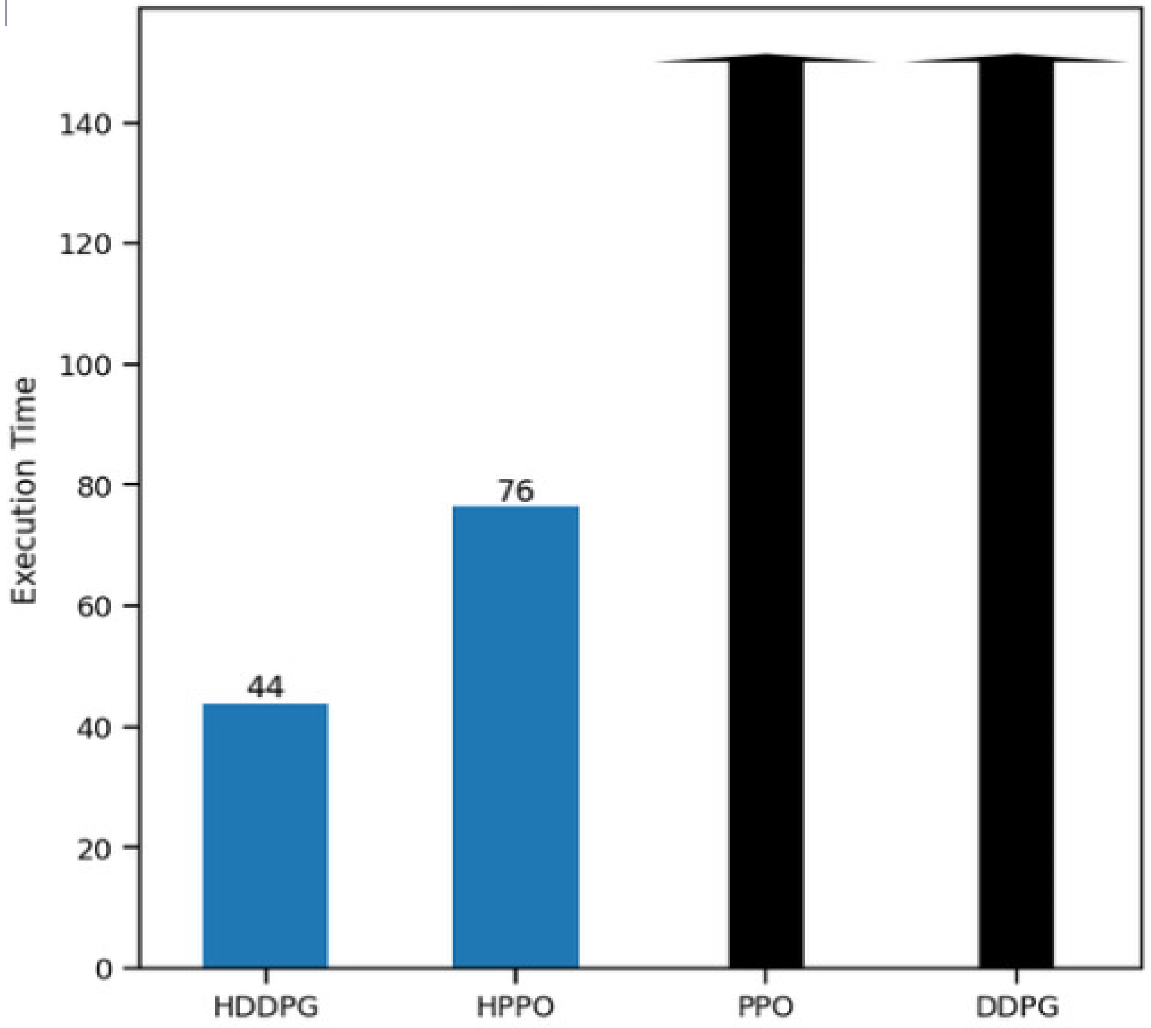
https://tensorboard.dev/experiment/eo3qO5a4Td2rBGqPTKEjRQ/#scalars

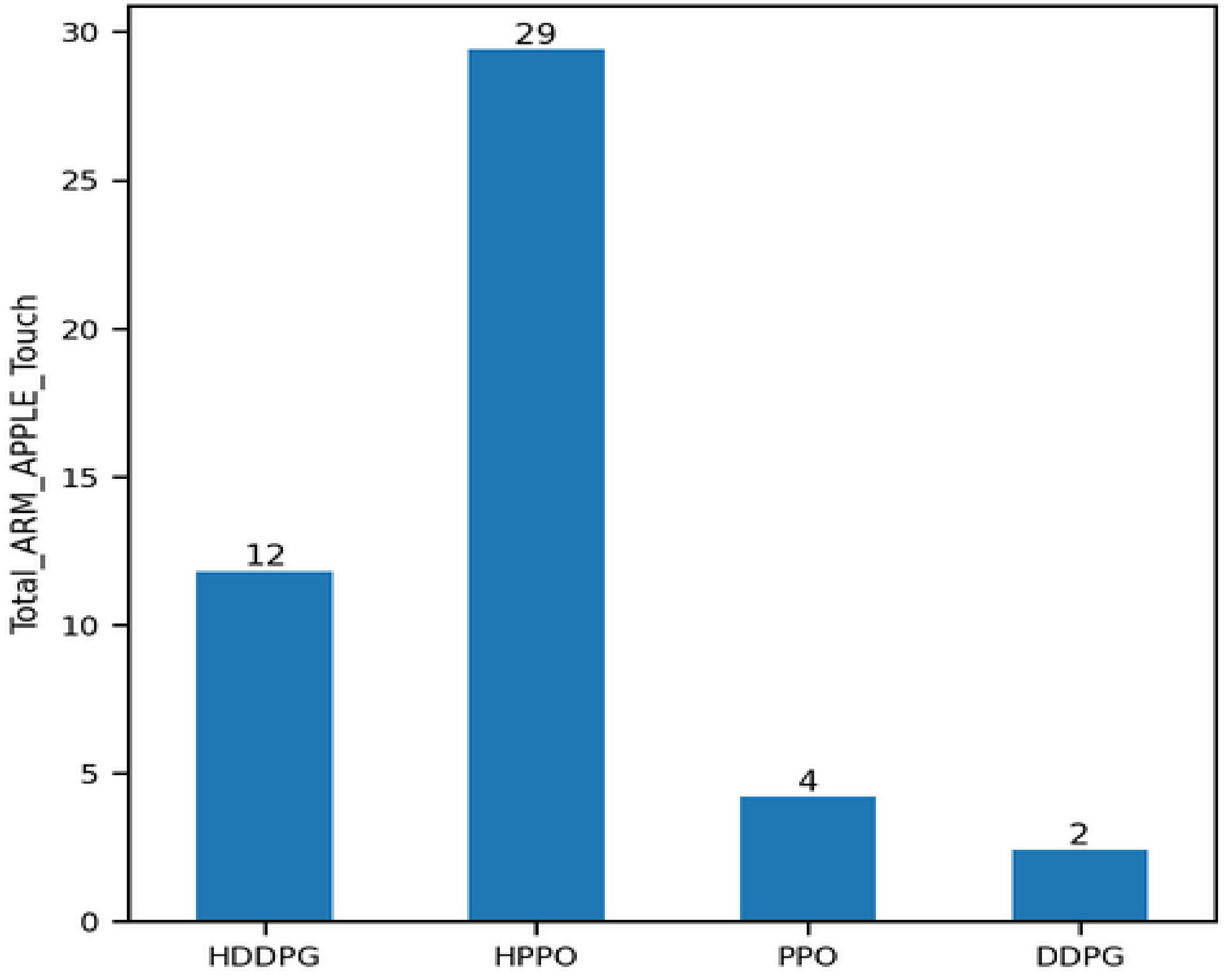
Harvesting Task Results





Harvesting Task Results





Important Observations

- Hierarchical structure allows for more flexible individual training of each model, resulting in improved performance.
- The collaboration of two agents with distinct objectives to fulfill their primary objective.
- The customizability option enables the replacement of the model with the best performing model, resulting in significant time savings during training.
- The hierarchical design's structure facilitates the addition of more functionality by expanding the number of sub-worker models to meet the required number of subtasks.

Conclusion

- A Hierarchical Reinforcement Learning (HRL) architecture is studied as a complex adaptive system to take complex environment
- The novel HRL architecture using manager-worker is implemented with the goal of expanding a reinforcement learning scheme towards a generalized AI through flexibility and expandability.
- Manager-worker structure demonstrated the ability of potential complex event horizon solver
- The proposed HDDPG and HPPO are compared with PPO and DDPG algorithms for performance evaluation. The results show the HDDPG demonstrated superior performance in reward gain, travel distance, and execution time.

Reference

1. *"Insights from Complexity Theory: Understanding Organisations better". by Assoc. Prof. Amit Gupta, Student contributor – S. Anish, IIM Bangalore. Retrieved 1 June 2012.*
2. *^ Jump up to:[a] [b] "Ten Principles of Complexity & Enabling Infrastructures". by Professor Eve Mitleton-Kelly, Director Complexity Research Programme, London School of Economics. CiteSeerX 10.1.1.98.3514.*
3. *^ Miller, John H., and Scott E. Page (1 January 2007). Complex adaptive systems : an introduction to computational models of social life. Princeton University Press. ISBN 9781400835522. OCLC 760073369.*
4. *^ Jump up to:[a] [b] "Evolutionary Psychology, Complex Systems, and Social Theory" (PDF). Bruce MacLennan, Department of Electrical Engineering & Computer Science, University of Tennessee, Knoxville. eecs.utk.edu. Retrieved 25 August 2012.*
5. *^ Foster, John (2006). "Why is economics not a complex systems science?" (PDF). Journal of Economic Issues.* ***40*** *(4): 1069–1091. doi:10.1080/00213624.2006.11506975. S2CID 17486106. Retrieved 18 January 2020.*
6. Um. D, Prasad. N, and Hocheol. S. 2022. "Hierarchical DDPG for Manipulator Motion Planning in Dynamic Environments" AI 3, no. 3: 645-658.