Location-Aware and Budget-Constrained Service Brokering in Multi-Cloud via Deep Reinforcement Learning

Tao Shi, Hui Ma, Gang Chen, Sven Hartmann

School of Engineering and Computer Science, Victoria University of Wellington, New Zealand

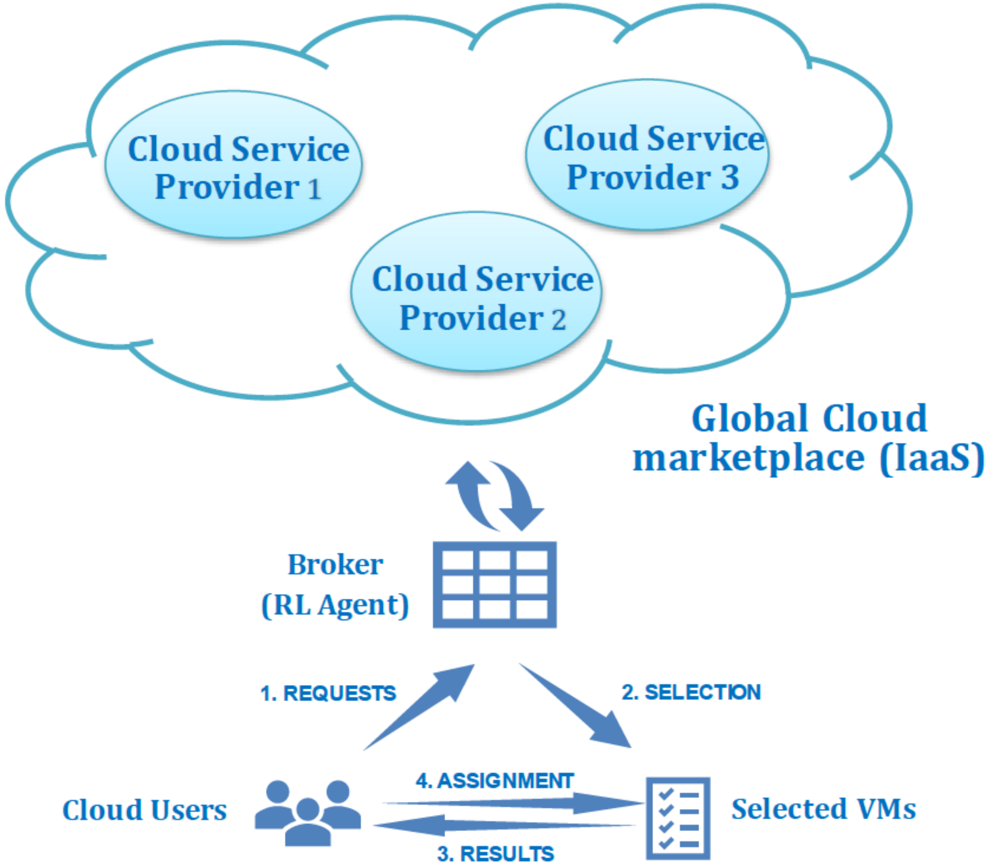
Department of Informatics, Clausthal University of Technology, Germany

CONTENTS

Introduction

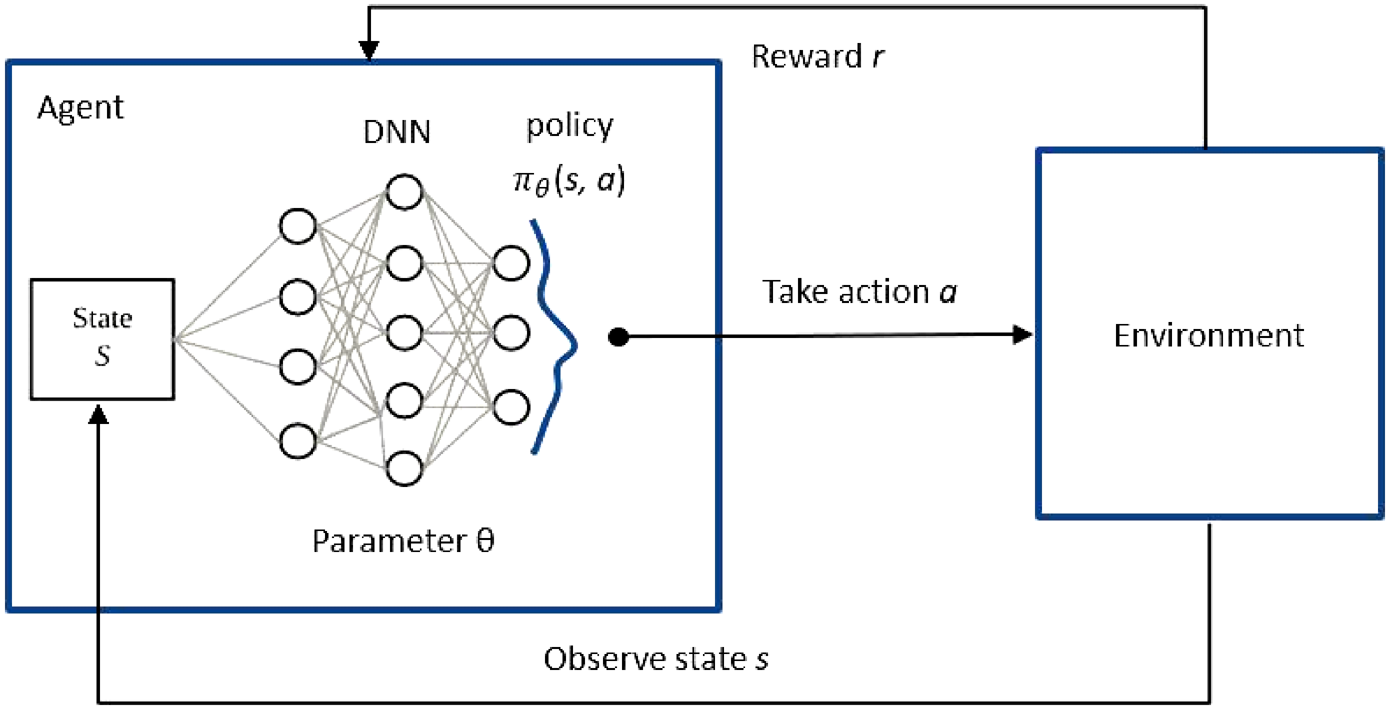
Service Brokering in Multi-Cloud

- Multi-cloud
  - High quality of services
  - Low operation cost
  - Avoid vendor lock-in
- Broker
  - Deployment and management of cloud services
- Location
  - Network latency
  - Cost
- Budget
  - Budgetary control to ensure financial viability



- Location-aware and Budget-constrained Service Brokering in Multi-cloud: LBSBM

Deep Reinforcement Learning (DRL)



- Maximize the overall reward during a long period of system operation
- Computationally efficient selection

Introduction

Challenges of DRL for LBSBM

- Unlimited VM instances available in multi-cloud
  - Novel components: state extractor and action executor

- Meet the budgetary constraint
  - New penalty-based reward function

Problem Description

ICSOC 2021

Location-aware and Budget-constrained Service Brokering in Multi-cloud: LBSBM

➢ VM types

- Capacities: CPU, memory
- Available regions

➢ Requests

- Resource requirements: CPU, memory
- Arrival time, duration
- User location
- Capacity-feasible assignment

➢ Assumptions

- The same VM instance can be used to process one request or multiple requests sequentially.
- Each request can only be assigned to a single VM instance.
- VM usage is charged on an hourly rate.

Problem Description

Location-aware and Budget-constrained Service Brokering in Multi-cloud: LBSBM

- Total cost (TC) and average network latency (ANL)

$$TC = \sum_{v \in V} \sum_{r \in R_v} C_{v,r} x_{v,r}, \ ANL = \frac{1}{N} \sum_{i=1}^{N} \sum_{v \in V} \sum_{r \in R_v} L_{i,r} y_{i,v,r}$$

- Objective

Minimize *ANL*

- Constraint

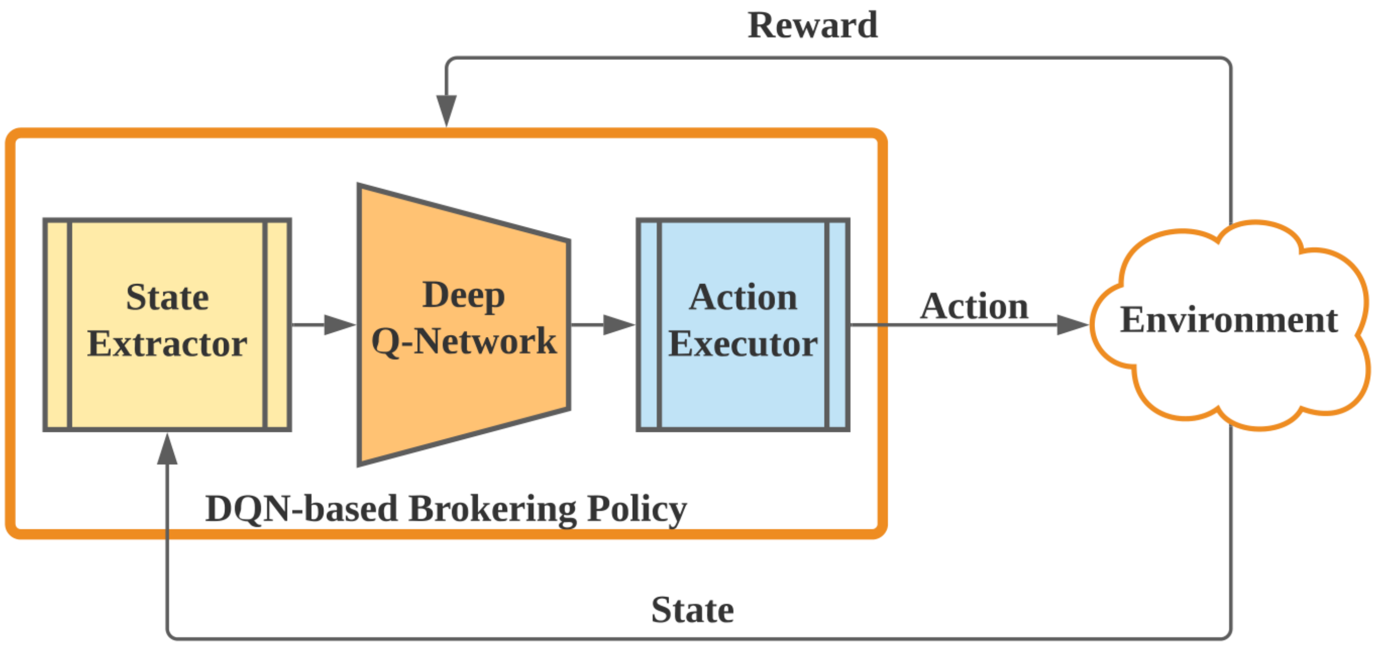
$$TC \le \sum_{i=1}^{N} b_i$$

$$b_i = C_{i,min} t_i + k \cdot (C_{i,max} t_i - C_{i,min} t_i)$$

DeepBroker: A DRL-Based Algorithm

DRL system

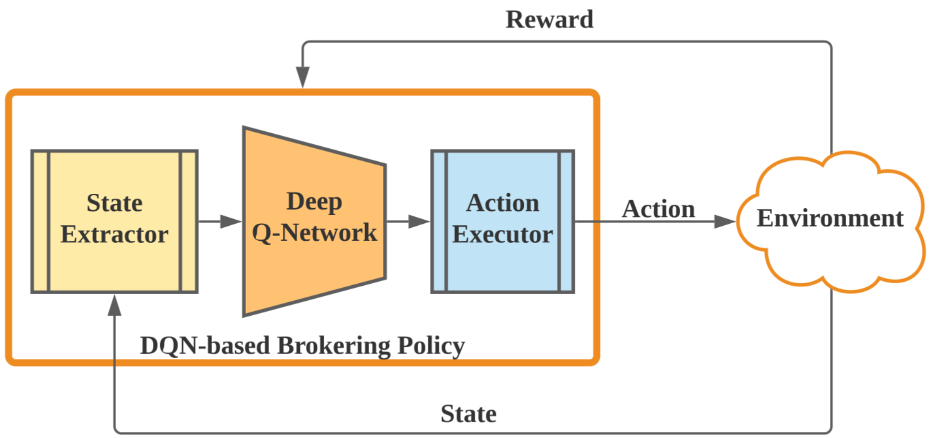
- State: The observed state includes the new request and currently leased VM instances.
- Action: To select a specific instance of capacity-feasible VM types, i.e., an idle VM instance or a newly selected VM instance, for request.



DeepBroker: A DRL-Based Algorithm

State Extractor

- New request
  - Resource requirements: CPU, memory
  - Duration
  - User location: Latency vector including the network latency between the user location and all the regions covered by multi-cloud data centers.
- Leased VM instances
  - For each VM type in each available region
  - Capacity feasible?
  - Idle? Maximum remaining time



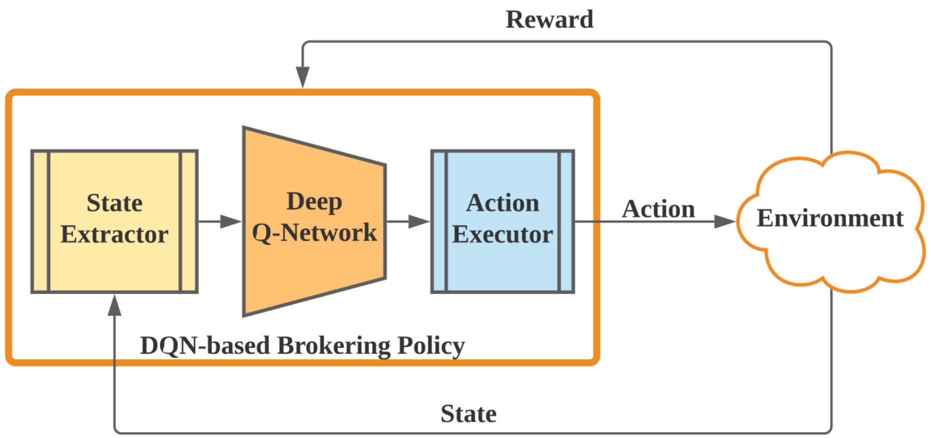
DeepBroker: A DRL-Based Algorithm

DRL for Training DQN

- Penalty-based reward function

$$r_i = -L_{i,r} - \max(0, (C_{v,r} t_i - b_i))$$

- Q-learning
- DQNs as function approximators
- Experience replay

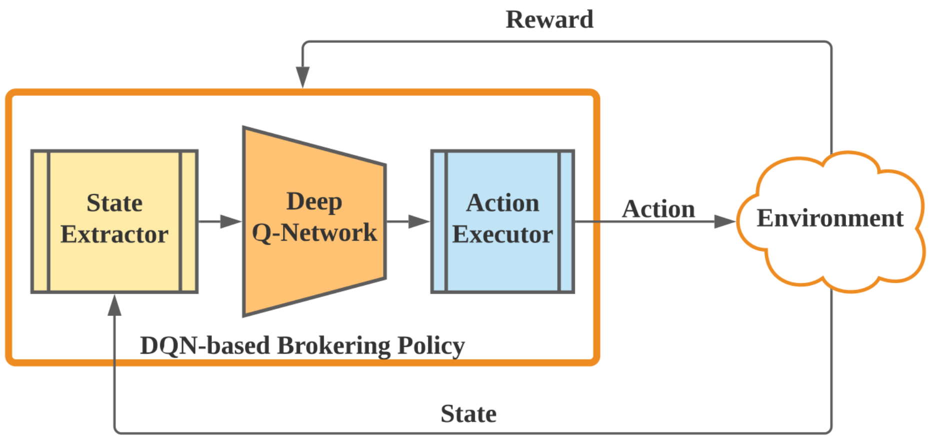


DeepBroker: A DRL-Based Algorithm

Action Executor

- Mask
  - Zero probabilities for capacity-infeasible VM types

- Specify VM instances
  - Idle? Maximum remaining time
  - New VM instance



Evaluation

Datasets

- VMs
  - Three leading cloud providers, i.e., Amazon Web Services (AWS), Microsoft Azure and Alibaba Elastic Compute Service (ECS)
  - 12 different VM types (4 from each)
  - 8 regions for major AWS, Azure and Alibaba data centers, i.e., Northern Virginia, Dublin, Singapore, Tokyo, Sydney, Northern California, Sao Paulo, and Mumbai.
- Requests
  - Public VM request workload in the Azure dataset [1]
- Network latency evaluation
  - Sprint IP backbone network databases

[1] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., Bianchini, R.: Resource central: understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 153–167 (2017)

Algorithm Implementation

- PyTorch
- Two fully-connected hidden layers, each with 64 nodes
- ReLUs, MSE, Adam
- Initial and minimum probability that DRL randomly chooses an action: 0.2 and 0.01
- Learning rate and discount factor: 0.001and 1.0
- Mini-batch size: 32

Evaluation

Baseline

➢ Greedy

- Assigns each user request to the best possible VM instance in terms of the weighted sum of the normalized cost and network latency.
- Tune an appropriate combination of weights based on the training workload.

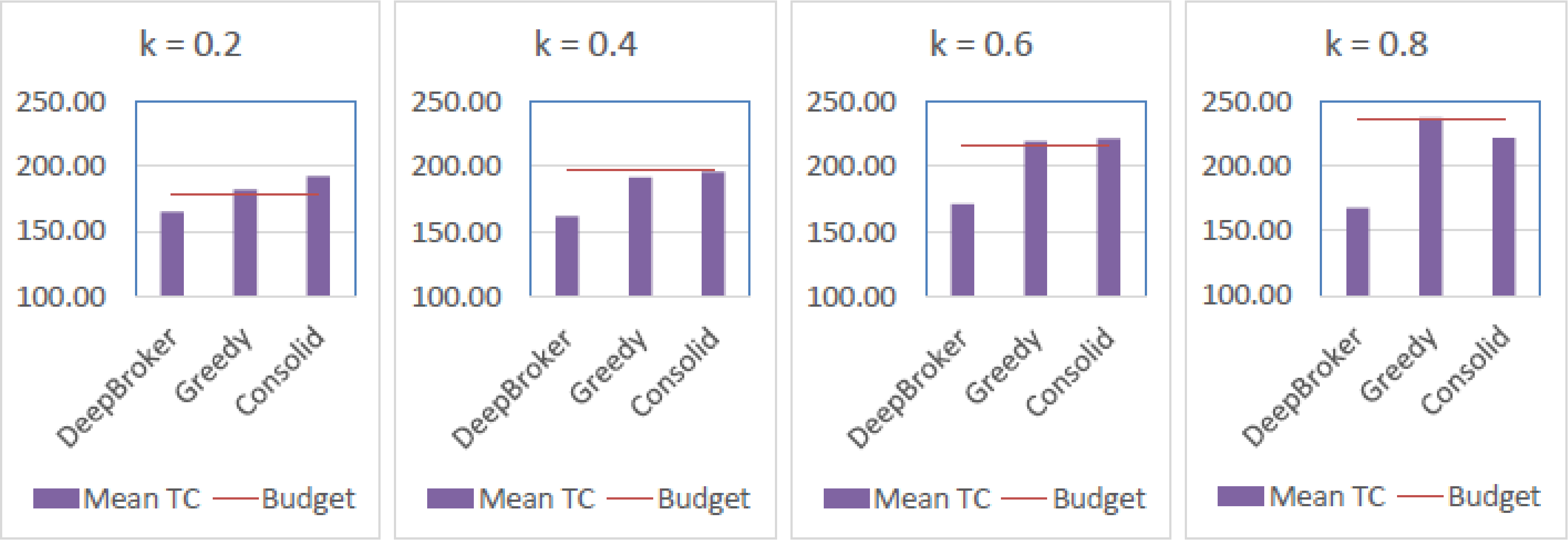
➢ Consolid

- Identifies a set of eligible processors with enough resources that can process the arriving request.
- Assigns the request to the processor with the highest post-allocation utilization.

Evaluation

Results: Budget Compliance

- Training workload: one day's workload with 365 user requests
- Test workload: the following day’s workload with 362 user requests
- 4 budget factors: 0.2, 0.4, 0.6, and 0.8,



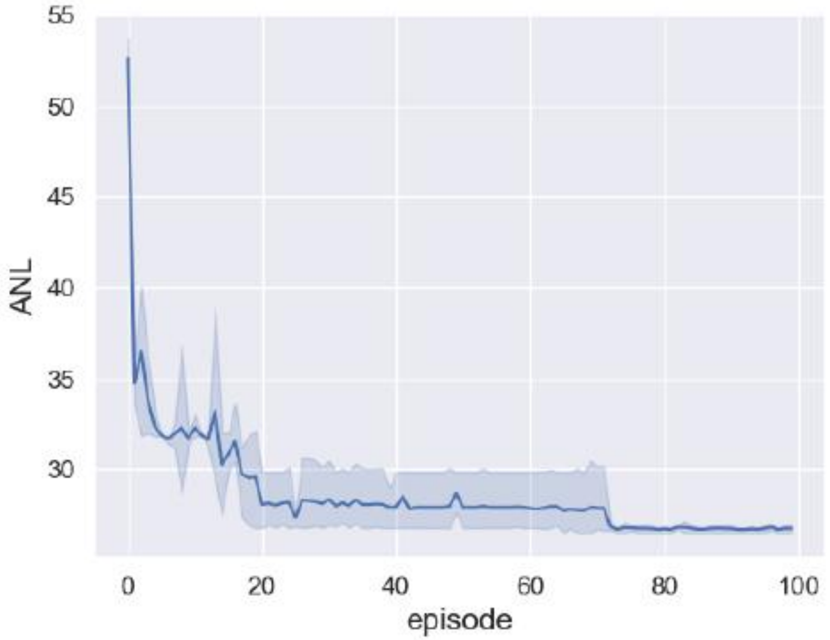
Results: ANL Comparison and VM Instance Numbers

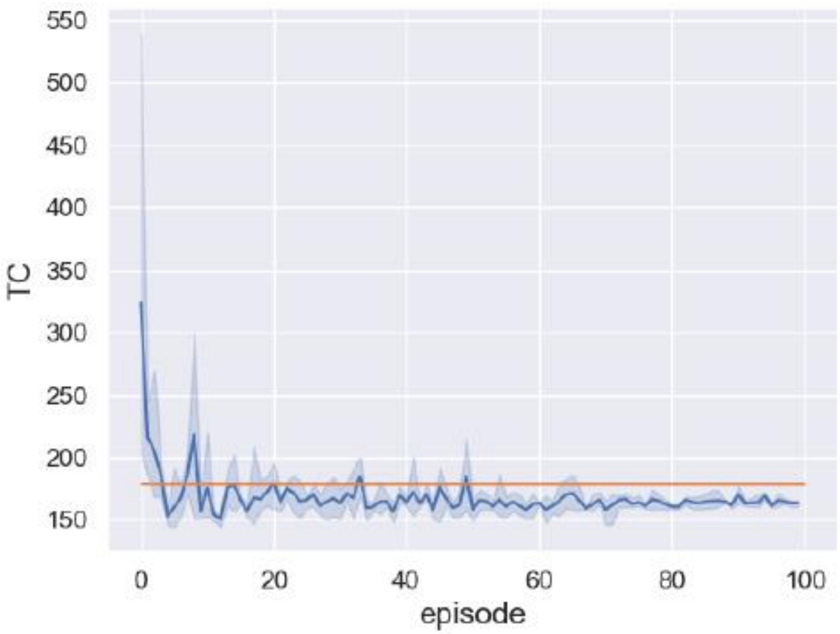
Algorithm performance comparison for the *LBSBM* problem under different budget factors (*ANL* in ms., Total budget and *TC* in USD, the best is bold).

k	Total budget	DeepBroker			Greedy based on [3]					Consolid based on [13]		
		ANL	TC	n	ANL	TC	ω_1	ω_2	n	ANL	TC	n
0.2	178.71	$\mathbf{26.83 \pm 0.23}$	164.98 ± 4.8	249	124.4 ± 0	182.21 ± 0	1.00	0.00	181	105.73 ± 0	192.39 ± 0	172
0.4	197.68	$\mathbf{26.93 \pm 0.3}$	162.18 ± 3.61	253	61.83 ± 0	191.75 ± 0	0.95	0.05	221	97.53 ± 0	196.1 ± 0	171
0.6	216.65	$\mathbf{26.86 \pm 0.2}$	171.65 ± 5.83	245	38.17 ± 0	219.09 ± 0	0.85	0.15	225	73.35 ± 0	221.56 ± 0	172
0.8	235.62	$\mathbf{27.01 \pm 0.29}$	167.82 ± 6.58	249	26.55 ± 0	237.6 ± 0	0.75	0.25	230	73.35 ± 0	221.56 ± 0	172

Evaluation

Convergence Analysis and Computational Overhead





- Training time: less than 30 min
- Computational overhead: within 1ms

Conclusions

Conclusions

- The paper studies the LBSBM problem, i.e., selecting VMs for arriving user requests to minimize the average network latency of VMs subject to the total budget over a time span.

- We propose a DRL-based algorithm, named DeepBroker, with the problem-specific state extractor, action executor, and penalty-based reward function to train the DQN-based service brokering policies.

- The experiments based on the real-world datasets show the trained brokering policies significantly outperform several heuristic-based algorithms in terms of both average network latency and budget satisfaction.

Thank You
Q&A

Presenter: Tao Shi

Emails: tao.shi@ecs.vuw.ac.nz