University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

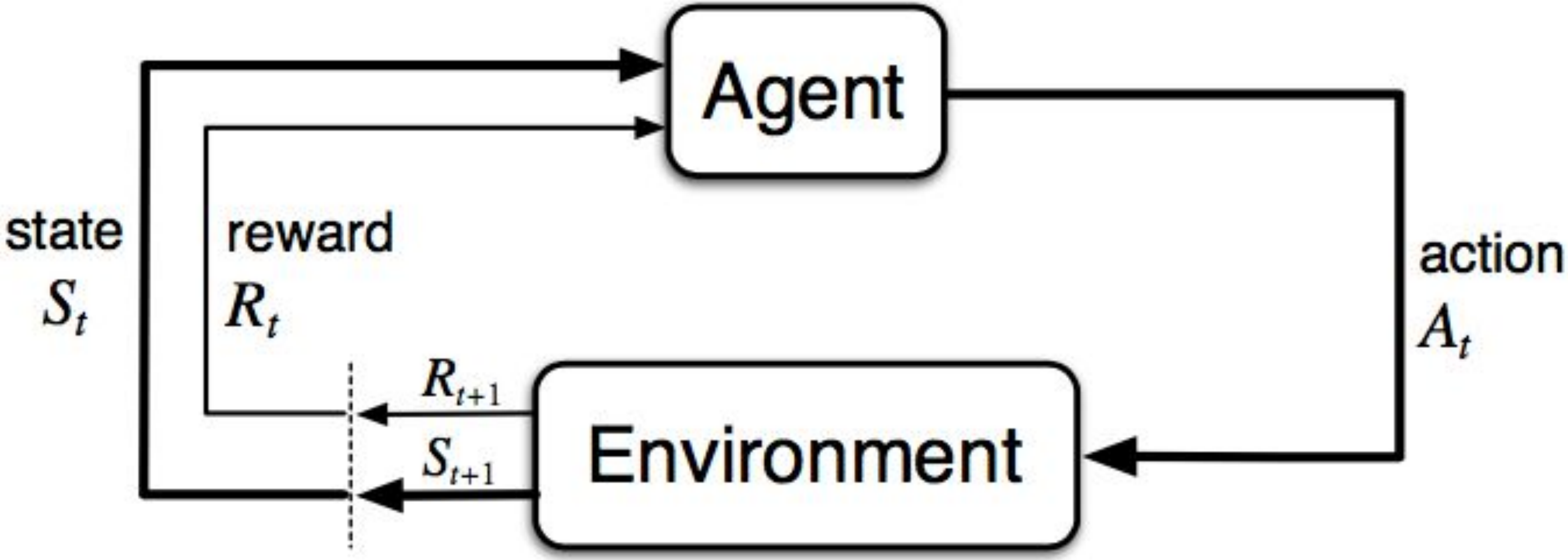
PARALLEL Q-LEARNING & ACTOR-CRITIC

Alina Vereshchaka

CSE 633 Parallel Algorithms (Dr. Russ Miller)

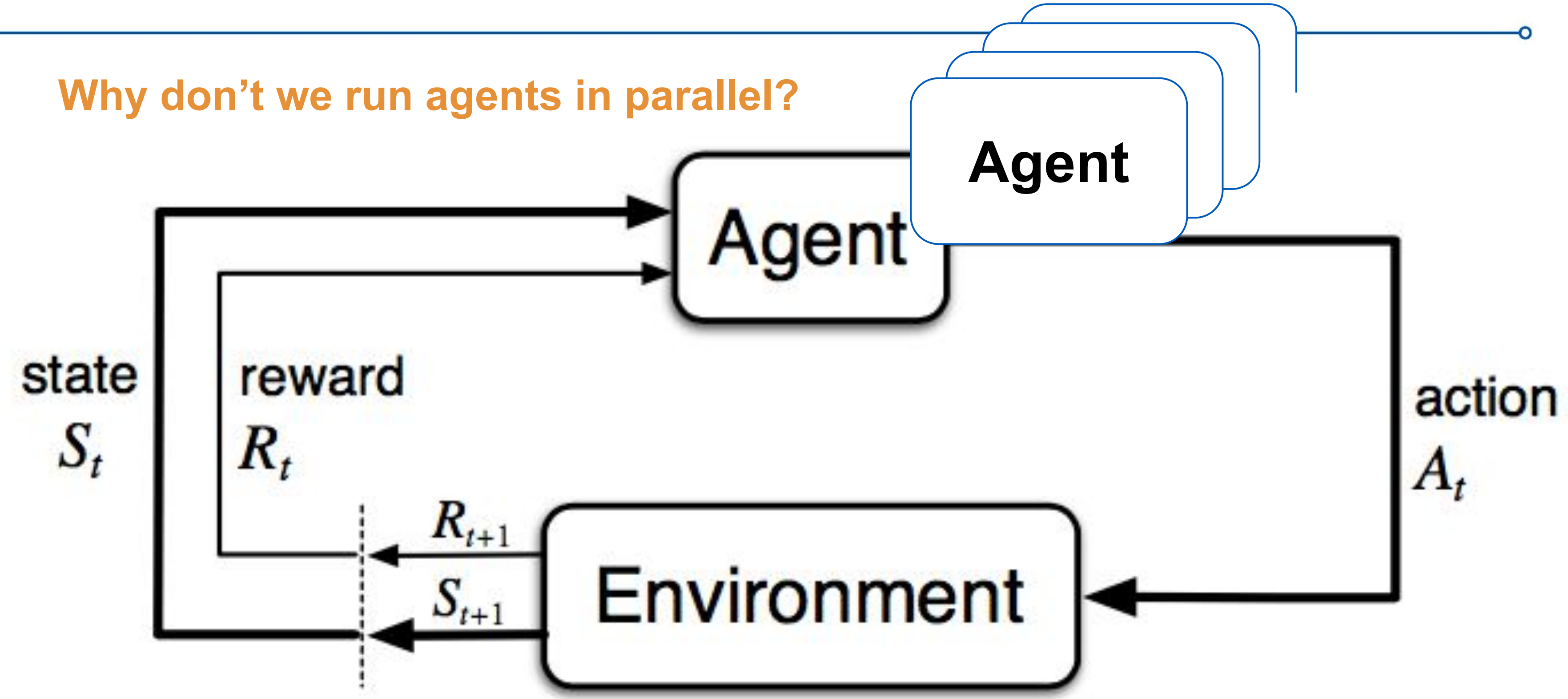
April 16, 2020

Markov Decision Process



Problem

Why don't we run agents in parallel?



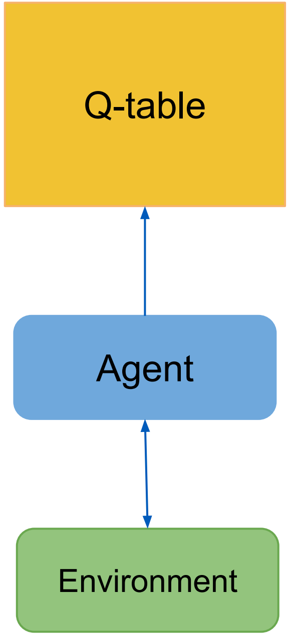
Q-Learning Algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

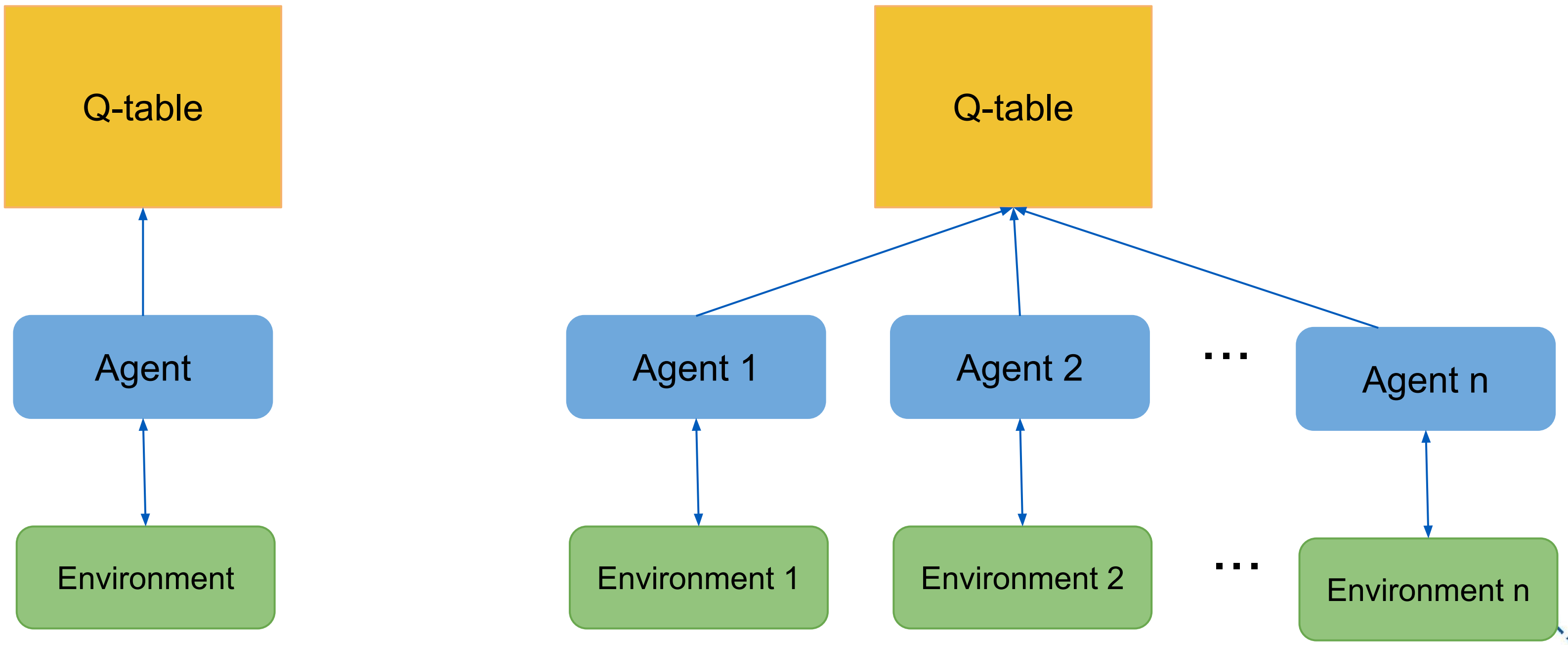
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Q-Learning Process

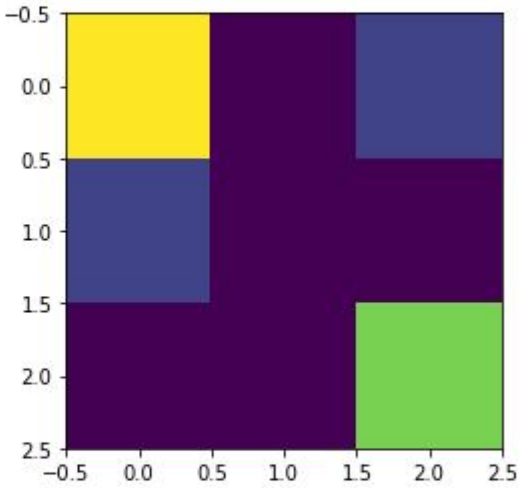


Parallel Q-Learning Process

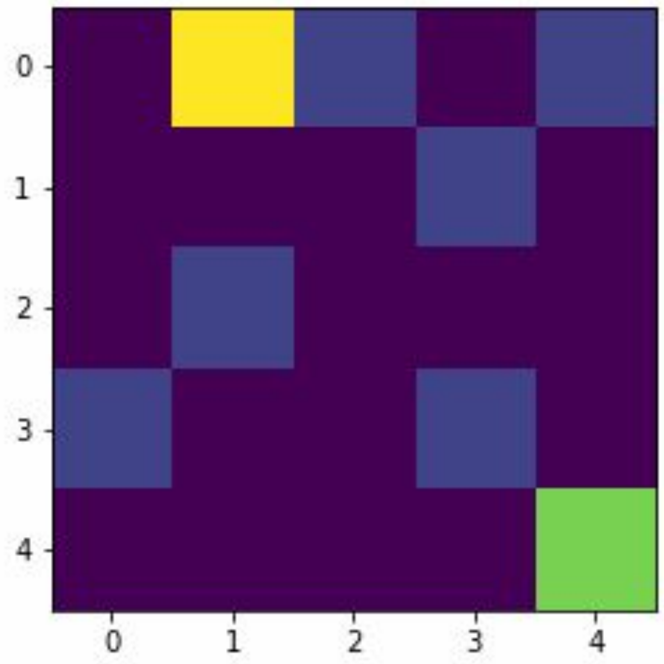


Problem - Grid world

- ***N* x *N*** square grid
- Agent (yellow) starts at random positions
- **Goal (green):** reach position **(N-1, N-1)**
- **Actions:** {Down (0), Up (1), Right (2), Left (3)}
- **Rewards:** {-1, -0.1, 0, 0.1, 1}



Random Agent



CCR Resources

Intel Xeon Gold 6130 (2/node)

Running

FISBATCH
2521470

Cluster	Academic Cluster
Job Id	2521470
Job Name	FISBATCH
User	avereshc
Account	wendong
Partition	skylake
State	RUNNING
Reason	None
Total Nodes	16
Node List	cpn-u23-35, cpn-u24-20, cpn-u24-21, cpn-u24-23, cpn-u24-24, cpn-u24-25, cpn-u24-27, cpn-u24-28, cpn-u24-29, cpn-u24-30, cpn-u24-31, cpn-u24-32, cpn-u24-33, cpn-u24-34, cpn-u24-35, cpn-u24-36
Total CPUs	512
Time Limit	1-00:00:00
Time Used	12:05
Memory	48000M

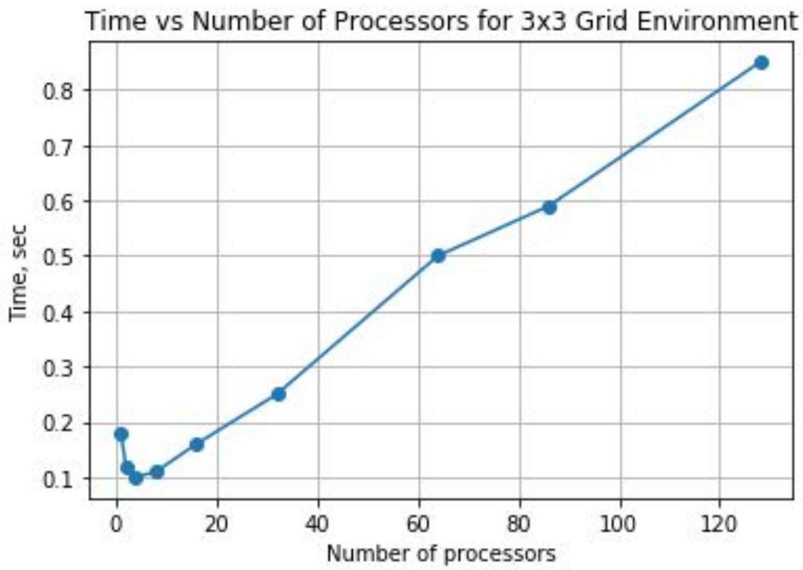
Output Location: `/user/avereshc`

Open in File Manager

>_ Open in Terminal

Delete

Performance of 3 x 3 Grid (episodes = 1000)

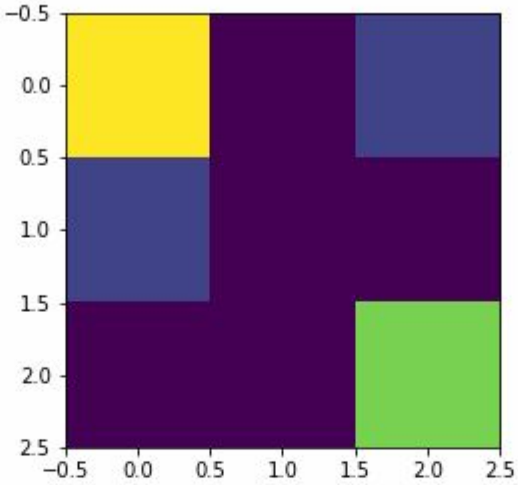


Processors	Time (in secs)
1	0.18
2	0.12
4	0.10
8	0.11
16	0.16
32	0.25
64	0.50
86	0.59
128	0.85

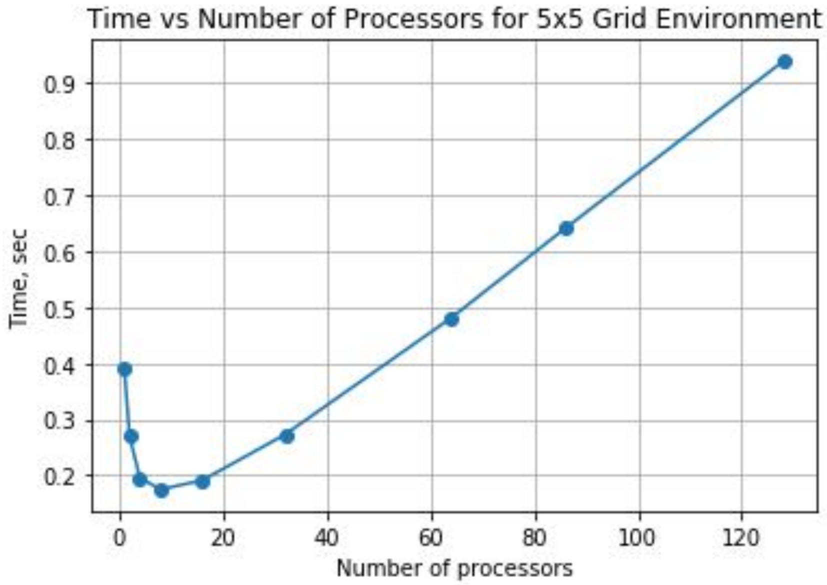
Results of 3 x 3 Grid (episodes = 1000)

```
print(agent.q_table)
```

```
[[[ 0.03803131  0.53740696  1.0729      0.45049737]
  [ 1.081       0.80910245 -0.12187289  0.85137205]
  [ 0.8217115  -0.29481583  0.          0.87081859]]

 [[ 0.27783203  0.73679678  1.081      -0.12274178]
  [ 1.09        0.4090045   1.07536036 -0.13012733]
  [ 1.09370094 -0.06834097  0.          0.80239136]]

 [[-0.01       -0.11530044  0.78215181  0.        ]
  [ 0.80135152  0.16739     1.1         0.35720553]
  [ 0.          0.          0.          0.        ]]]
```



Performance of 5 x 5 Grid (episodes = 2000)

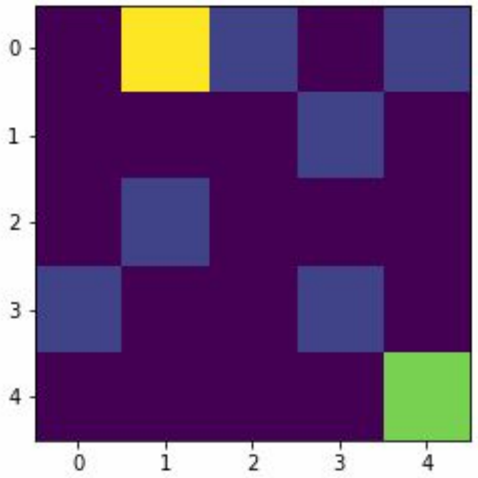


Processors	Time (in secs)
1	0.39
2	0.27
4	0.195
8	0.175
16	0.191
32	0.273
64	0.481
86	0.641
128	0.939

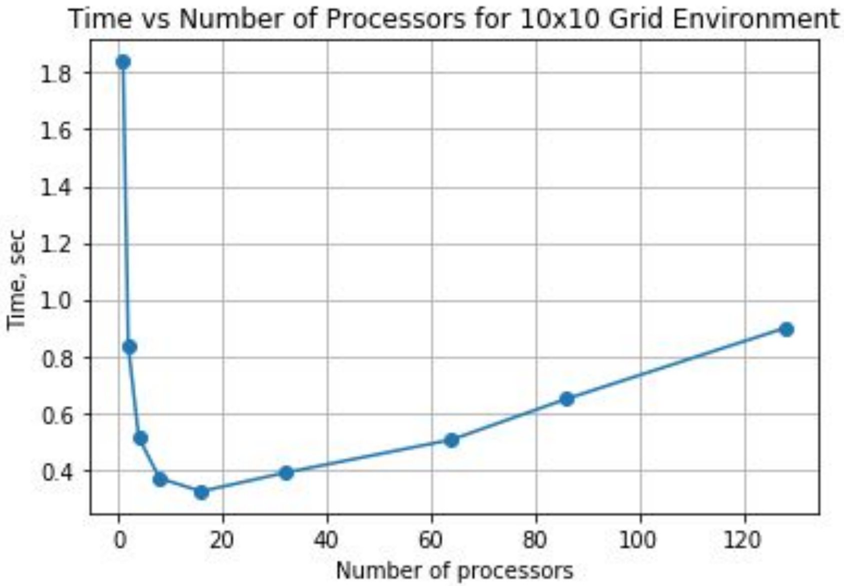
Results of 5 x 5 Grid (episodes = 2000)

```
print(agent.q_table)
```

```
[[[ 1.047829    0.74745711  1.04172104  0.66081483]
  [ 1.05314392  0.84484722  0.05305683  0.71752684]
  [ 1.059049   -0.14685591  0.1578792   0.591269  ]
  [ 0.06550979 -0.04124037 -0.46002143 -0.1469156 ]
  [ 0.60888141  0.         -0.70324048 -0.06703397]]

 [[ 0.86002122  0.80571519  1.0531441   0.58883431]
  [ 0.0589567   0.84716299  1.059049    0.69579001]
  [ 1.06561    -0.14689648  0.05871875  0.82878334]
  [ 1.0729     -0.06888295  0.7563102   0.85061755]
  [ 0.99282828  0.          0.2157909  -0.15449581]]

 [[-0.00286144  0.84782955  0.04199235  0.29319149]
  [ 1.06113535  0.85299416  1.06560983  0.51270305]
  [ 1.0729      0.85902868  1.0463074  -0.1396225 ]
  [ 0.08050105 -0.13747675  1.081       0.64558224]
  [ 1.09        0.28890989  0.58592215  0.49507816]]

 [[ 1.0655101   0.60849239  0.43314204  0.        ]
  [ 1.07289991 -0.14170876  0.61100408 -0.21275152]
  [ 1.081       0.86516686  0.07177179  0.61531979]
  [ 1.09        0.85587013  0.29483261  0.4089936 ]
  [ 1.1         0.16402212  0.36446386 -0.07294629]]

 [[ 0.44017342 -0.25809691  1.07288728  0.08655158]
  [ 0.7818582   0.83029131  1.081       0.29010182]
  [ 0.88094363  0.87200764  1.09        0.16517202]
  [ 0.88999985 -0.11901329  1.1         0.53968254]
  [ 0.          0.          0.          0.        ]]]
```

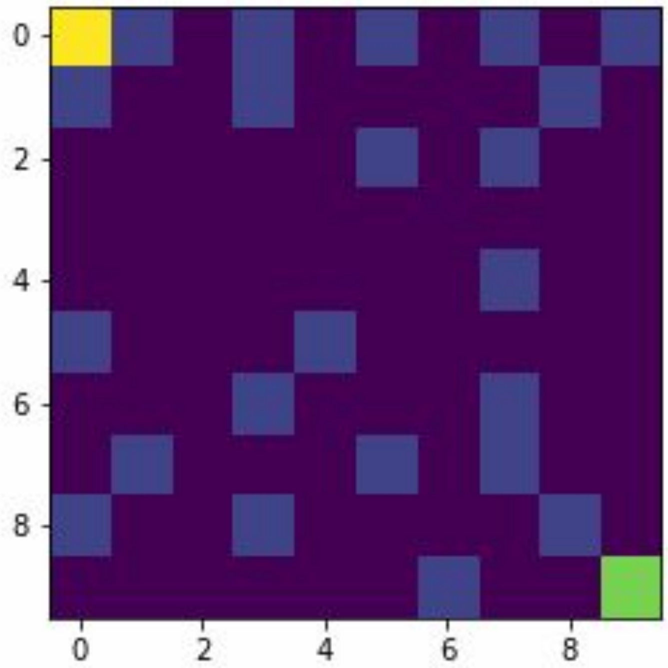


Performance of 10 x 10 Grid (episodes = 5000)

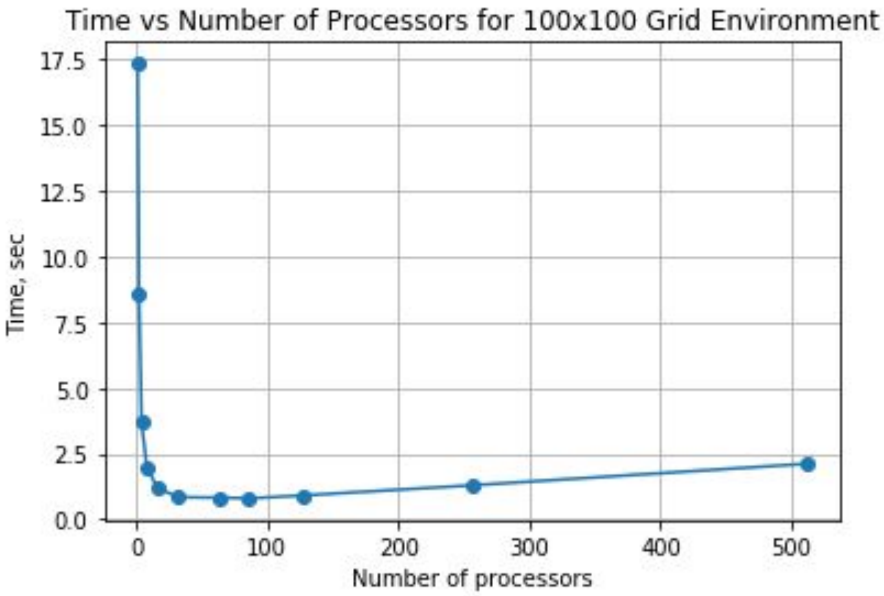


Processors	Time (in secs)
1	1.84
2	0.83
4	0.51
8	0.36
16	0.32
32	0.38
64	0.50
86	0.64
128	0.89

Results of 10 x 10 Grid (episodes = 5000)

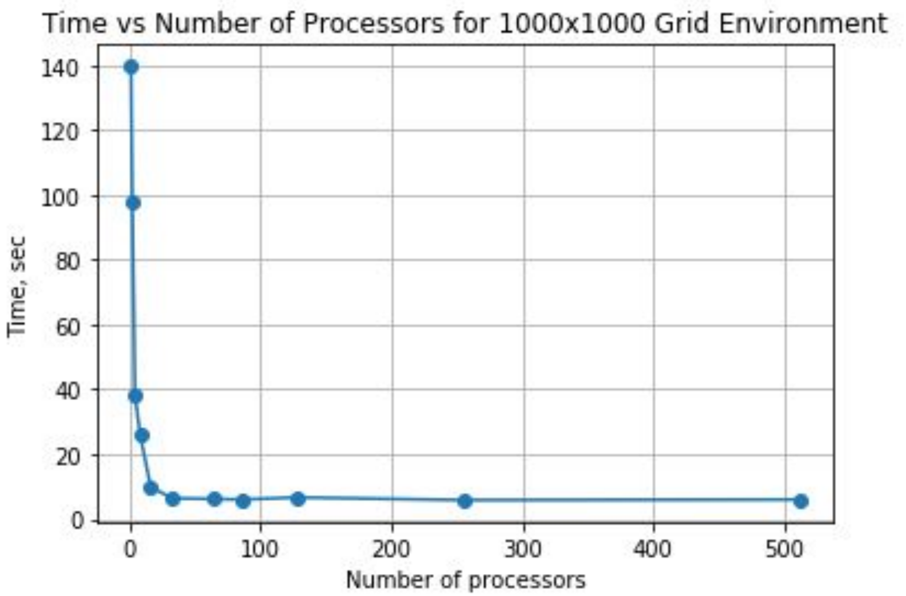


Performance of 100 x 100 Grid (episodes = 50 000)



Processors	Time (in secs)
1	17.33
2	8.59
4	3.68
8	1.98
16	1.20
32	0.84
64	0.82
86	0.79
128	0.91
256	1.29
512	2.11

Performance of 1000 x 1000 Grid (episodes = 500 000)



Processors	Time (in secs)
1	139.70
2	97.94
4	38.61
8	25.92
16	9.99
32	6.46
64	6.18
86	6.02
128	6.65
256	5.84
512	6.00

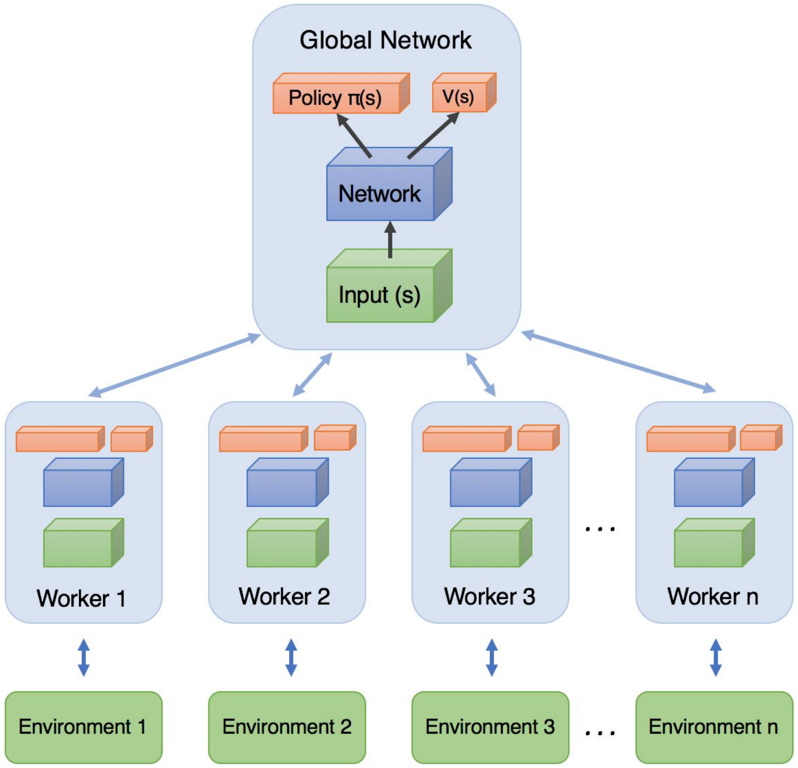
Asynchronous Advantage Actor Critic (A3C)

Asynchronous Advantage Actor-Critic:

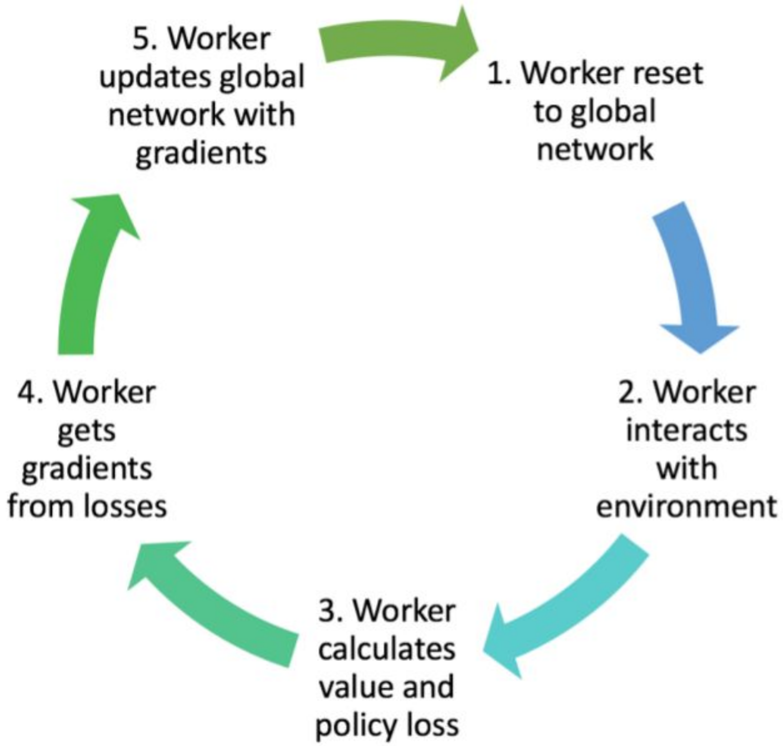
- Sample for data can be parallelized using several copies of the same agent use N copies of the agents (workers) working in parallel collecting samples and computing gradients for policy and value function
- After some time, pass gradients to a main network that updates actor and critic using the gradients of all agents
- After some time the worker copy the weights of the global network

This parallelism decorrelates the agents' data, so no experience replay buffer needed

Asynchronous Advantage Actor Critic (A3C)



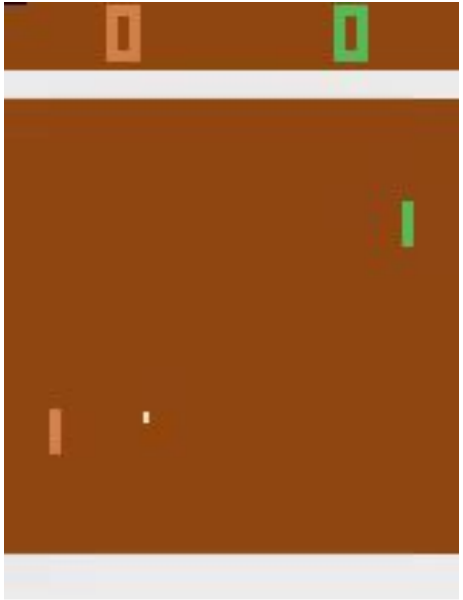
Asynchronous Advantage Actor Critic (A3C)



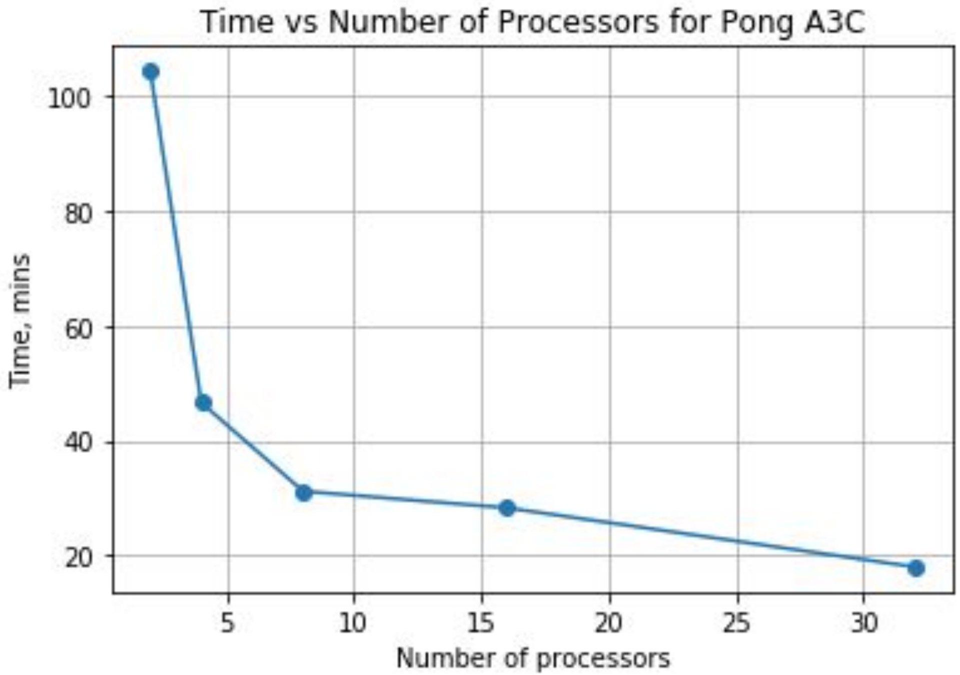
Pong Deterministic by OpenAI Gym

Observation: an RGB image of the screen (210, 160, 3)

Each action is repeatedly performed for a duration of k frames



Performance of A3C for Pong (OpenAI Gym)



Processors	Time (in mins)
2	104.27
4	46.59
8	31.18
16	28.29
32	18.03

CCR Server Used: Intel Xeon Gold 6130 (2/node), NVidia Tesla V100 (2/node)

Conclusion

- Parallelization was done in multiprocessing (Python) to evenly distribute episodes to each process with a random agent starting position for each episode. This way each process will have its own Q-table and will eventually be reduced by summing the main Q-table.
- There is an improvement in runtime as the number of processes increases. However, it is not very scalable because it is unable to maintain efficiency with increasing problem size and number of processes.
- Parallel Q-learning showed a faster convergence comparing to a sequential version

References

1. Richard S. Sutton and Andrew G. Barto, "Reinforcement learning: An introduction", Second Edition, MIT Press, 2019

2. Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. 2016.

3. CCR Knowledge Base https://ubccr.freshdesk.com/support/solutions

Thank you!