Raven: Scheduling Virtual Machine Migration during Data Center Upgrades with Reinforcement Learning

Chen Ying
Baochun Li
Xiaodi Ke
Lei Guo

Summarized by Jennifer Ly

TODAY'S AGENDA

1. Background & Context
2. Brief Deep RL Summary
3. Raven: The Scheduler
4. Results
5. Conclusions
6. How does this relate?

BACKGROUND & CONTEXT

- Common for modern data centers to require maintenance upgrades for physical machines(PMs)
  - migrate virtual machines (VMs)
  - reduce downtime and/or disruptions
  - migrating images takes the longest
- Must carefully select destination PM and schedule the VM migration
- Not much related works or within 'normal' situations
  - network topology and link capacities would be initially unknown

Problem to Solve

Can we leverage deep reinforcement learning to schedule VM migration to ensure the lowest total migration time to upgrade the PMs? And can this be done without prior knowledge of the network topology and network link capacities

		Upgrade PM #0	Upgrade PM #1	Upgrade PM #2
PM #0 4 CPUs, 8GB memory	VM #0 (1 CPU, 1GB memory)			VM #0 (1 CPU, 1GB memory) VM #1 (1 CPU, 2GB memory) VM #2 (2 CPUs, 3GB memory)
PM #1 4 CPUs, 12GB memory	VM #1 (1 CPU, 2GB memory)	VM #1 (1 CPU, 2GB memory)		
PM #2 4 CPUs, 8GB memory	VM #2 (2 CPUs, 3GB memory)	VM #0 (1 CPU, 1GB memory) VM #2 (2 CPUs, 3GB memory)	VM #0 (1 CPU, 1GB memory) VM #1 (1 CPU, 2GB memory) VM #2 (2 CPUs, 3GB memory)	

Fig. 2: Scheduling VM migration: a schedule that takes 9 seconds of total migration time.

		Upgrade PM #0	Upgrade PM #2	Upgrade PM #1
PM #0 4 CPUs, 8GB memory	VM #0 (1 CPU, 1GB memory)		VM #2 (2 CPUs, 3GB memory)	VM #1 (1 CPU, 2GB memory) VM #2 (2 CPUs, 3GB memory)
PM #1 4 CPUs, 12GB memory	VM #1 (1 CPU, 2GB memory)	VM #0 (1 CPU, 1GB memory) VM #1 (1 CPU, 2GB memory)	VM #0 (1 CPU, 1GB memory) VM #1 (1 CPU, 2GB memory)	
PM #2 4 CPUs, 8GB memory	VM #2 (2 CPUs, 3GB memory)	VM #2 (2 CPUs, 3GB memory)		VM #0 (1 CPU, 1GB memory)

Fig. 3: Scheduling VM migration: a schedule that takes 7 seconds of total migration time.

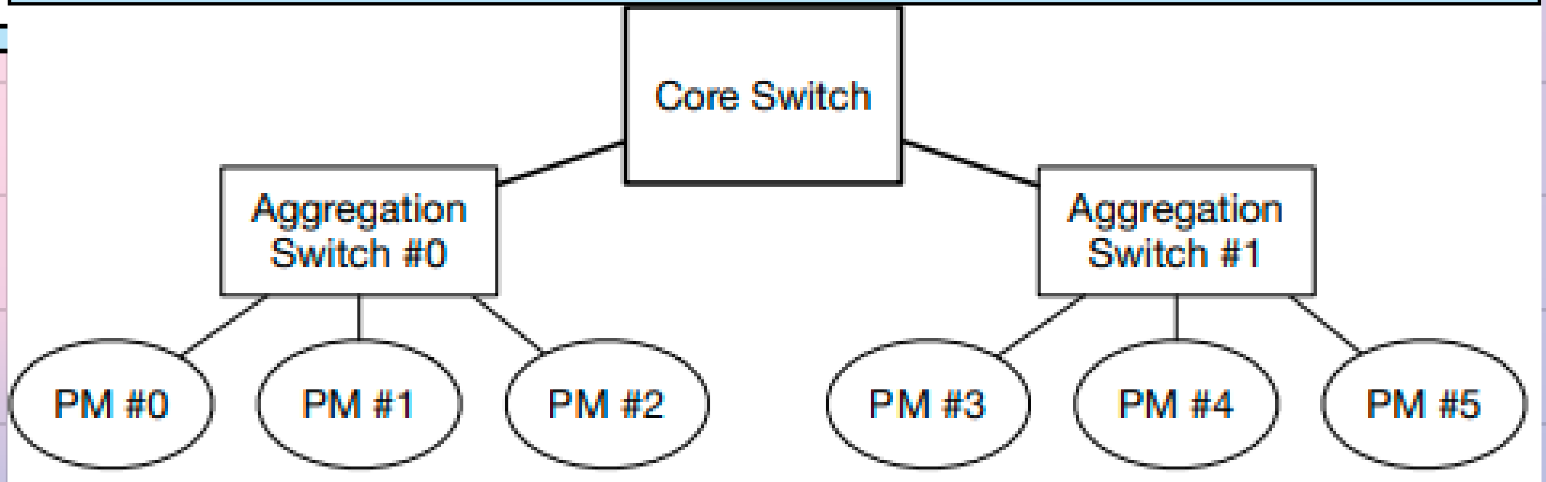
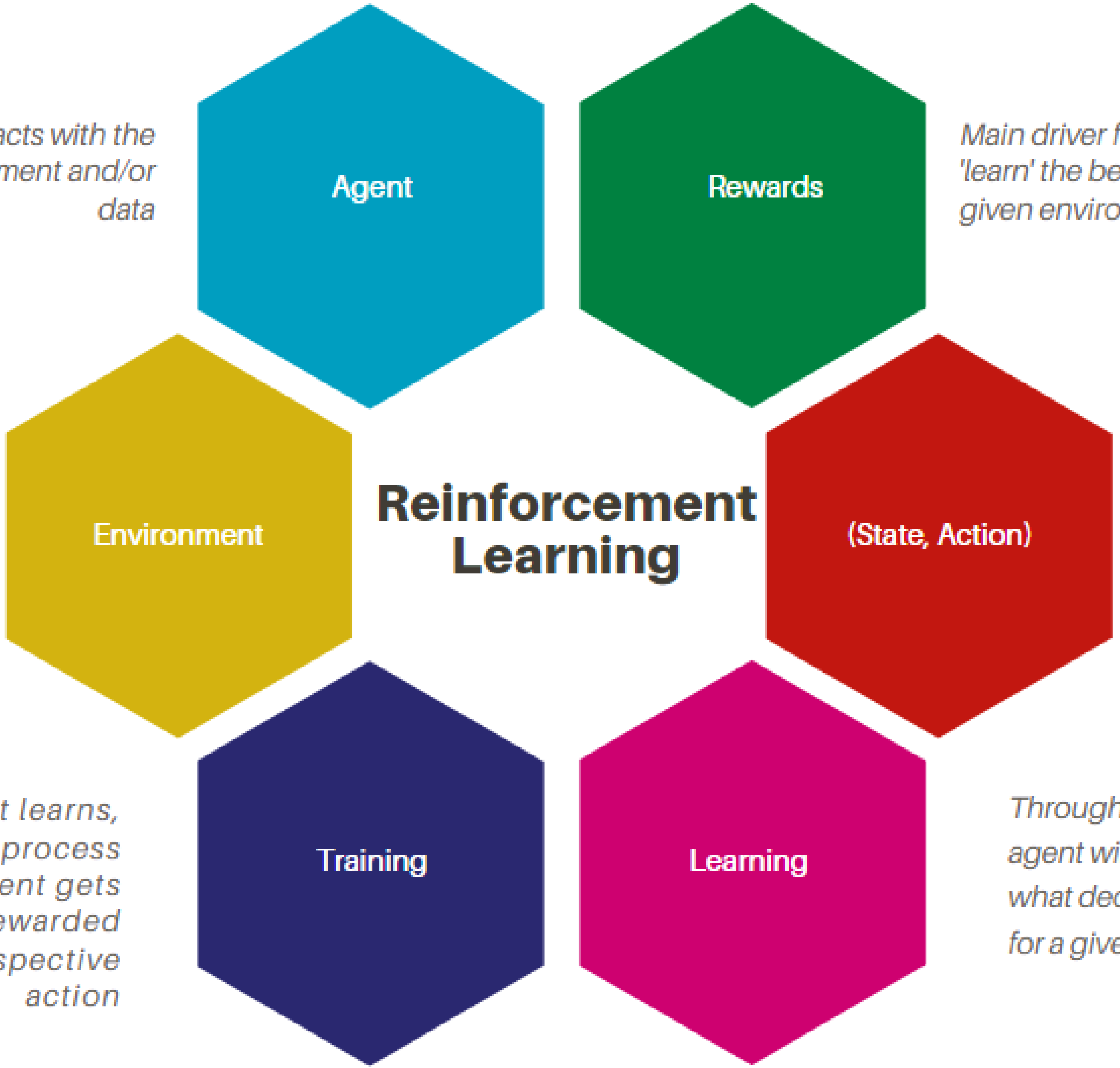


Fig. 1: The network topology of the example.

BRIEF REINFORCEMENT LEARNING SUMMARY



Directly interacts with the given environment and/or data

Main driver for the agent to 'learn' the best policy in its given environment

environment or data for agent to interact with to evaluate its current state and take a next action

Given a state, the agent is allowed to make any legal action available based from its current state

how the agent learns, part in the process where the agent gets punished or rewarded for its respective action

Through training, the agent will begin to learn what decision to best take for a given state

BRIEF DEEP REINFORCEMENT LEARNING SUMMARY

Extra features

- Fully connected neural network
  - adjustable policy $\pi(a|s;\theta)$ and parameters
- Cross-Entropy method is used in calculations to find optimal policy $\pi(a|s;\theta^*)$

Parameter for -> sampling (3)

$$\hat{v} = \arg\max_{v} \frac{1}{N} \sum_{n \in [N]} \mathbf{1}_{\{R(x_n) \geq \xi\}} \frac{f(x_n;u)}{f(x_n;w)} \log f(x_n;v), \tag{3}$$

$$\hat{\theta}_k = \arg\max_{\theta_k} \sum_{n \in [N]} \mathbf{1}_{\{R(x_n) \geq \xi_k\}} \left(\sum_{a_t, s_t \in x_n} \pi(a_t|s_t;\theta_k) \right), \tag{4}$$

<- Parameter estimator at iteration 'k' (4)

It's reinforcement learning with a few extra features..

RAVEN?

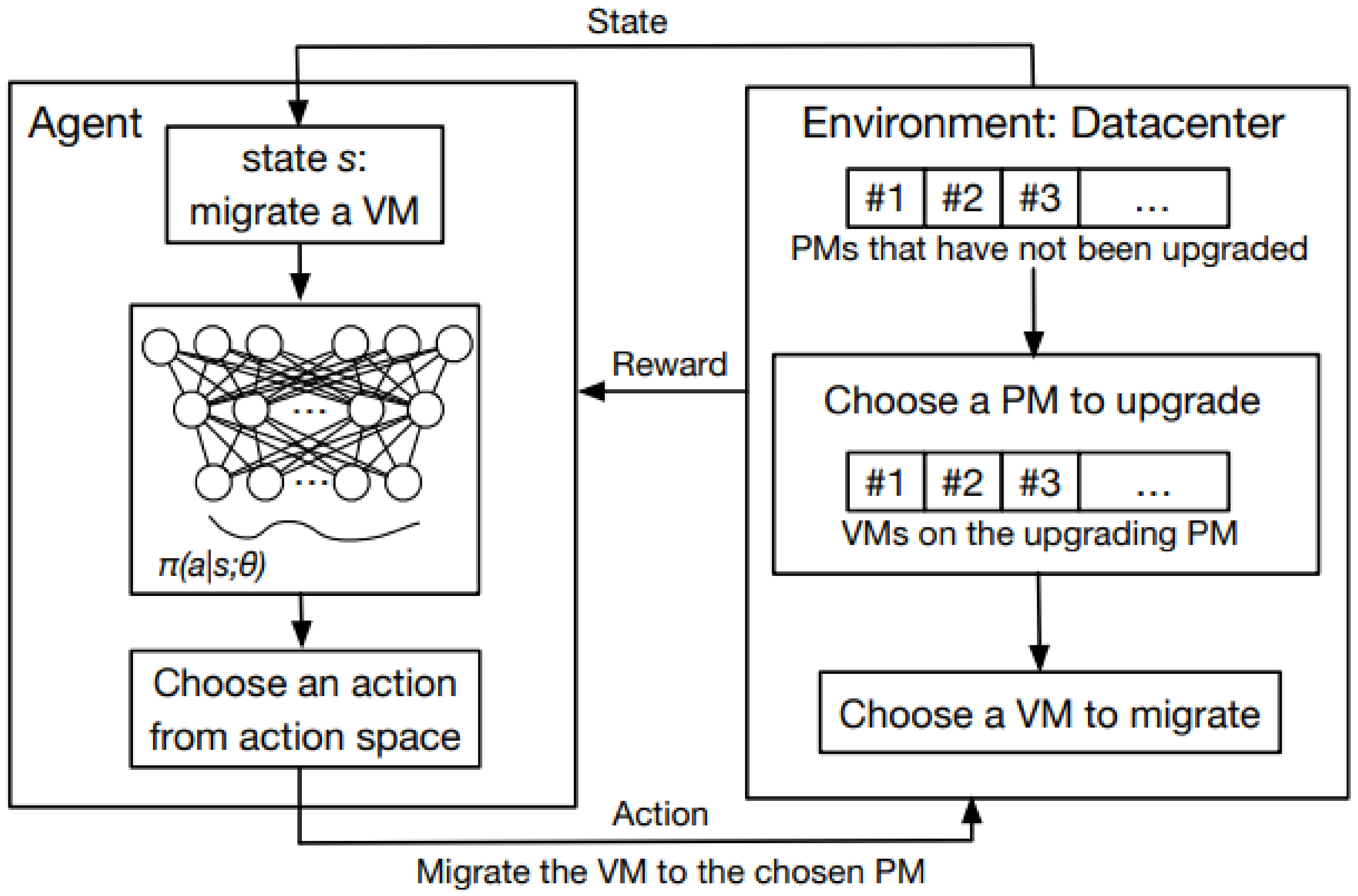


Fig. 4: The architecture of *Raven*.

State of the environment:

$$s_t = \{s_{t1}, s_{t2}, \ldots, s_{tJ}, v_t^{\text{cpu}}, v_t^{\text{mem}}, v_t^{\text{pm id}}\},$$

Episode: Finish upgrading all PMs
Time step

Start: Pick PM that needs to be upgraded

Per timestep: VM is migrated

Action: destination PM index

State Space: $s_{tj} = \{s_{tj}^{\text{status}}, s_{tj}^{\text{total cpu}}, s_{tj}^{\text{total mem}}, s_{tj}^{\text{used cpu}}, s_{tj}^{\text{used mem}}\},$

Reward: lower total migration time

RESULTS

Experiment was done on varying network topologies that were either 2-layer and 3-layer settings with 16 episodes for each setting

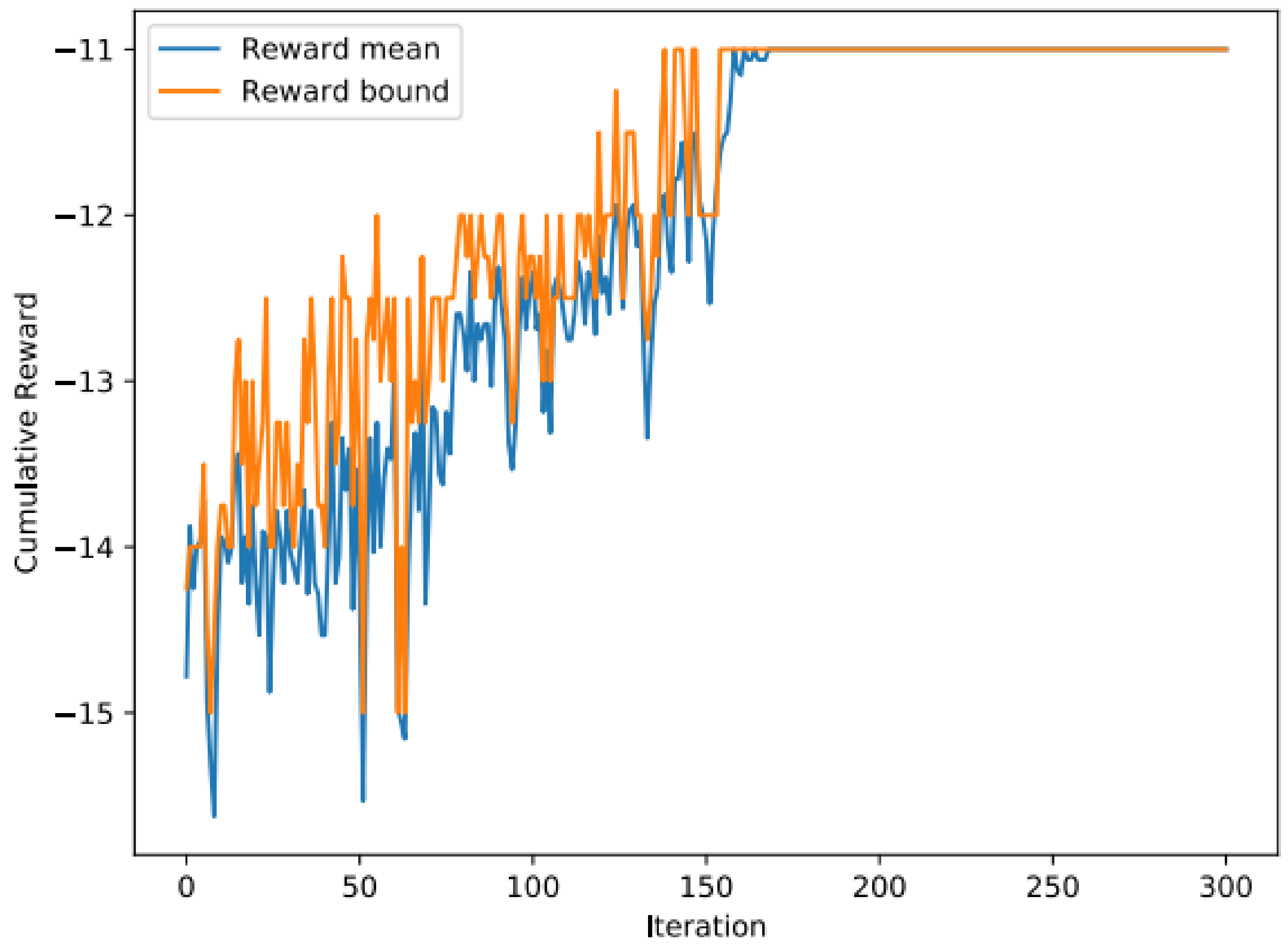


Fig. 5: The learning curve of *Raven* in datacenter with 9 physical machines and 30 virtual machines.

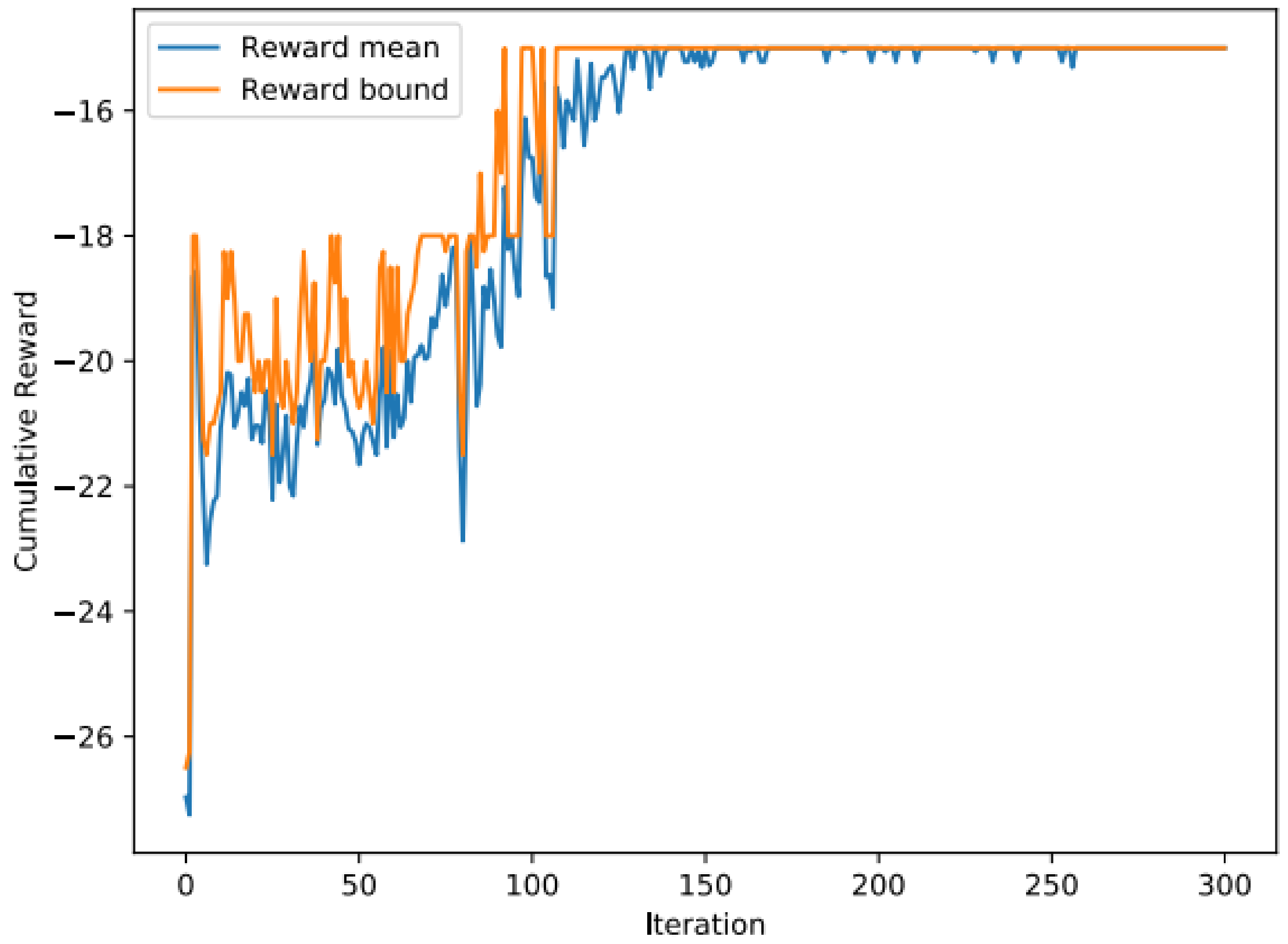


Fig. 6: The learning curve of *Raven* in datacenter with 10 physical machines and 40 virtual machines.

RESULTS CTD

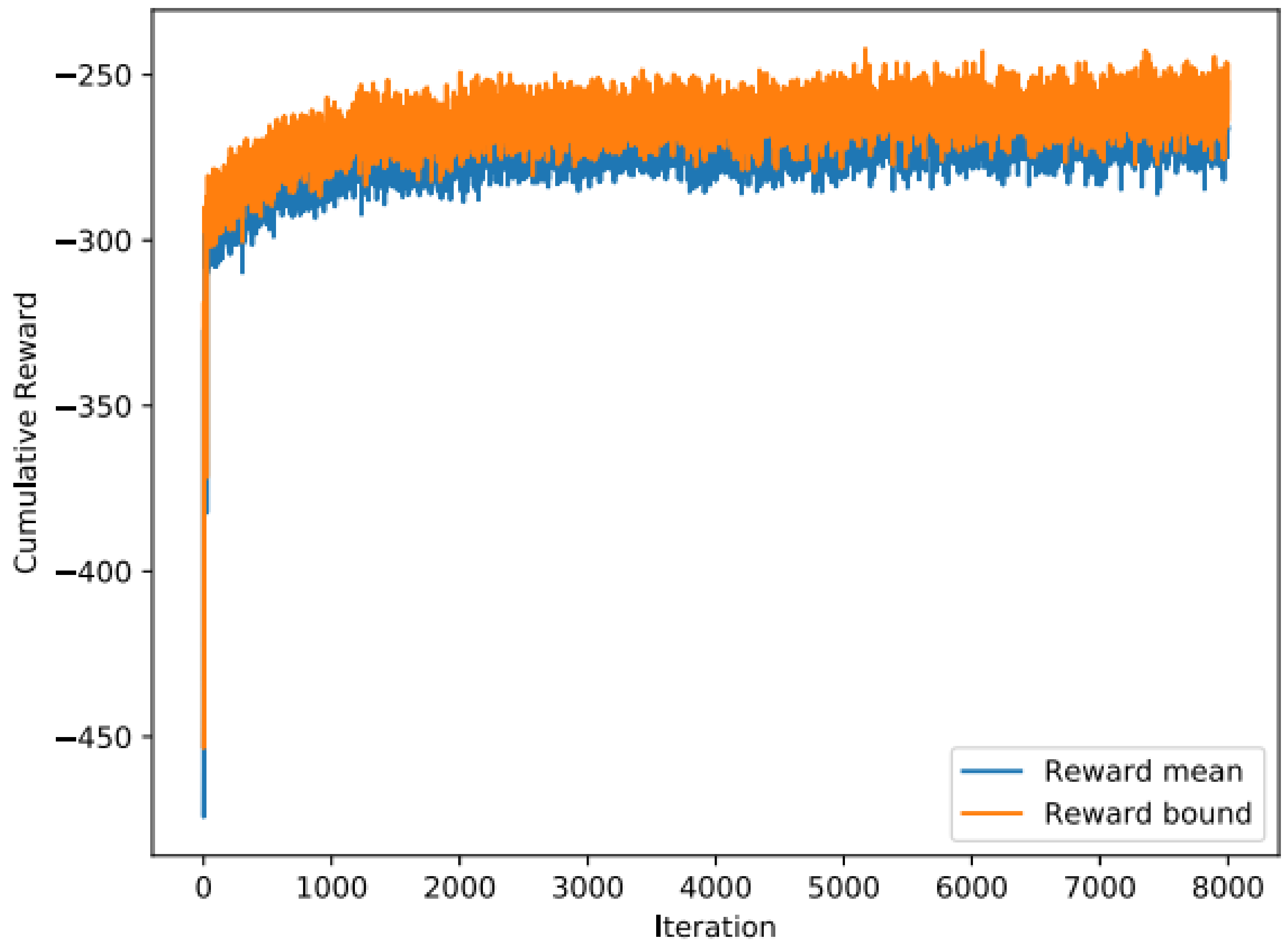


Fig. 8: The learning curve of *Raven* in datacenter with 260 physical machines and 520 virtual machines.

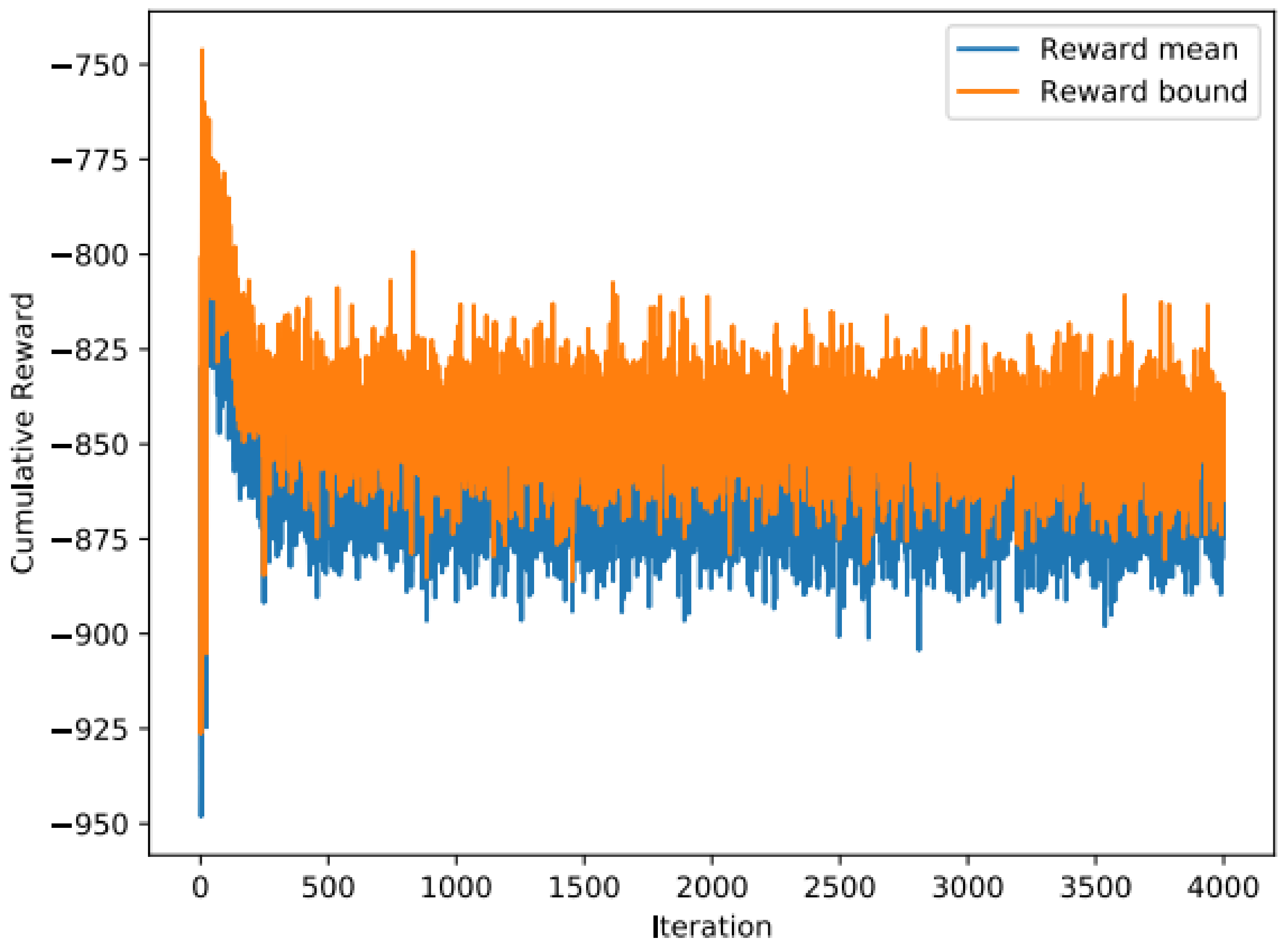


Fig. 9: The learning curve of *Raven* in datacenter with 260 physical machines and 1158 virtual machines.

RESULTS CTD

Table 1: Total migration time within different datacenters.

Datacenter Setting			Total Migration Time		
Number of PMs	Number of VMs	Number of Aggregation Switches	Min-DIFF	Heuristic	*Raven*
50	100	0	35.56 (11%)	57.50 (45%)	31.50
50	100	2	130.5 (27%)	110.0 (14%)	94.00
50	150	0	55.50 (31%)	103.00 (63%)	38.00
50	150	2	168.00 (16%)	159.00 (11%)	140.50
100	200	0	43.50 (28%)	91.50 (66%)	31.00
100	200	2	227.00 (35%)	221.50 (34%)	146.00
100	300	0	65.50 (2%)	192.50 (66%)	63.99
100	300	2	339.00 (15%)	378.00 (24%)	286.00
260	520	6	791.00 (58%)	510.00 (34%)	332.14
260	520	7	681.00 (35%)	445.78 (1%)	440.15
260	520	8	651.25 (58%)	440.00 (39%)	268.00
260	1158	6	944.31 (7%)	924.42 (5%)	869.97
260	1158	7	903.00 (10%)	910.50 (11%)	805.85
260	1158	8	918.50 (15%)	870.95 (11%)	774.23

RESULTS CTD

Same datacenter settings but with 10 different VM mappings

Table 2: Average total migration time within different datacenter.

Datacenter Setting			Average Total Migration Time		
Number of PMs	Number of VMs	Number of Aggregation Switches	Min-DIFF	Heuristic	*Raven*
50	100	0	36.20 (0%)	57.55 (36%)	36.39
50	100	2	117.00 (1%)	107.38 (-7%)	115.13
50	150	0	47.25 (-1%)	94.50 (49%)	47.94
50	150	2	172.95 (9%)	165.83 (5%)	157.00
100	200	0	53.05 (17%)	100.16 (56%)	43.54
100	200	2	248.24 (15%)	214.80 (2%)	209.14
100	300	0	66.80 (7%)	156.65 (60%)	61.96
100	300	2	335.95 (12%)	299.65 (1%)	296.32
260	520	6	725.55 (46%)	451.52 (14%)	388.94
260	520	7	667.70 (35%)	465.76 (7%)	433.17
260	520	8	622.65 (45%)	420.08 (19%)	338.65
260	1158	6	971.67 (12%)	1020.25 (17%)	846.06
260	1158	7	912.01 (14%)	987.62 (20%)	782.55
260	1158	8	836.54 (10%)	969.42 (22%)	748.93

CONCLUSIONS

Needed Improvements:

- consider live VM migration
- struggles to converge under certain settings

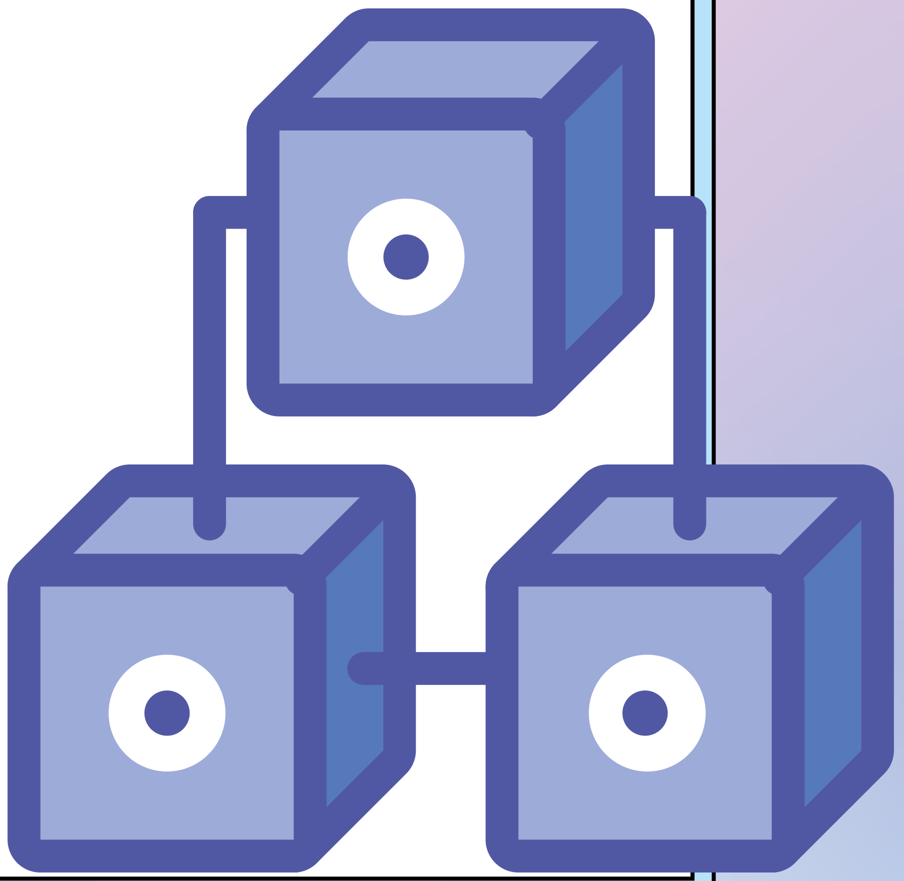
Overall

- ✓ closer to a real-life use case scenario
- ✓ does better in larger network topologies
- ✓ still has the shortest times overall compared to:
    - Min-DIFF
    - Heuristic evaluation

HOW DOES THIS RELATE?

- Deep reinforcement learning looks more promising to pursue
  - large fat tree, harder to maintain a traditional Q Table

- Similar unique example, with no obvious starting event
  - good example to 'translate'



Works Cited

Ying, C., Li, B., Xiaodi, K., & Guo, L. (2020). Raven : Scheduling Virtual Machine Migration During Datacenter Upgrades with Reinforcement Learning. *Mobile Networks and Applications*, 1-12.

Thank you!

Any questions, comments, and/or concerns?