Foraging in Replenishing Patches

Group 2

End Term Presentation

CS698R 2021-22

Members : Abhay (180014), Ankur Gupta (180109), Anshul Rai (180117), Archit Bansal (180134)
TA Mentors : Mr. Kshitij Kumar, Mr. Gagesh Madaan
Course Instructor : Prof. Ashutosh Modi
Github : [Link] (Private Repo)

The Problem

Aim

Helping the gatherer collect maximum amount of berries in given time

Conditions

- Only 4 bushes have berries
- On harvesting a rewarding bush:
  - Berries on that bush decrease
  - Berries on other rewarding bush replenish
- Each action takes some time to complete

http://trap-lining-v2.s3-website.ap-south-1.amazonaws.com/

Motivation

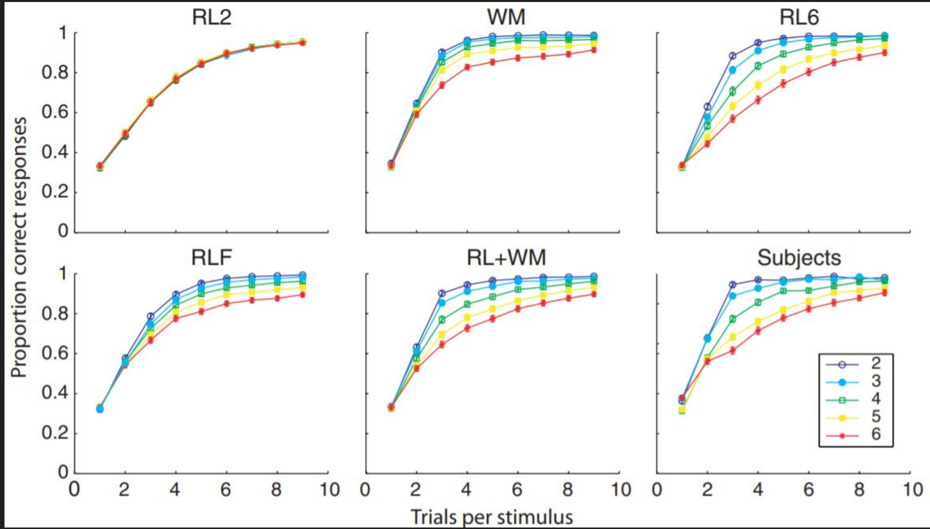
- Foraging is an important task done by all animals
    - Can also be seen in nomadic people and migratory animals
- Very relevant to neuroscience research and understanding the role of working memory
- This is a realistic environment with decreasing rewards and replenishment

https://www.nytimes.com/2009/12/22/science/22obstarch.html ; https://en.wikipedia.org/wiki/Foraging

Literature Review

How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis

- Made 5 models to understand the role of working memory
  - RL model (Monte Carlo with softmax decision function)
  - RL model with multiple learning rate
  - Forgetful RL model
  - Working Memory model
  - Reinforcement Learning + WM model
- Learning curve of different models compared to human



https://onlinelibrary.wiley.com/doi/10.1111/j.1460-9568.2011.07980.x

Reinforcement Learning Signals in the Human Striatum Distinguish Learners from Non-learners during Reward Based Decision Making

- Used RL model to determine the neural basis of difference in performance of humans in reward based decision based tasks.
- Experimented through a four armed bandit task, each with different probabilities of success.
- Modelled human behaviour using a choice recency model which encapsulates the reward and the previous choice.
- To capture low-order autocorrelation in the choices (link), an index c_i is maintained for each bandit i, which track how recently it had been chosen, and allowed this to bias choices.

https://www.jneurosci.org/content/jneuro/27/47/12860.full.pdf

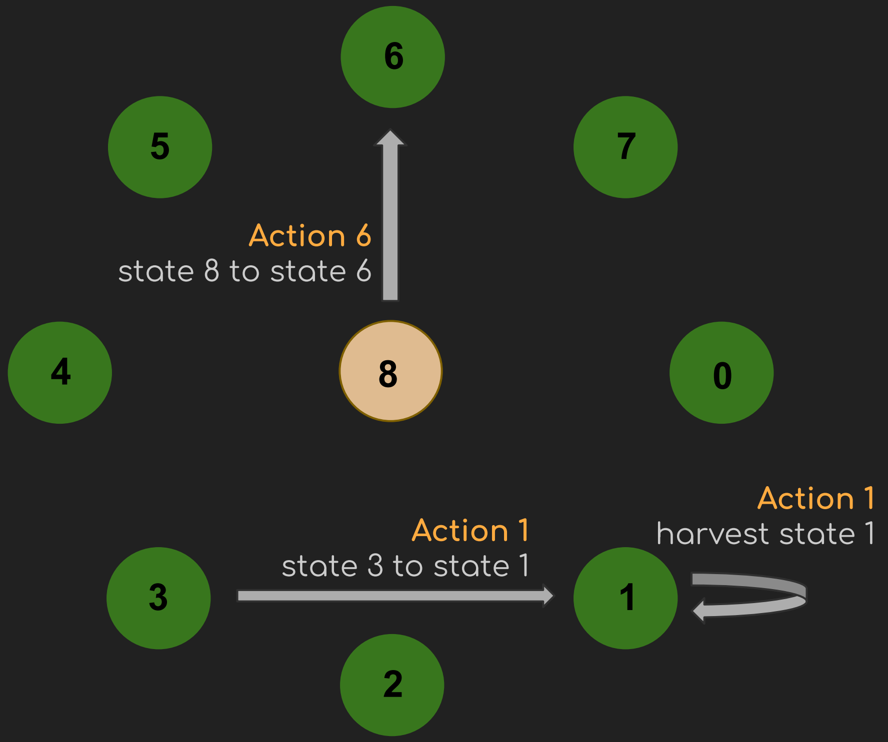
Environment Details

Basic Setup

- 4 out of 8 bushes are rewarding
- Each block is played for 300 seconds

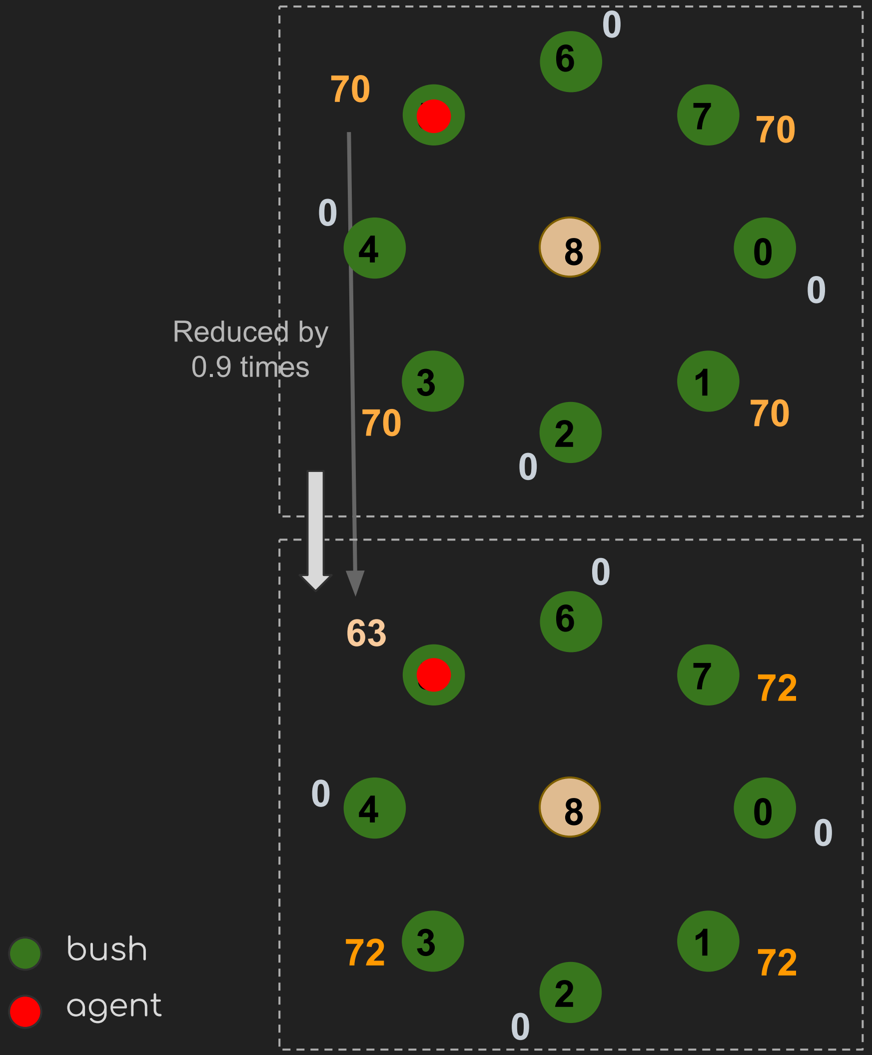
States and Action

- There are 9 total states
- State 8 is our initial state and once agent leaves it can't come back
- The actions are moving to any state labelled 0-7 and harvesting at the present state



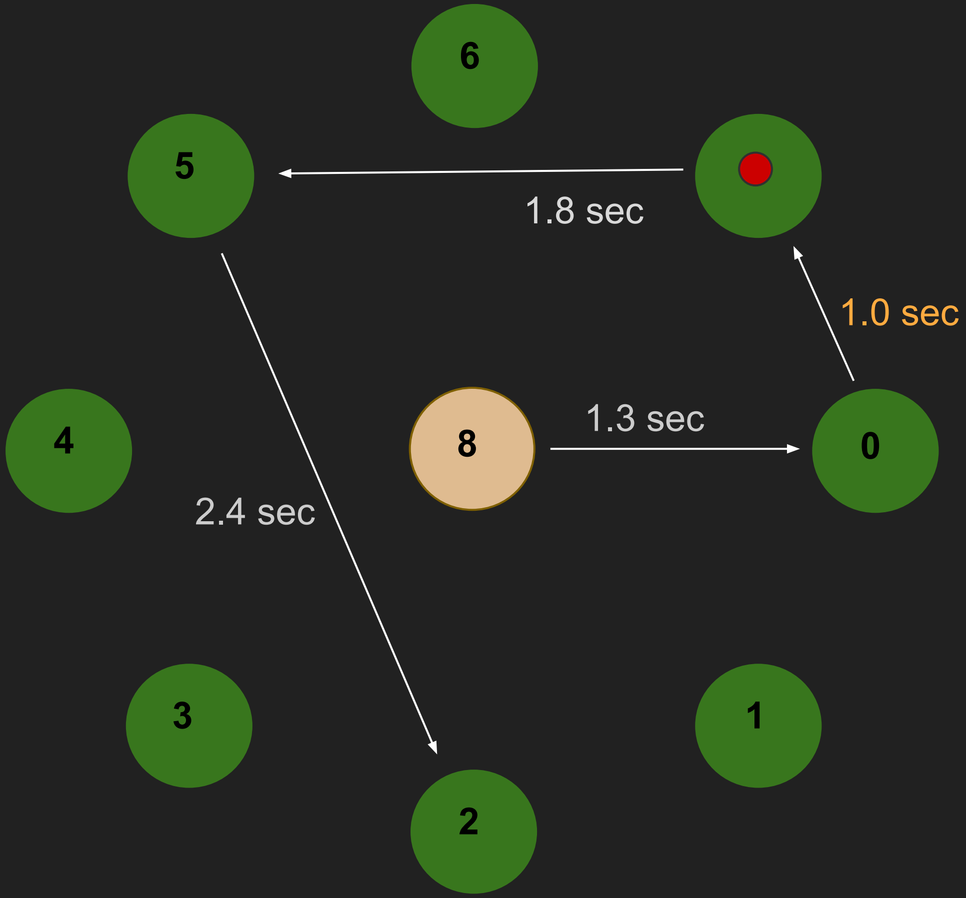
Reward Updation

- Only 4 bushes have reward which can grow to a max value of 200
- Whenever we mine a bush currently having a reward r
  - $\text{Collected Reward} += floor(0.9 \cdot r)$
  - $r = floor(0.9 \cdot r)$
  - When we mine a bush having reward all the other bushes with reward are replenished according to their replenishment rate



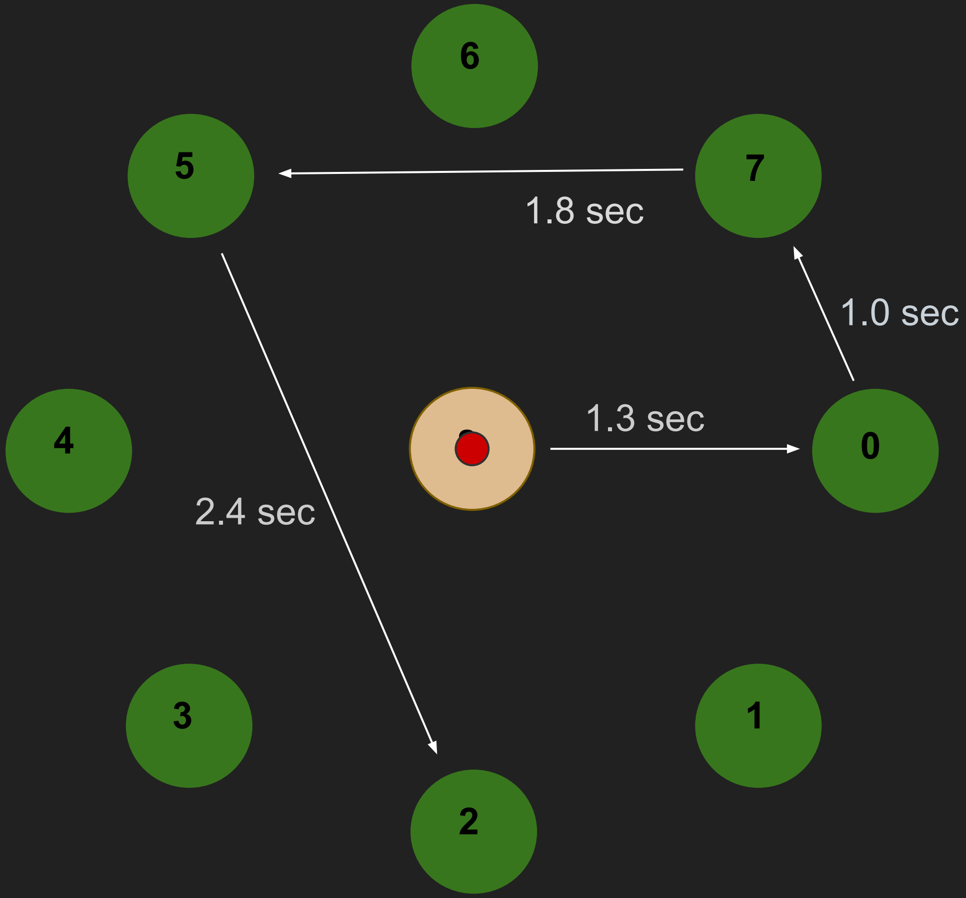
Time Cost for Actions

- Every Action takes a predetermined amount of time.
- Harvesting takes 1 second



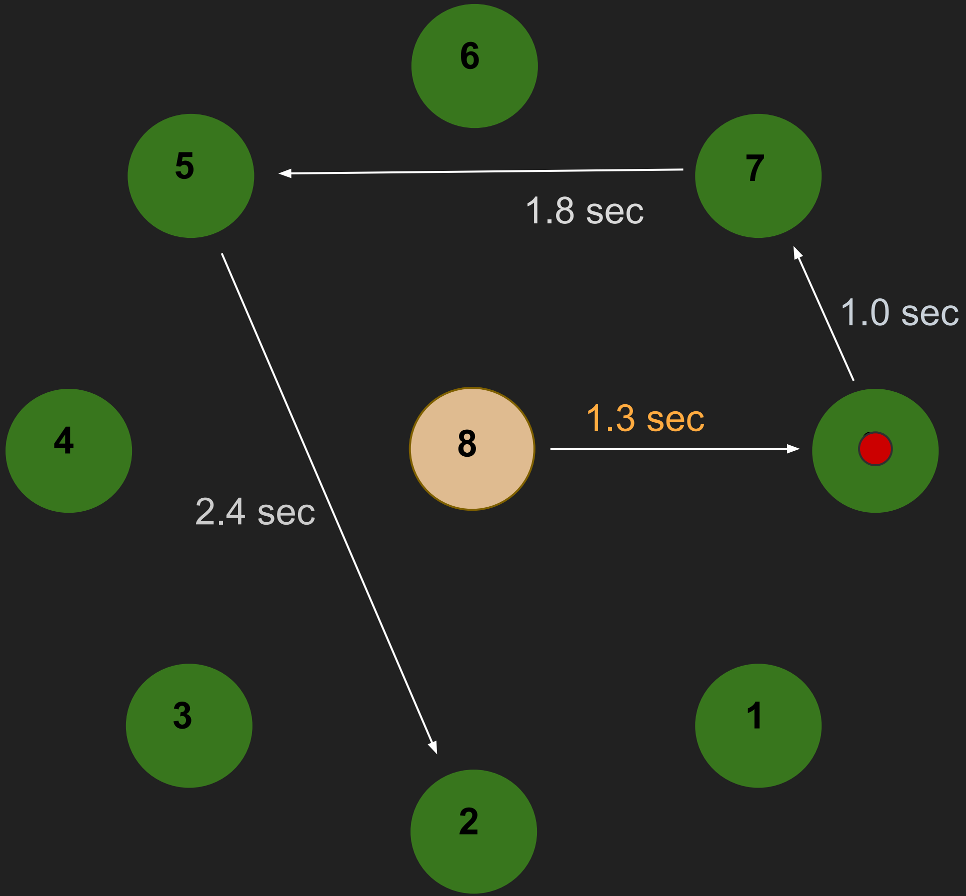
Time Cost for Actions

- Every Action takes a predetermined amount of time.
- Game ends after 300 seconds
- Harvesting takes 1 second



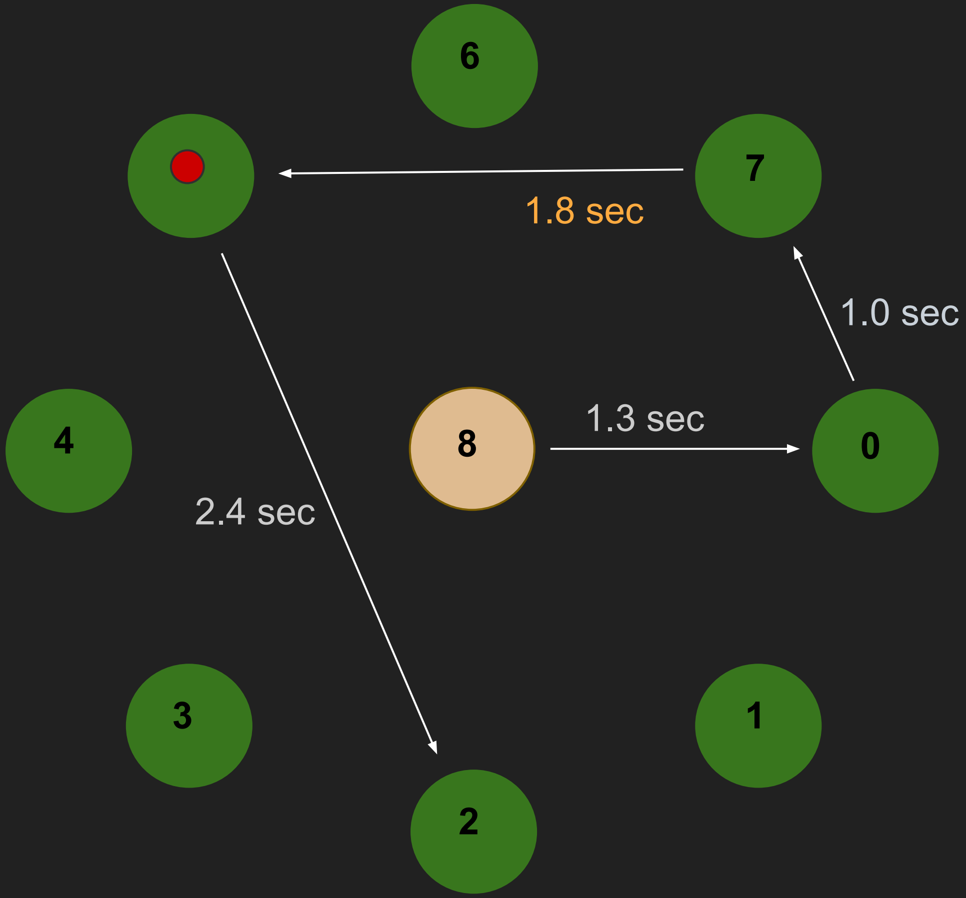
Time Cost for Actions

- Every Action takes a predetermined amount of time.
- Harvesting takes 1 second



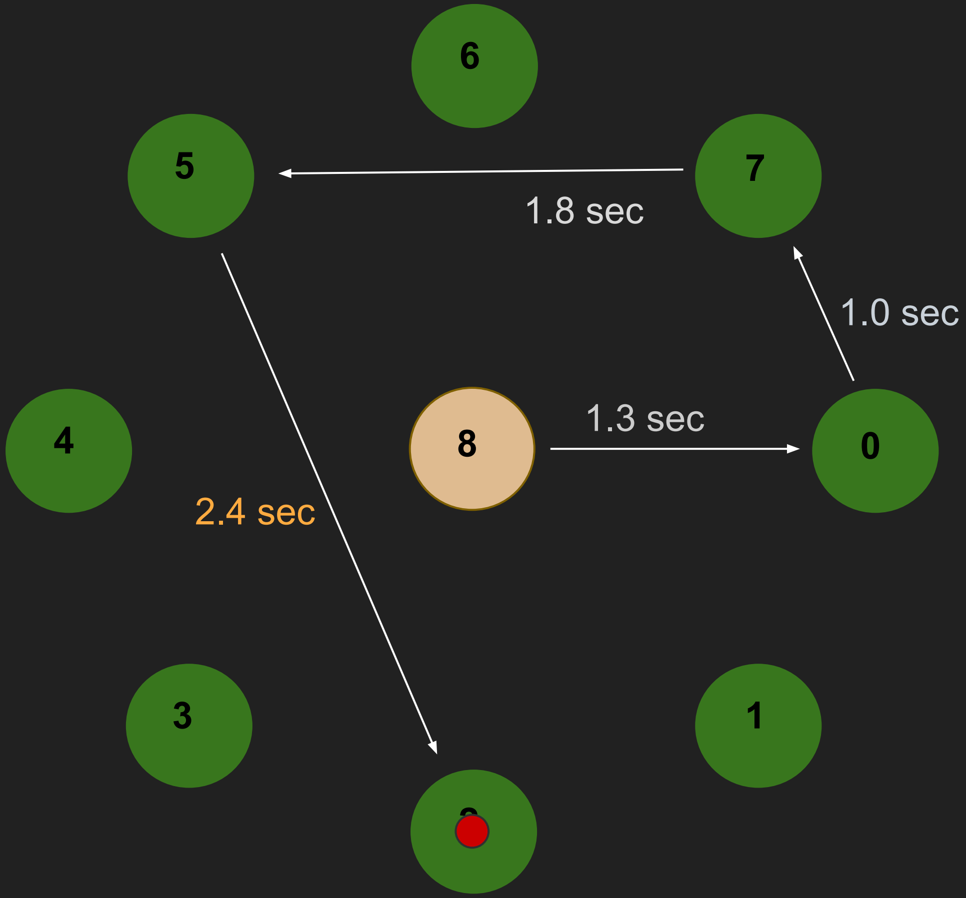
Time Cost for Actions

- Every Action takes a predetermined amount of time.
- Harvesting takes 1 second



Time Cost for Actions

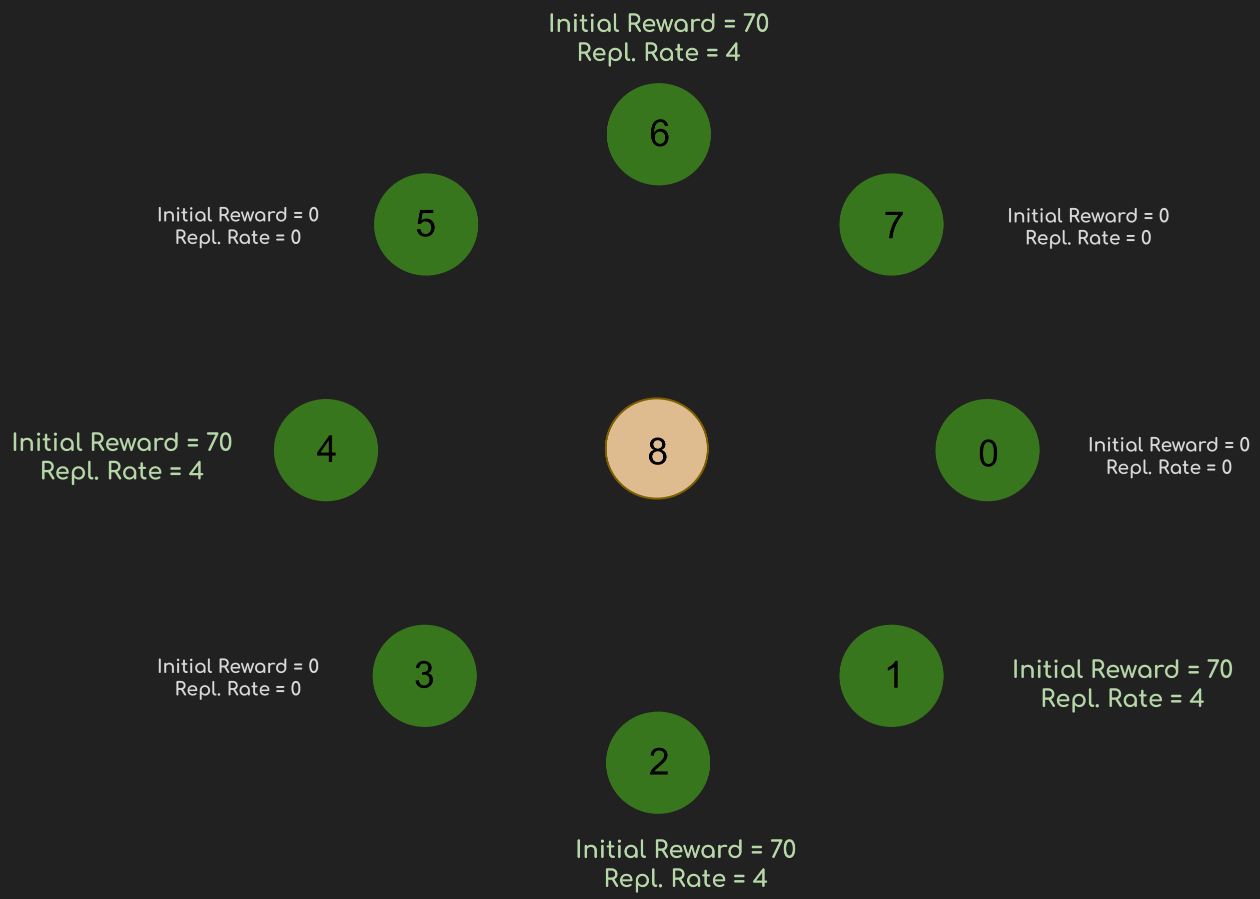
- Every Action takes a predetermined amount of time.
- Harvesting takes 1 second



Different Block Diagrams

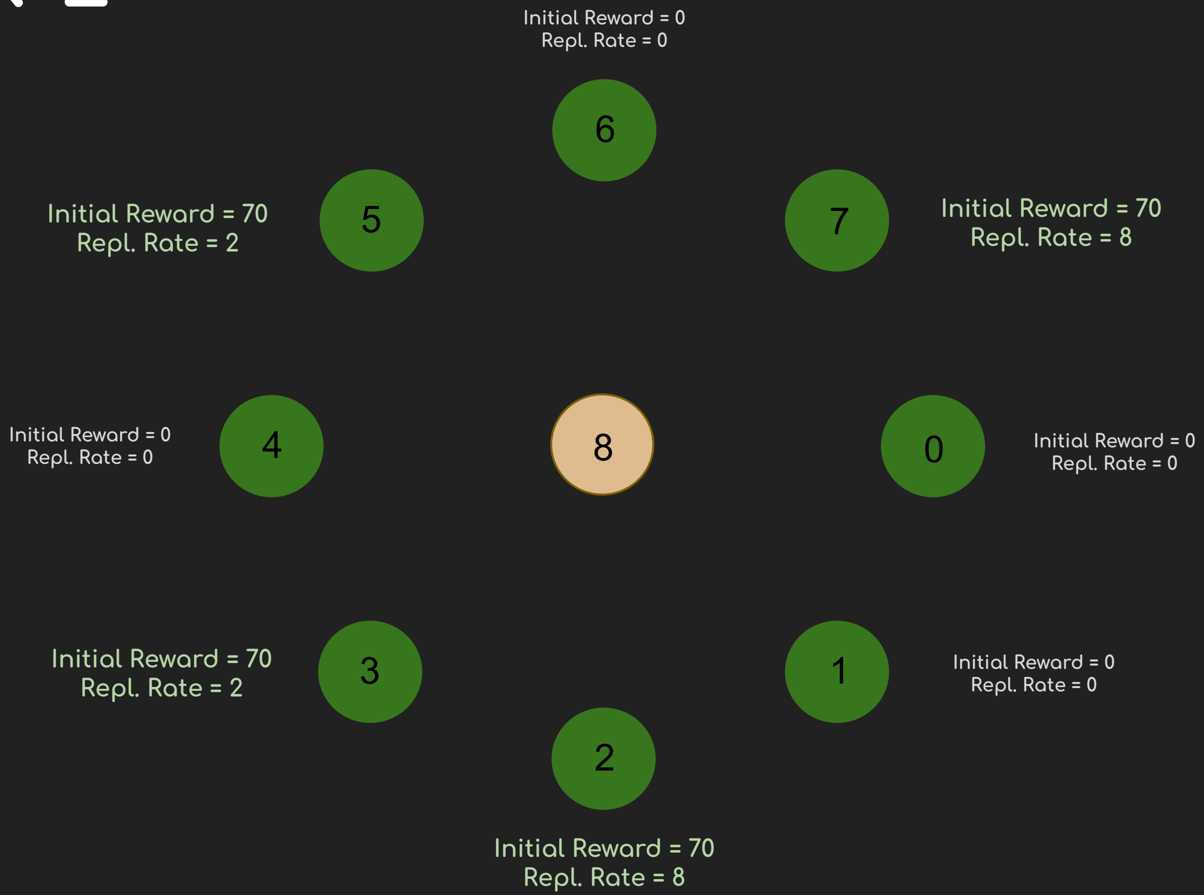
- 3 types of blocks
  - Constant replenishing rate
  - 2 different replenishing rates
  - 4 different replenishing rates
- Different types of blocks to understand importance of WM

Block-1

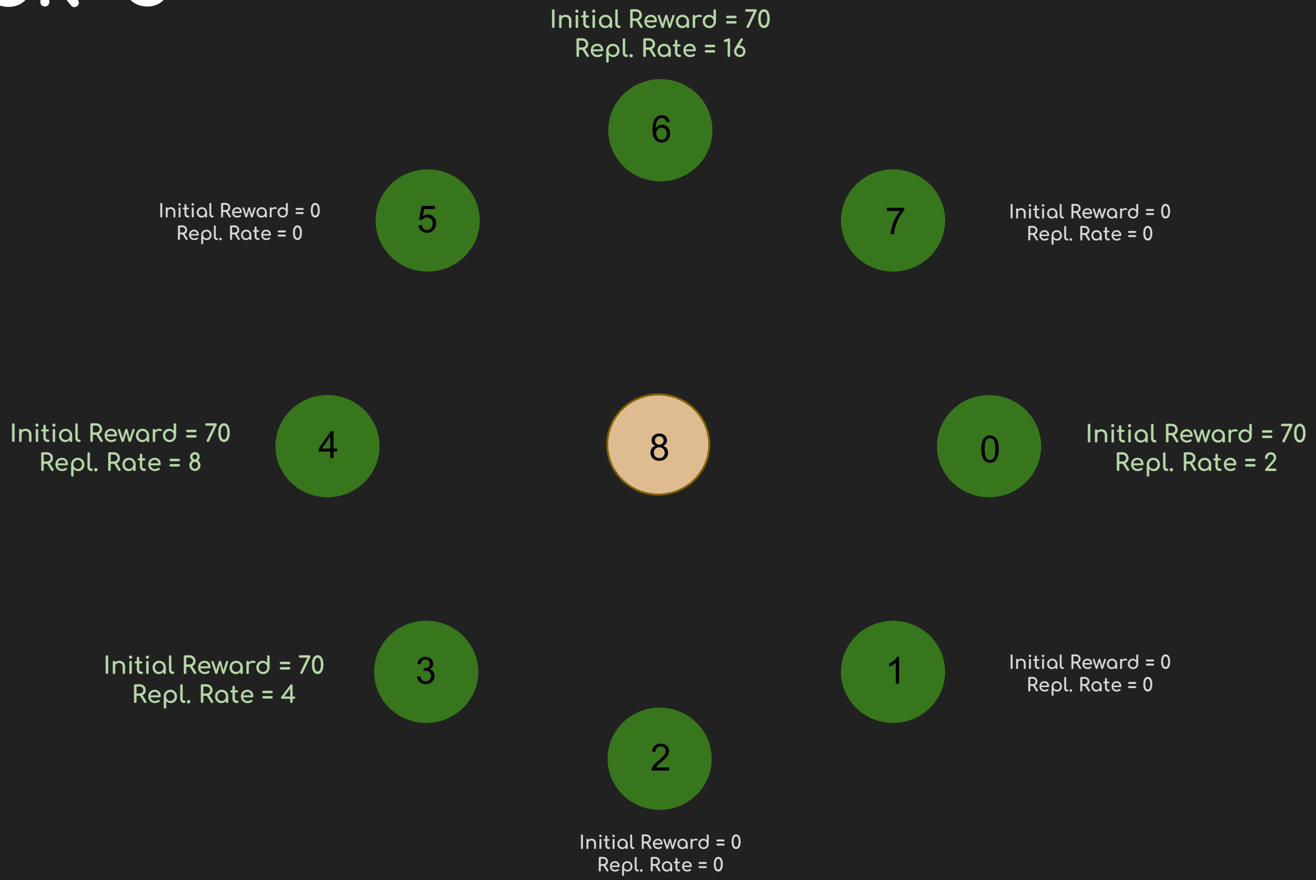


Initial Reward = 70
Repl. Rate = 4
6
Initial Reward = 0
Repl. Rate = 0
5
7
Initial Reward = 0
Repl. Rate = 0
Initial Reward = 70
Repl. Rate = 4
4
8
0
Initial Reward = 0
Repl. Rate = 0
Initial Reward = 0
Repl. Rate = 0
3
1
Initial Reward = 70
Repl. Rate = 4
2
Initial Reward = 70
Repl. Rate = 4

Block-2


Initial Reward = 0
Repl. Rate = 0
6
Initial Reward = 70
Repl. Rate = 2
5
7
Initial Reward = 70
Repl. Rate = 8
Initial Reward = 0
Repl. Rate = 0
4
8
0
Initial Reward = 0
Repl. Rate = 0
Initial Reward = 70
Repl. Rate = 2
3
1
Initial Reward = 0
Repl. Rate = 0
2
Initial Reward = 70
Repl. Rate = 8

Block-3


Initial Reward = 70
Repl. Rate = 16
6
Initial Reward = 0
Repl. Rate = 0
5
7
Initial Reward = 0
Repl. Rate = 0
Initial Reward = 70
Repl. Rate = 8
4
8
0
Initial Reward = 70
Repl. Rate = 2
Initial Reward = 70
Repl. Rate = 4
3
1
Initial Reward = 0
Repl. Rate = 0
2
Initial Reward = 0
Repl. Rate = 0

Baseline Results

- We ran SARSA algorithm to estimate Q[s][a] value
- Final Greedy Policy = [4,6,0,0,6,4,6,7,0] (a particular action for a given state)
- Only gives a reward of 517

Why do we use Q(s,r,a)?

- The major drawback of our baseline was that it did same action on a state using the final policy irrespective of the reward(feedback) that it gets from the patch about the current status of the bush's reward

- For the baseline, we had implemented Tabular RL methods having Q values of the form Q(s,a).

- This formulation of Q(s,a) doesn't support concept of decreasing and replenishing rewards of our environment.

- To incorporate these factor, we have modified our Q value function such that it takes the previous reward as an input, i.e. Q(s,r,a).

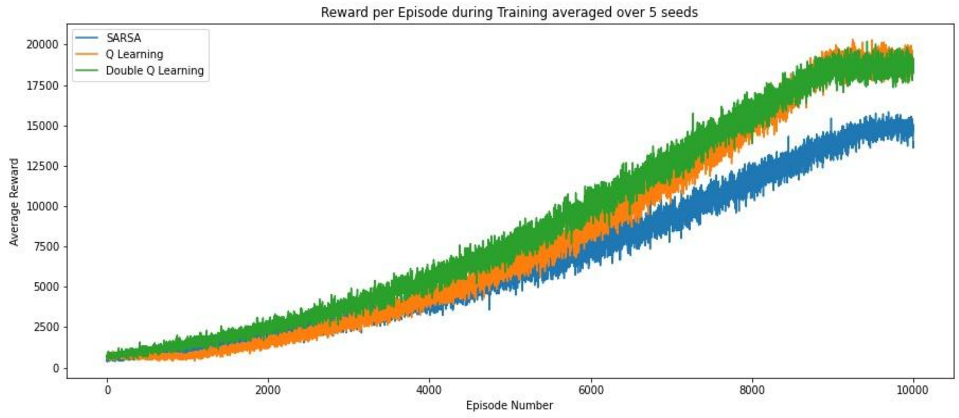
Tabular RL Methods

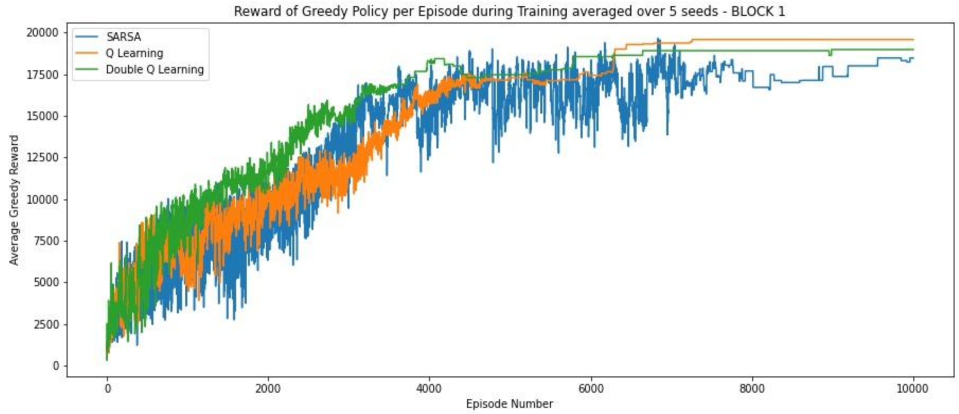
Tabular RL Models

Experiment Results on Block **1**

Algorithm	Score*
SARSA	18473.5
Q Learning	19586.8
Double Q Learning	18983.8

*The scores are averaged over multiple iterations

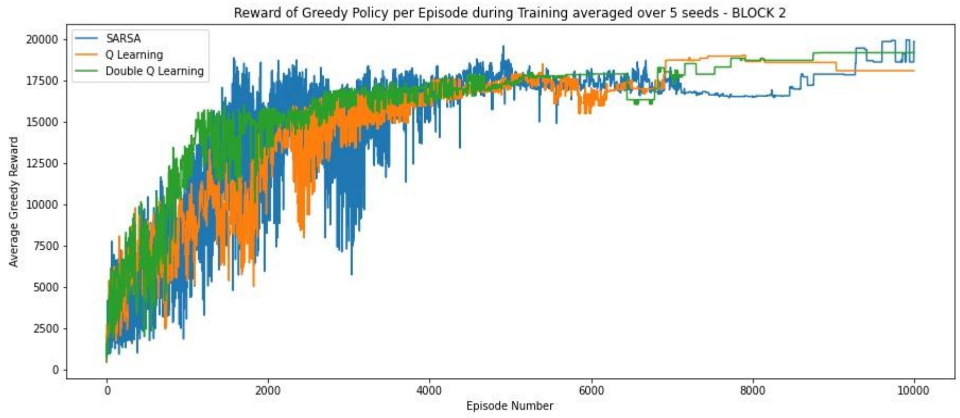


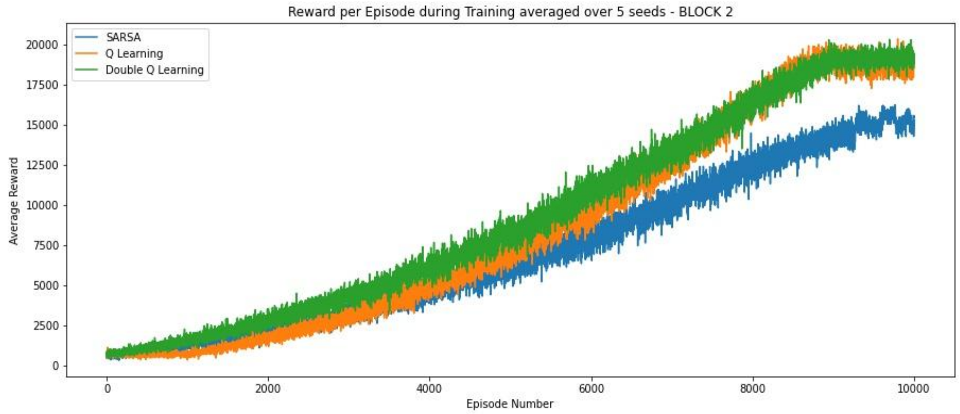


Tabular RL Models

Experiment Results on Block **2**

Algorithm	Score
SARSA	19859.5
Q Learning	18093.4
Double Q Learning	19181.6

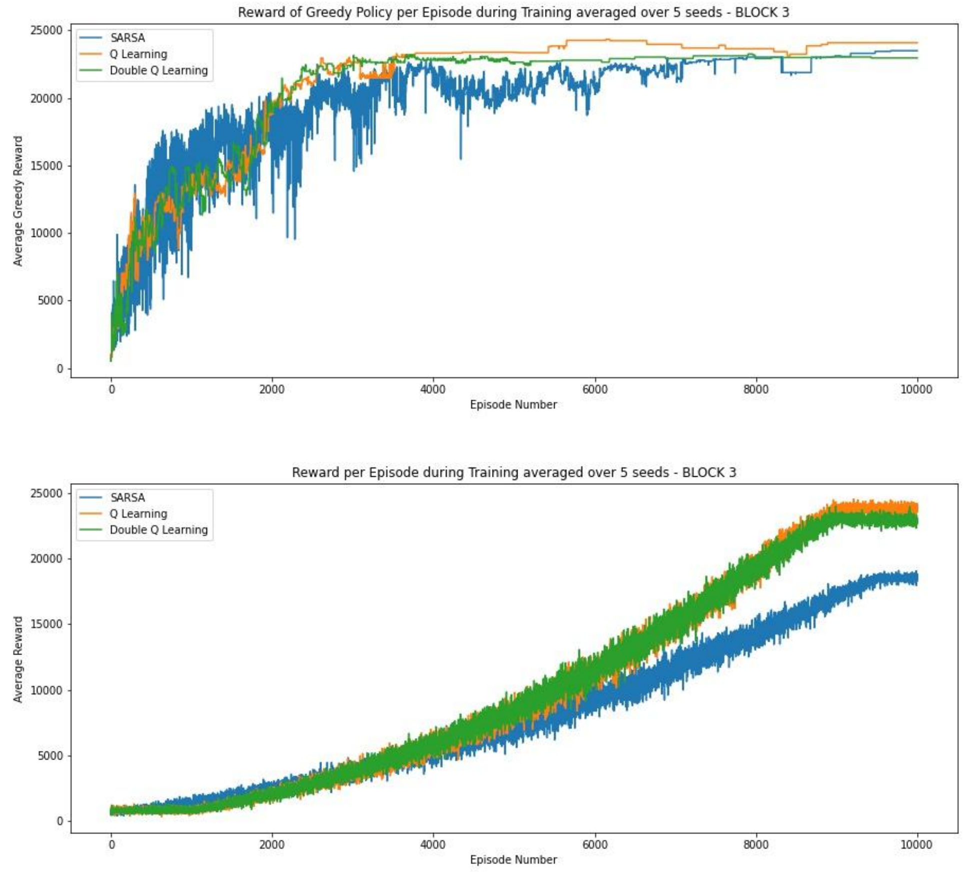




Tabular RL Models

Experiment Results on Block **3**

Algorithm	Score
SARSA	23484
Q Learning	24073.6
Double Q Learning	22949.6

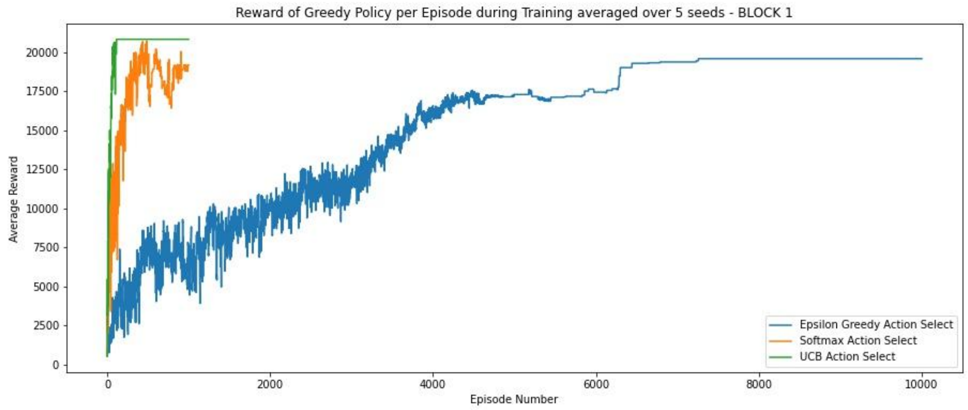


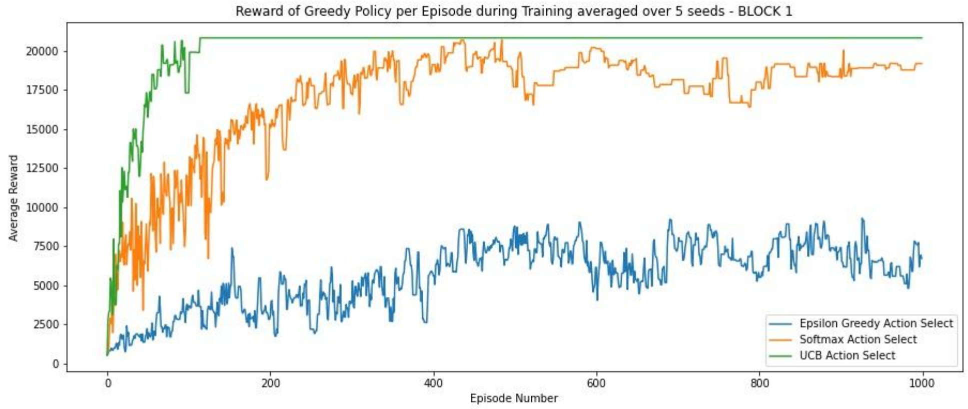
Comparing Different Action Selection Strategy

Experiment Results on Block **1**

Control Algorithm : **Q-Learning**

Algorithm	Score
Epsilon Greedy	19586.8
Softmax	19186.6
UCB	20827



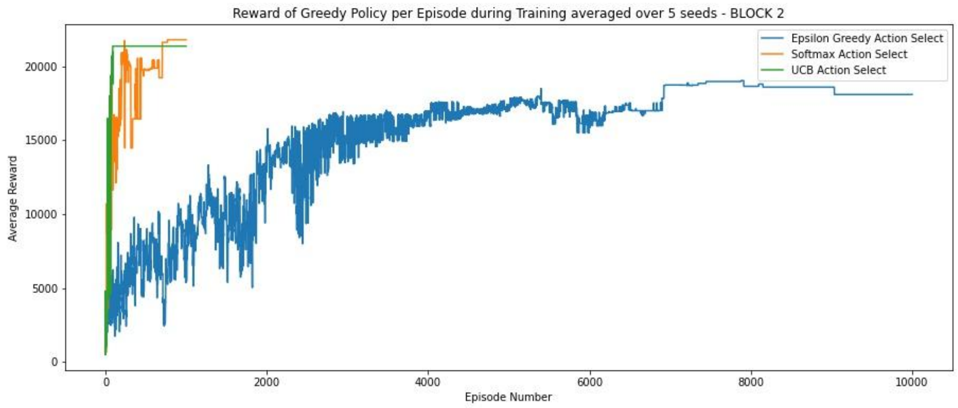


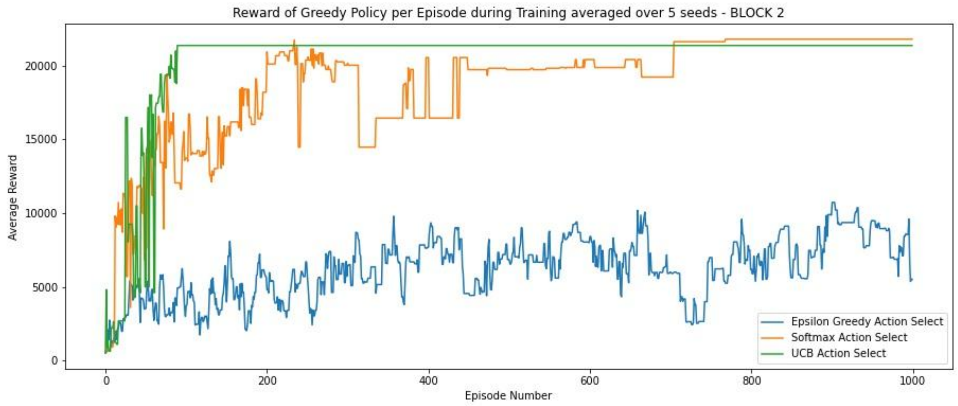
Comparing Different Action Selection Strategy

Experiment Results on Block **2**

Control Algorithm : **Q-Learning**

Algorithm	Score
Epsilon Greedy	18093.4
Softmax	21797
UCB	21366



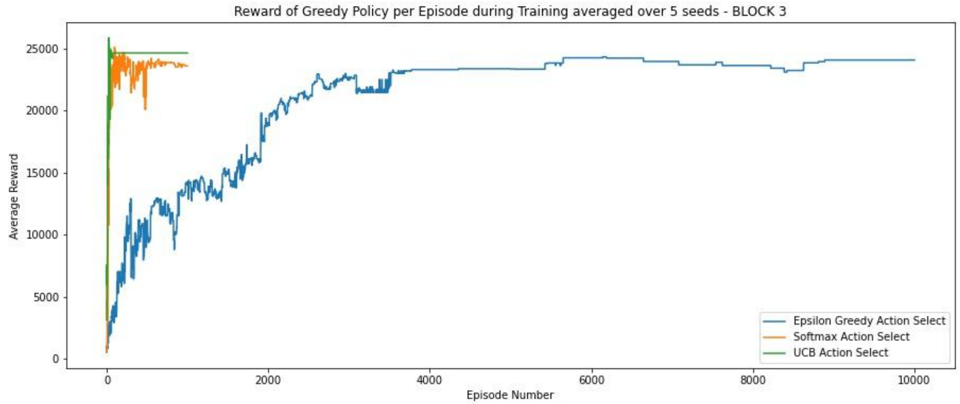


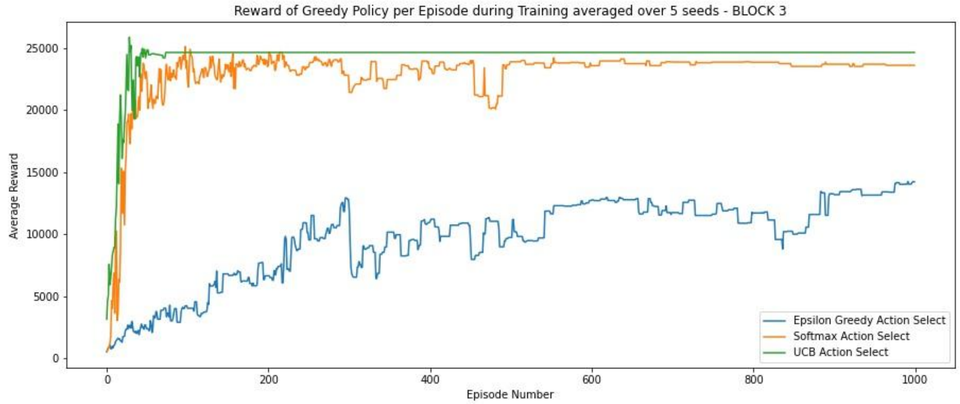
Comparing Different Action Selection Strategy

Experiment Results on Block **3**

Control Algorithm : **Q-Learning**

Algorithm	Score
Epsilon Greedy	24073
Softmax	23607
UCB	24642



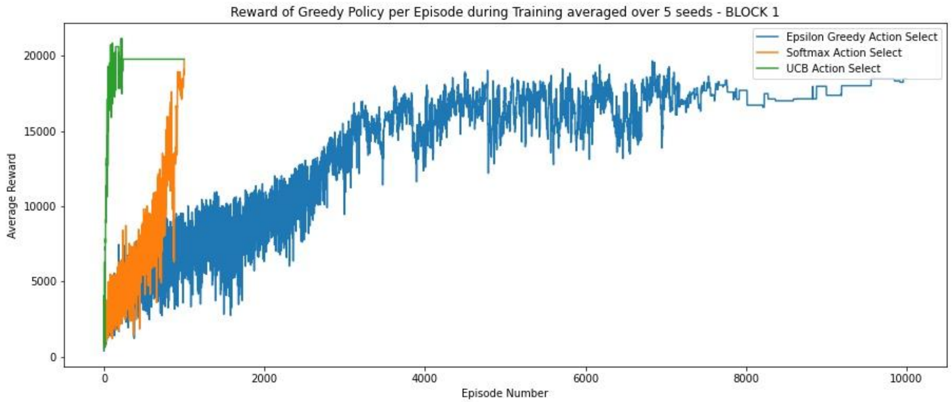


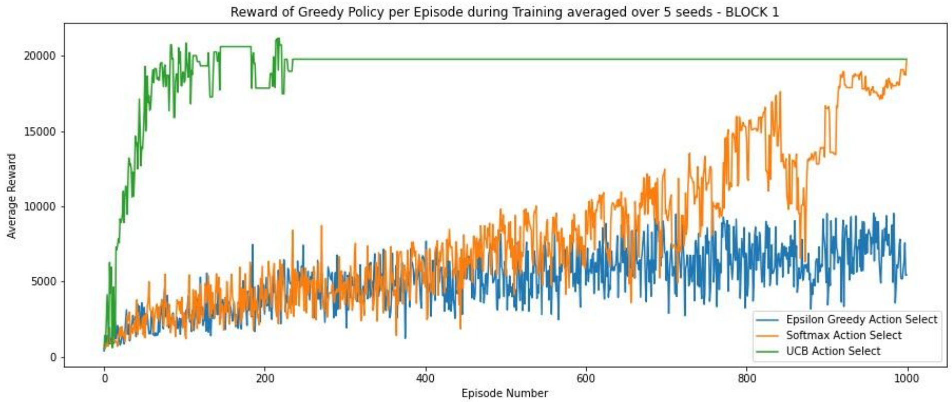
Comparing Different Action Selection Strategy

Experiment Results on Block **1**

Control Algorithm : **SARSA**

Algorithm	Score
Epsilon Greedy	18093.4
Softmax	21797
UCB	21366



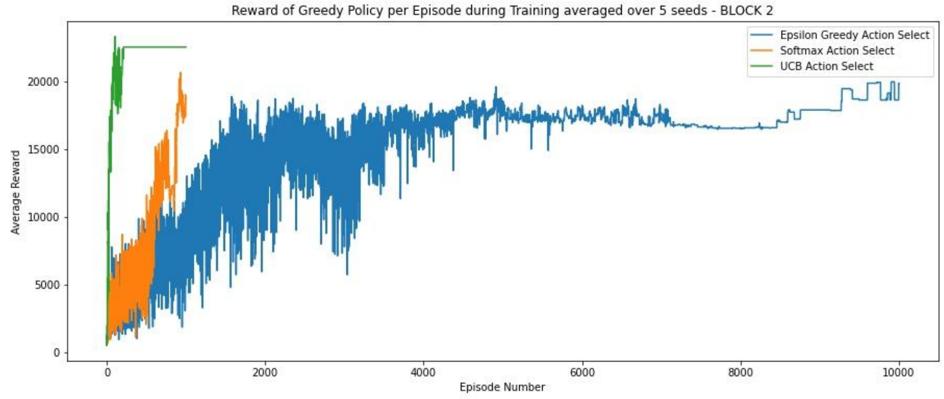


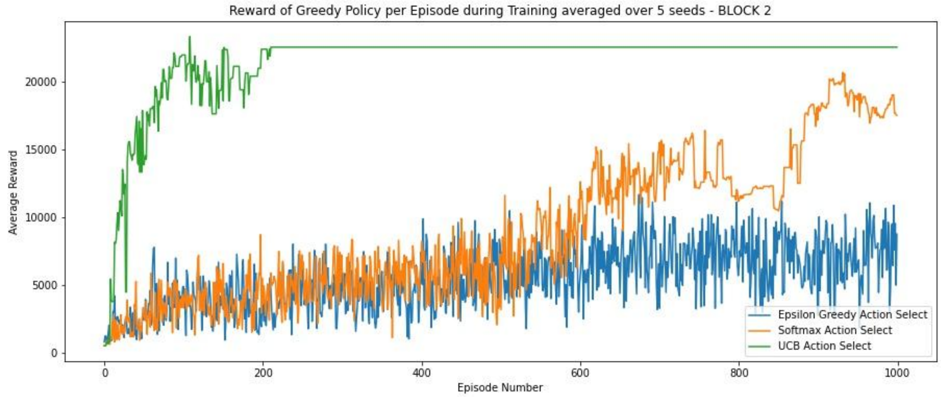
Comparing Different Action Selection Strategy

Experiment Results on Block **2**

Control Algorithm : **SARSA**

Algorithm	Score
Epsilon Greedy	18473.5
Softmax	19670.75
UCB	19769



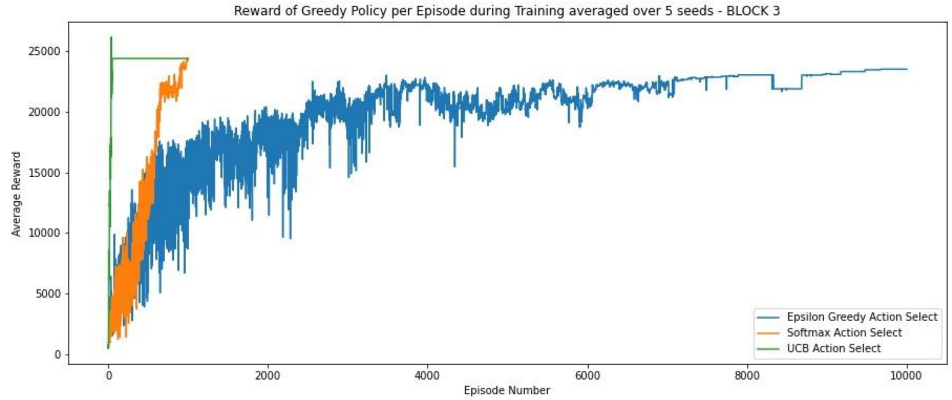


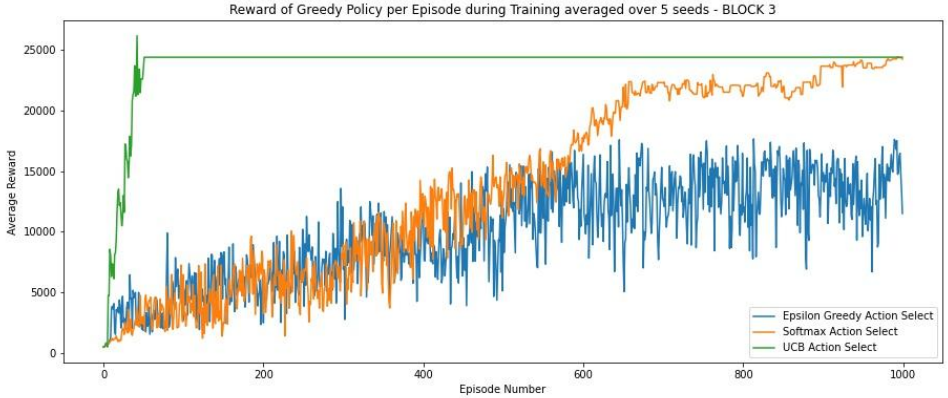
Comparing Different Action Selection Strategy

Experiment Results on Block **3**

Control Algorithm : **SARSA**

Algorithm	Score
Epsilon Greedy	23484
Softmax	24209.5
UCB	24363





Working Memory (WM)

Human Learning takes very less iterations to understand the environment but RL agent performs very poorly in the starting iterations

→

Humans have a working memory which can be used to store limited(6 to 9) and recent things ⟶ Useful to remember which bushes are rewarding and their replenishing rates

→

We imitate human learning by creating a WM decision module which works by estimating the **replenishing rate**

$$EstimatedReward = lastReward + replenishingRate \cdot timeElapsed$$

$$estimatedRepRate = (reward - lastReward)/timeElapsed$$

$$repRate = 0.6 \cdot repRate + 0.4 \cdot estimatedRepRate$$

❖ 4 types of experiment setting:
- Perfect Memory
- Limited Memory
- Decaying Memory
- Limited and Decaying Memory

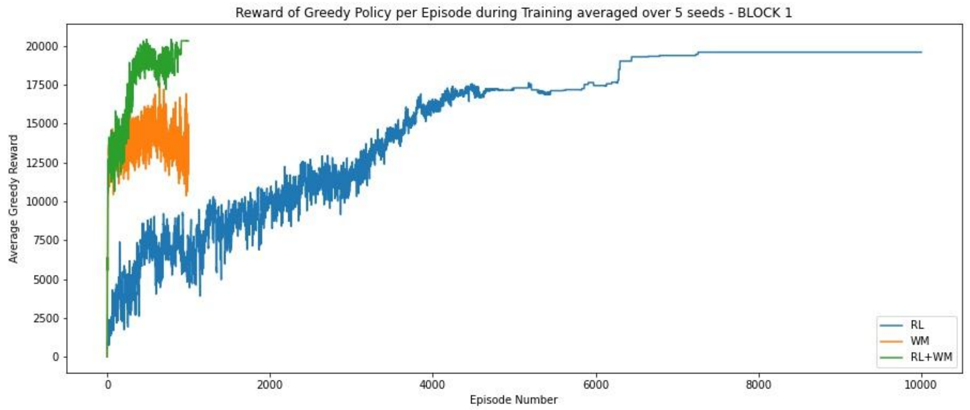
WM: Perfect

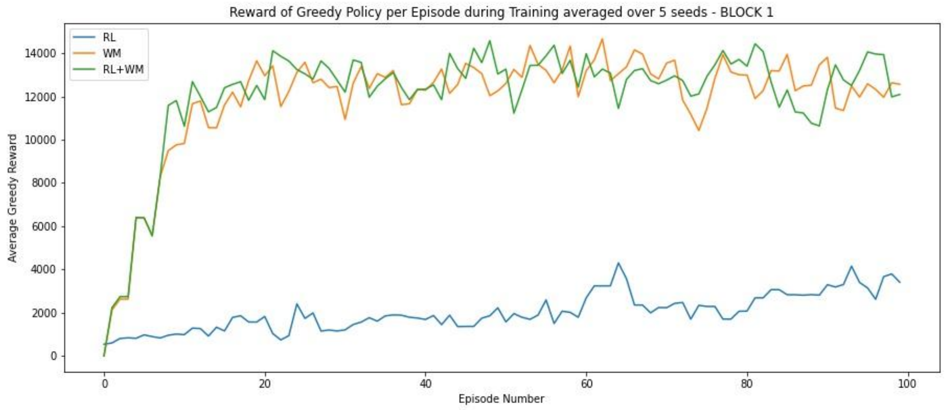
Experiment Results on Block **1**

- We assume infinite working memory
- For each bush we store:
  a. Last reward time
  b. Last received reward

  to predict replenishing rate for each bush
- 24 variables are stored in total.

Algorithm	Score
RL	19586.8
WM	14334.8
RL + WM	20309.8





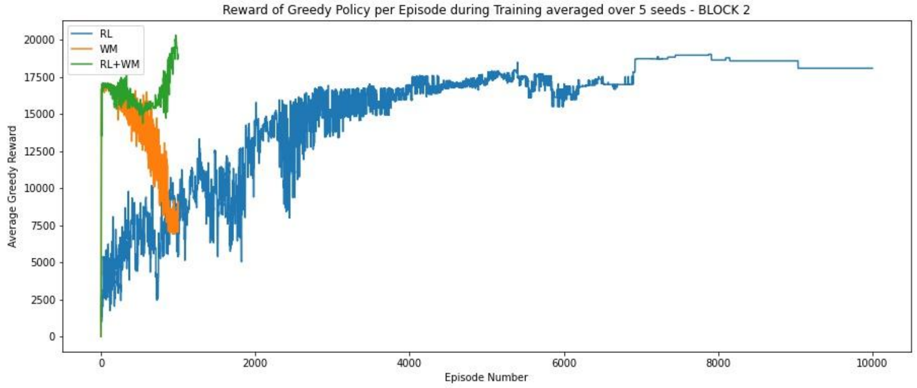
WM: Perfect

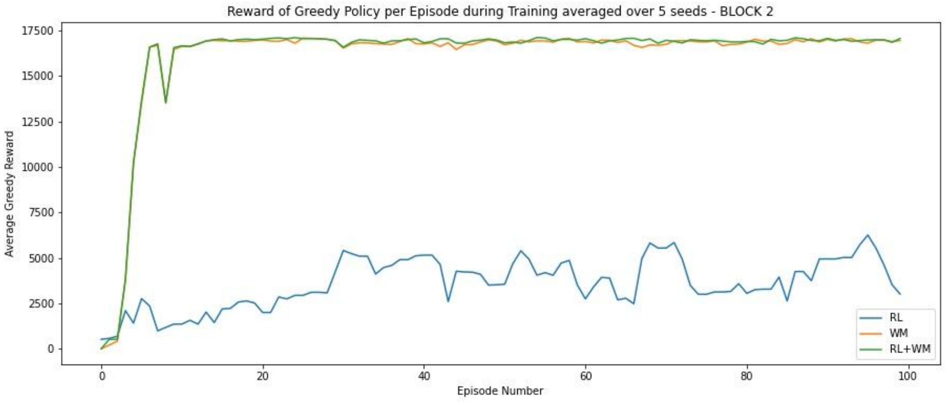
Experiment Results on Block **2**

- We assume infinite working memory
- For each bush we store:
    a. Last reward time
    b. Last received reward

  to predict replenishing rate for each bush
- 24 variables are stored in total.

Algorithm	Score
RL	18093
WM	8058
RL + WM	18929.8





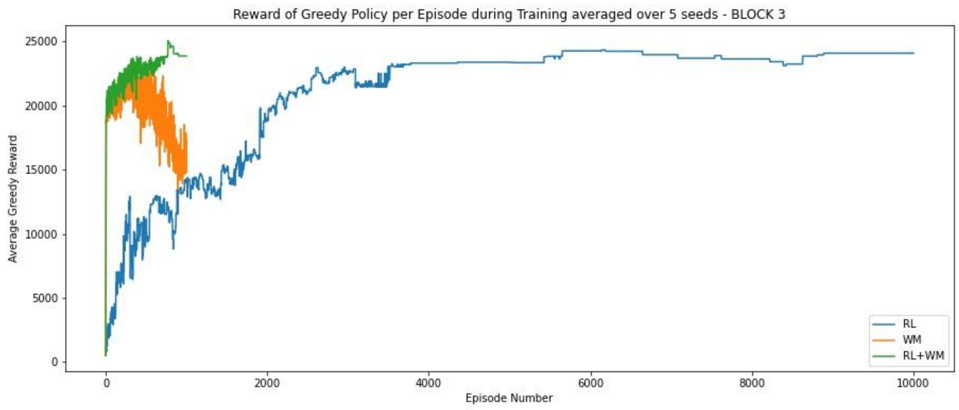
WM: Perfect

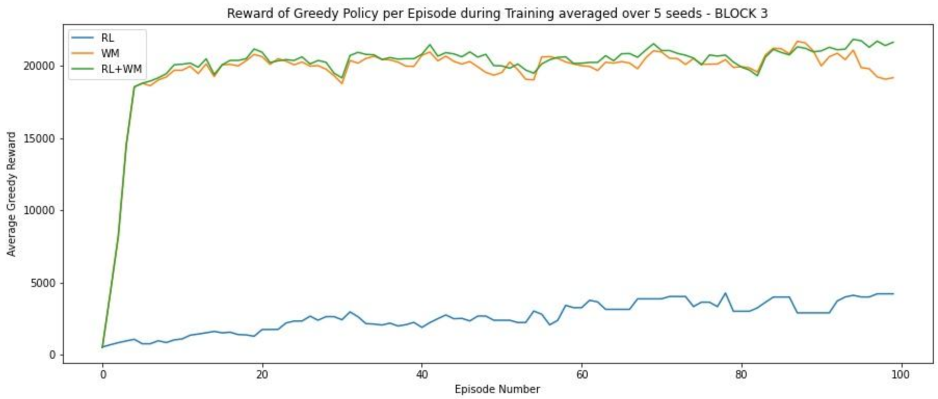
Experiment Results on Block **3**

- We assume infinite working memory
- For each bush we store:
  a. Last reward time
  b. Last received reward

  to predict replenishing rate for each bush
- 24 variables are stored in total.

Algorithm	**Score**
RL	24073.2
WM	17308.6
RL + WM	23857.0

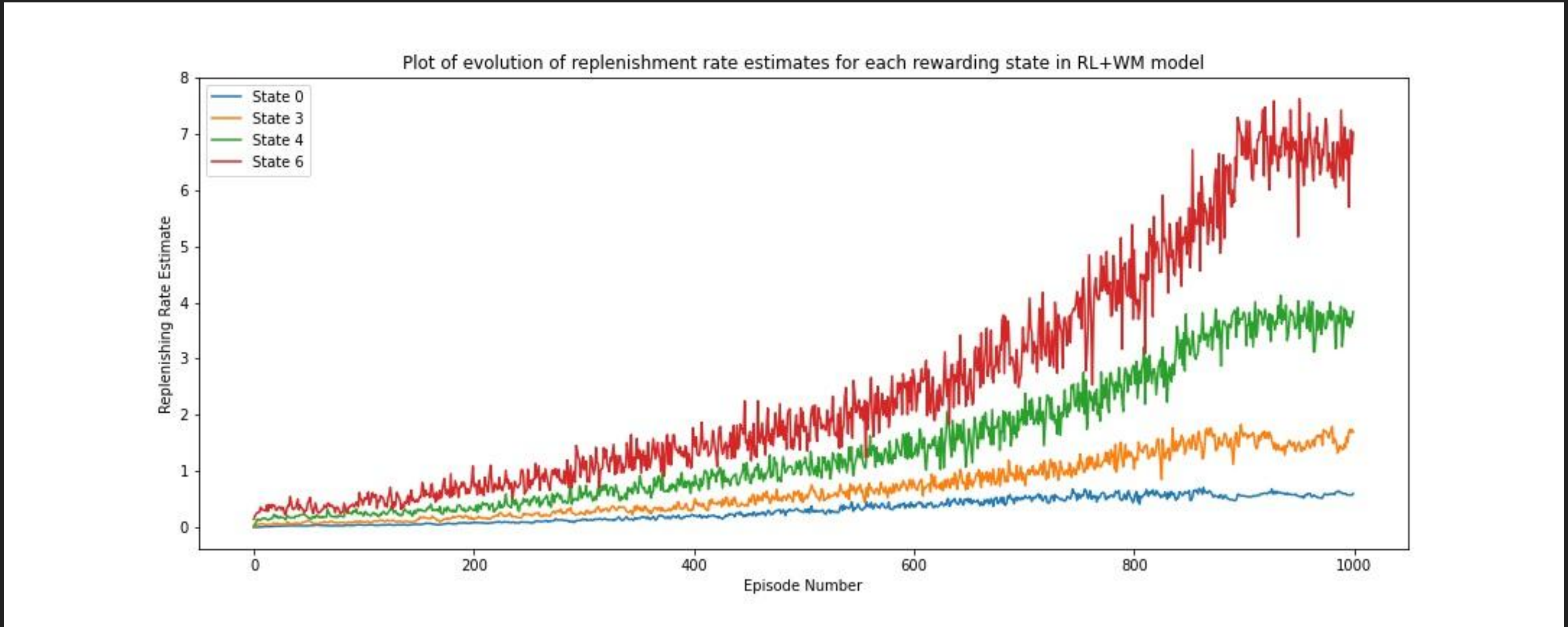




Variation of Replenishment rate with episodes

Experiment Results on Block **3**

- Block 3 has all different replenishing rate (2,4,6 and 8); estimates converge to true values

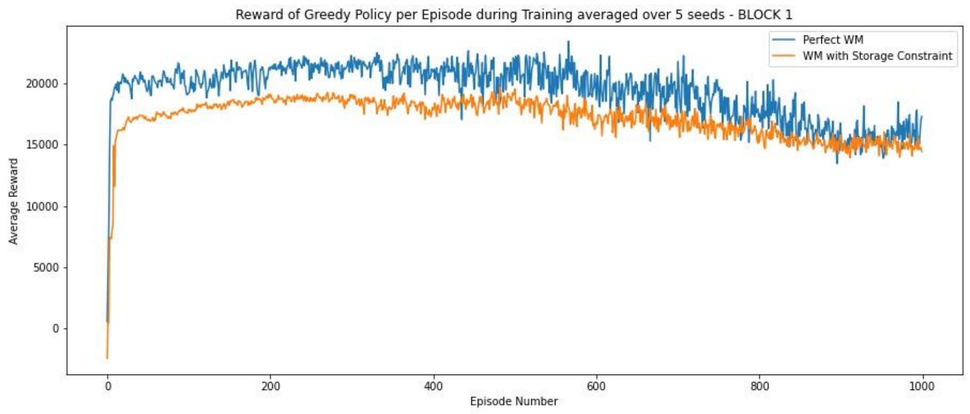


WM: storage constraint

Experiment Results on Block **1**

In this case we have limited the number of items(9) to store information

Algorithm	Score
Perfect WM	17308.6
WM with storage constraint	16453.6

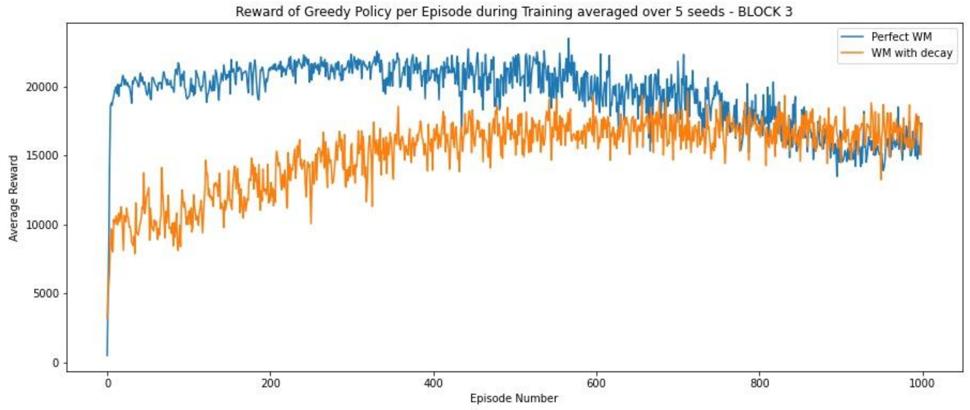


WM: Recollection Error

Experiment Results on Block **3**

- This experiment formulates human memory decay situation. Not everything is remembered precisely!
- Here, agent gets a value which is sampled from $N(\mu, \sigma^2)$ where μ is the estimated value stored in array and σ^2 is dependent on the time elapsed since last time this state was harvested.

Algorithm	Score
Perfect WM	17308.6
WM with decay	17142.4

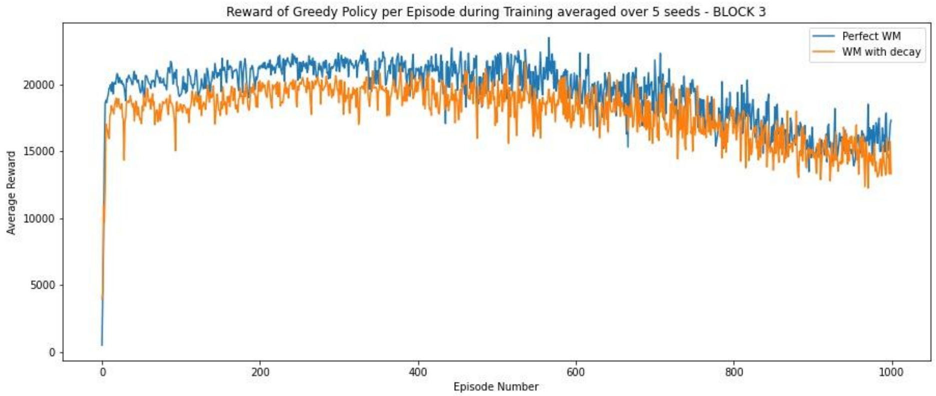


WM: Complete Model of Memory

Experiment Results on Block **3**

- It is a combination of the Limited Memory and the Decaying memory scenario
- We incorporate both limitations, i.e. have a limited working memory and forget it as time proceeds.

Algorithm	Score
Perfect WM	17308.6
WM with decay	13304.2

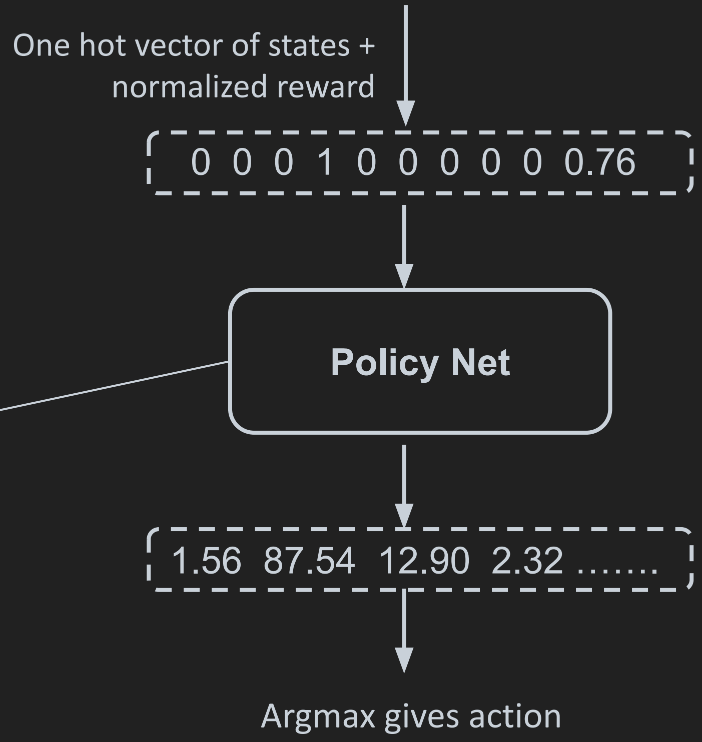


Deep RL Models

- Input: current state & last reward
- Output: corresponding action

- Used a 3 layer NN to learn the relationship between input and output

- Experimented by including time but results were not significant.



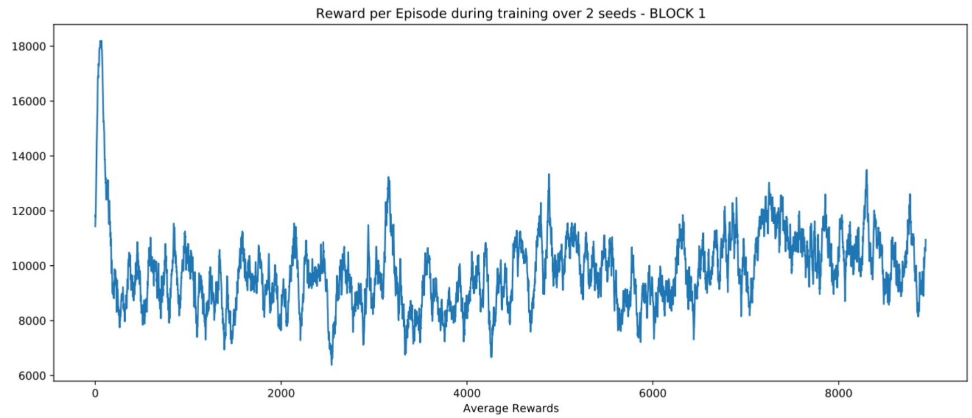

```
[180]  1 policy_net

DQN(
  (fc1): Linear(in_features=10, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=8, bias=True)
)
```

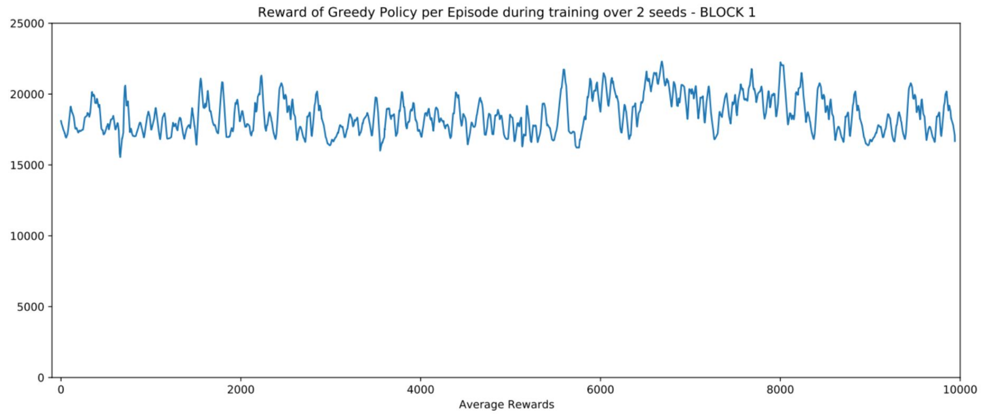
DeepQN

Experiment Results on Block **1**

- Reward: 22321
- Outperforms X, Y and Z

```
PARAMETERS:
- LR: 0.0001
- BATCH_SIZE = 128
- Loss: MSELoss
- Optimizer: Adam
- Output Update = 4
- Memory Size=10000
```



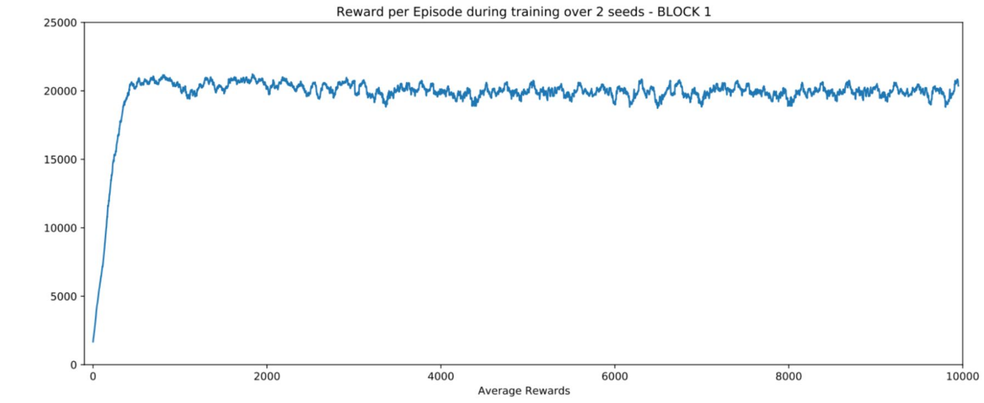


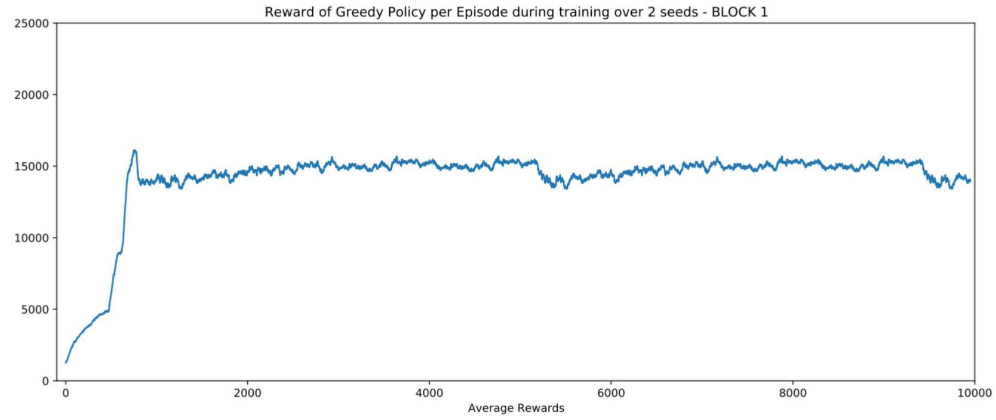
DeepQN

Experiment Results on Block **2**

- Reward: 17132
- Outperforms X, Y and Z

```
PARAMETERS:
- LR: 0.0001
- BATCH_SIZE = 128
- Loss: MSELoss
- Optimizer: Adam
- Output Update = 4
- Memory Size=10000
```



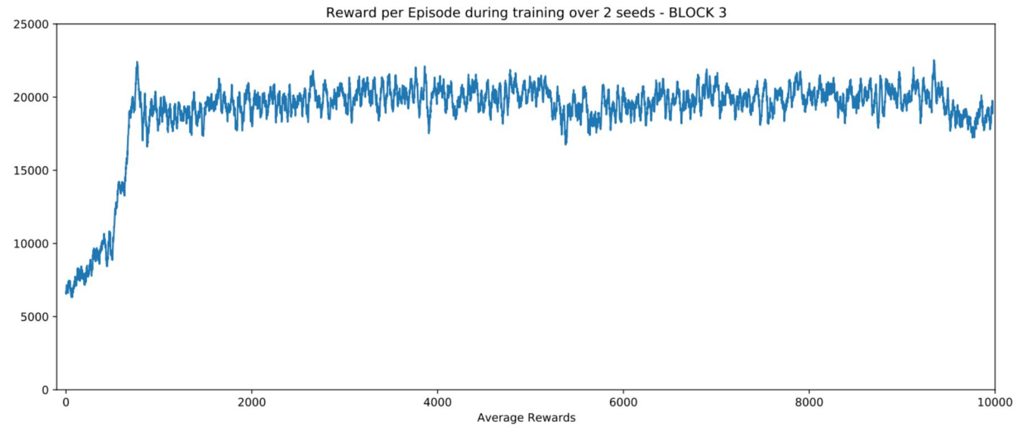


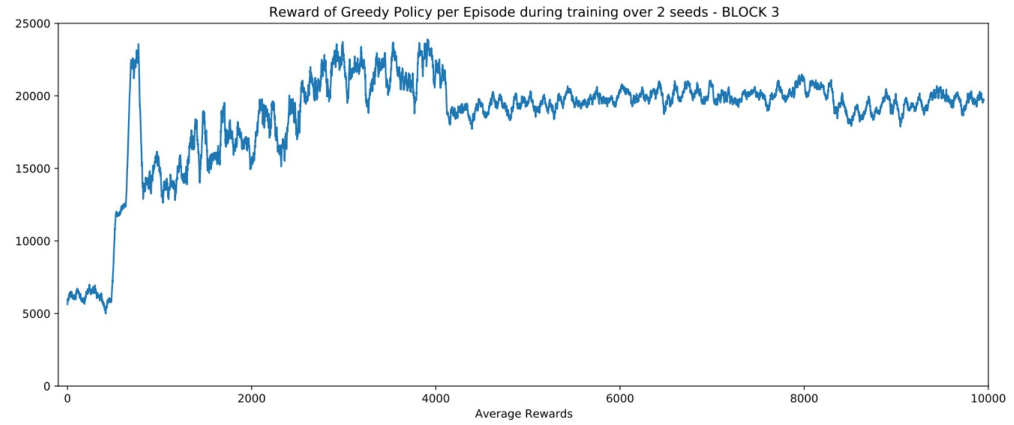
DeepQN

Experiment Results on Block 3

- Reward: 23867
- Outperforms X, Y and Z

```
PARAMETERS:
- LR: 0.0001
- BATCH_SIZE = 128
- Loss: MSELoss
- Optimizer: Adam
- Output Update = 4
- Memory Size=10000
```





Choice Recency Module

> Modifying the softmax_action_select function in the SARSA algorithm by including the effect of previous action choice.

> This is done using a choice vector. Whenever a given action take place, the corresponding choice value for that state and action is set to 1 while the choice values for that state and other actions is decayed by the factor of d, known as choice recency index.

> The action is chosen via the softmax function where probabilities are represented as shown below :

$$prob[a] = \frac{\exp\left((\mathrm{Q}[s][r]\,[a] + \mathrm{b}[s] \cdot \mathrm{choice}[s][r][a])/\tau \right)}{\sum_a \exp\left((\mathrm{Q}[s][r]\,[a] + \mathrm{b}[s] \cdot \mathrm{choice}[s][r][a])/\tau \right)}$$

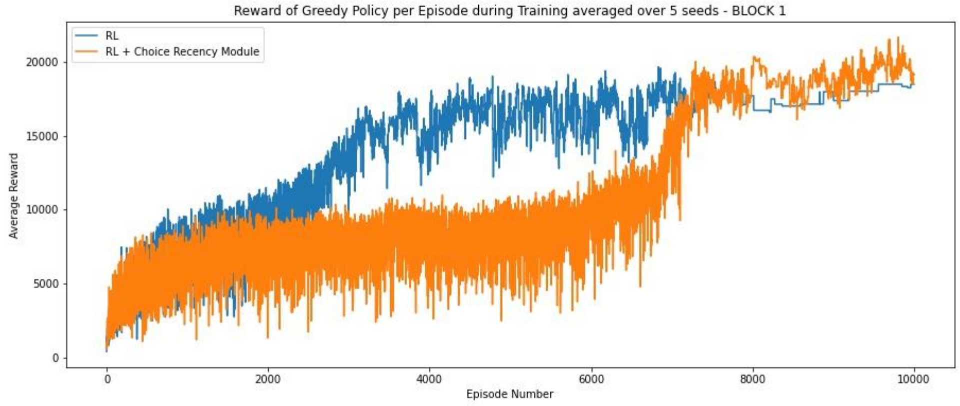
> The coefficient b is a function of the state. We exploited the Working Memory to learn the optimal values of b. This was done with the help of learning replenishing rates. This ensures that more the replenishing rate, more positive is the value of b, hence the greater probability.

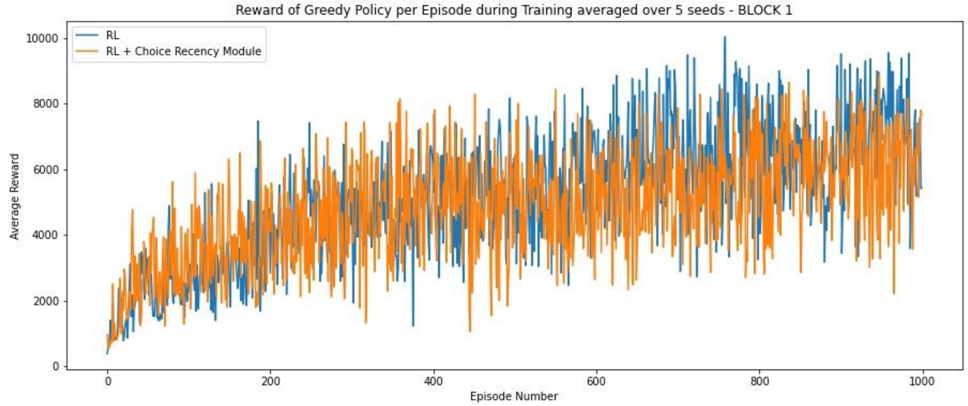
Choice Recency Module

Experiment Results on Block **1**

Algorithm	Score
RL	18473.5
RL + Choice Recency Model	19139.33

PARAMETER:
- choice recency index: 0.7
- tau: exp decay 10000 to 10



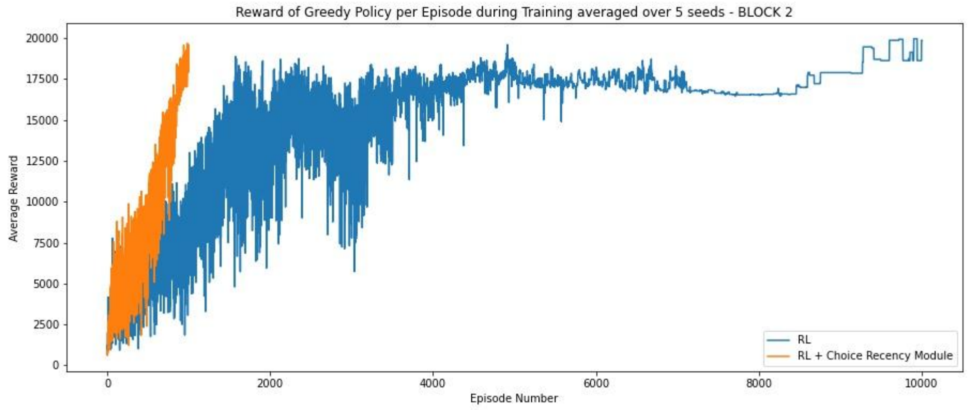


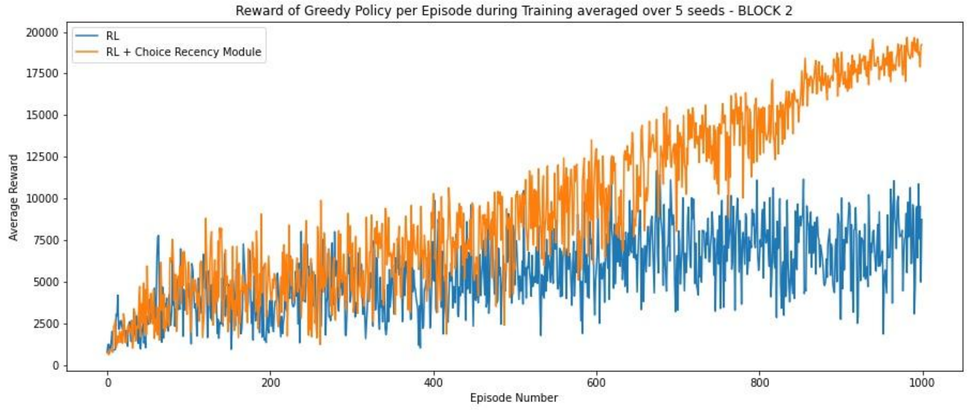
Choice Recency Module

Experiment Results on Block **2**

Algorithm	Score
RL	19859.5
RL + Choice Recency Model	20240.67

```
PARAMETER:
 -  choice recency index: 0.7
 -  tau: exp decay 10000 to 10
```



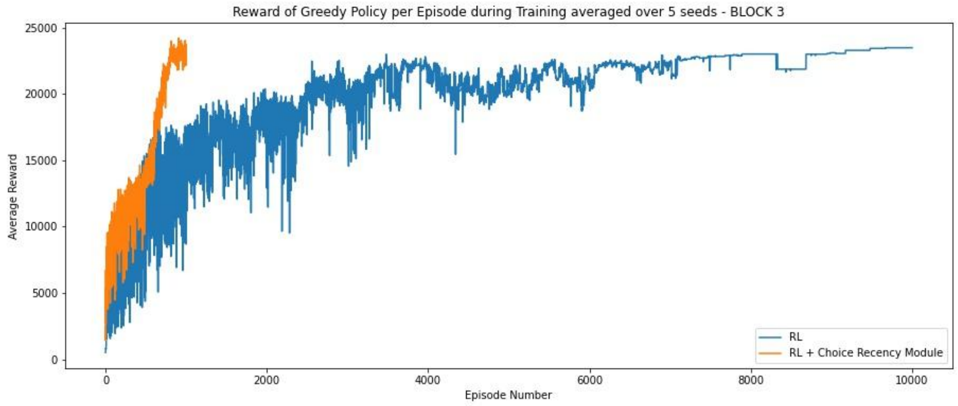


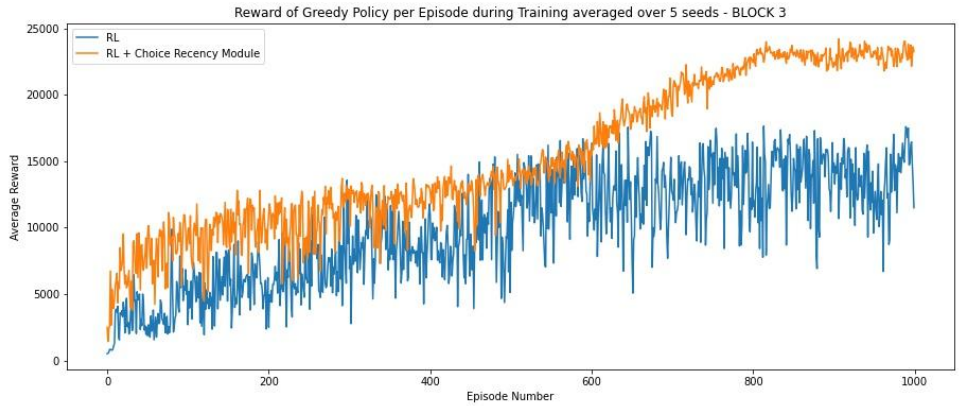
Choice Recency Module

Experiment Results on Block **3**

Algorithm	Score
RL	23484.8
RL + Choice Recency Model	23856.5

```
PARAMETER:
 -   choice recency index: 0.7
 -   tau: exp decay 10000 to 10
```





Algorithm	Action Selection Strategy	Block-1	Block-2	Block-3
SARSA	Epsilon-Greedy	18093.4	18473.5	23484
	Softmax	21797	19670.75	24209.5
	UCB	21366	19769	24363
Q-Learning	Epsilon-Greedy	19586.8	18093.4	24073
	Softmax	19186.6	21797	23607
	UCB	20827	21366	**24642**
Double Q-learning	Epsilon-Greedy	18983.8	19181.6	22949.6
RL + Working Memory	Epsilon-Greedy	20309.8	18929.8	23857.0
DQN	Epsilon-Greedy	**22321**	17132	23867
RL + Choice Recency Model	Epsilon-Greedy	19139.33	**20240.67**	23856.5

Overall Comparison of top performing algorithms

Conclusion & Future Work

1. We implemented various algorithms to find the best policy to play these foraging environments. Many of these algorithms give results which are very close to each other.
2. To try to understand and mimic human learning we included the WM module which quickly learns the rewarding and fast replenishing bushes.
3. This module helped us drastically reduce our training time but suffered drawback as real human have a very limited capacity of working memory.
4. However if we want to mimic real human behaviour even more closely we should also add a randomness error in the q value estimates as t.

Work Contribution of Members

All team members believe that each member has contributed significantly to the project

Work	Member
Environment Class, Implemented Working Memory	Anshul
Environment Testing, Tabular RL methods and experiments	Archit
Environment Rendering, Deep Q network	Ankur
Baseline Agent, Implemented Choice based RL model	Abhay
Literature Survey, Presentation and Report	All

Thank You