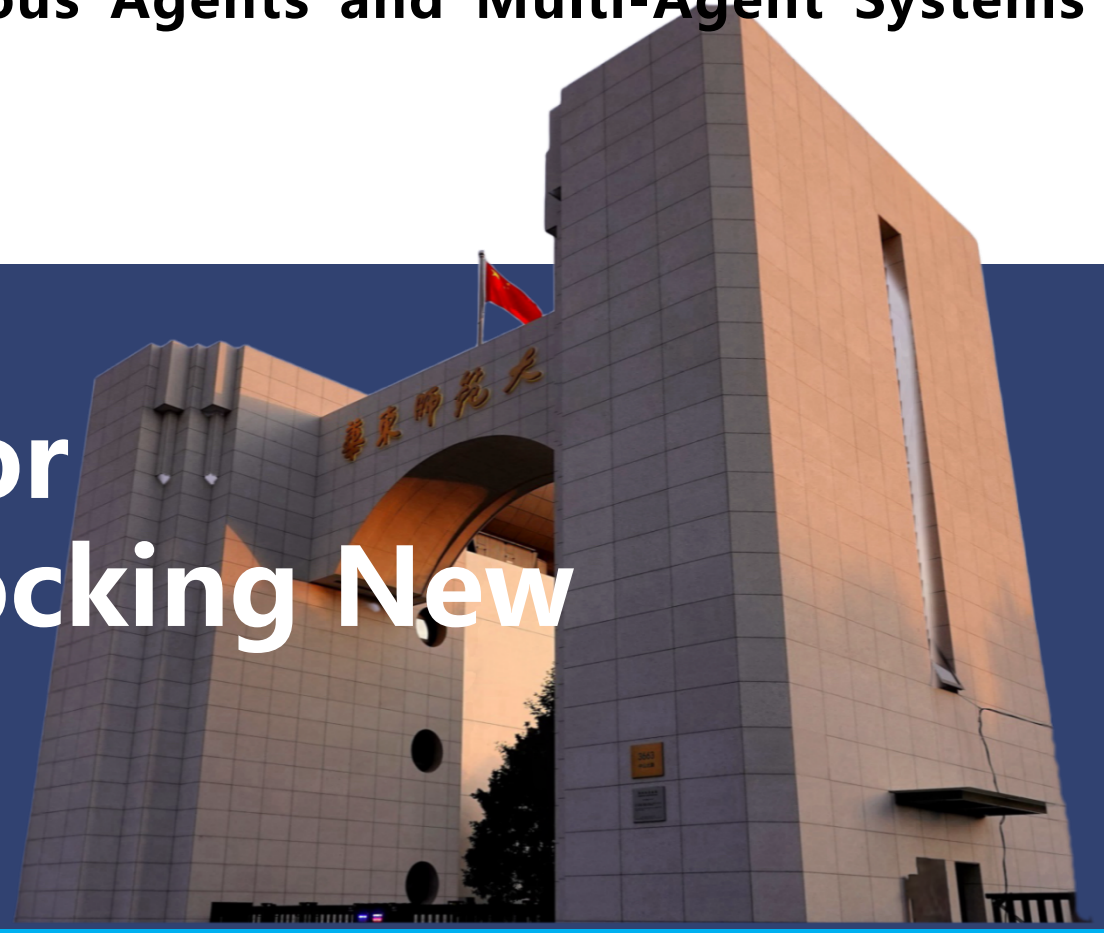
The 23rd International Conference on Autonomous Agents and Multi-Agent Systems



Reinforcement Learning for Operations Research: Unlocking New Possibilities (Part II)

Xiangfeng Wang, **Junjie Sheng** (East China Normal University)
Wenhao Li (The Chinese University of Hong Kong, Shenzhen)
Contact email: liwenhao@cuhk.edu.cn

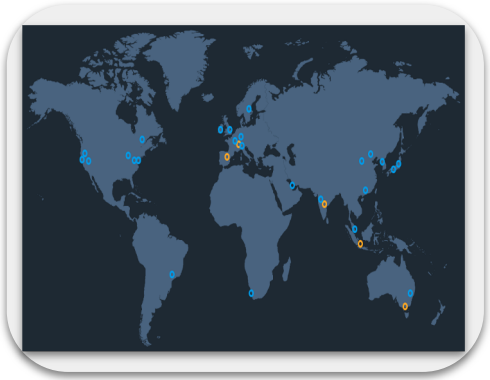
The 23rd International Conference on Autonomous Agents and Multi-Agent Systems

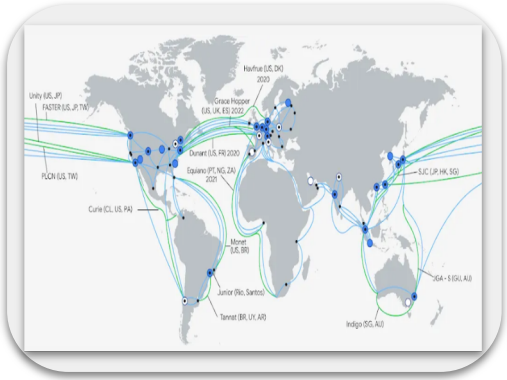
Reinforcement Learning for Cloud Resource Scheduling

Xiangfeng Wang, **Junjie Sheng** (East China Normal University)
Wenhao Li (The Chinese University of Hong Kong, Shenzhen)
Contact email: liwenhao@cuhk.edu.cn

Cloud Resource Scheduling: Background

Cloud computing market grown fast in recent years.





aws

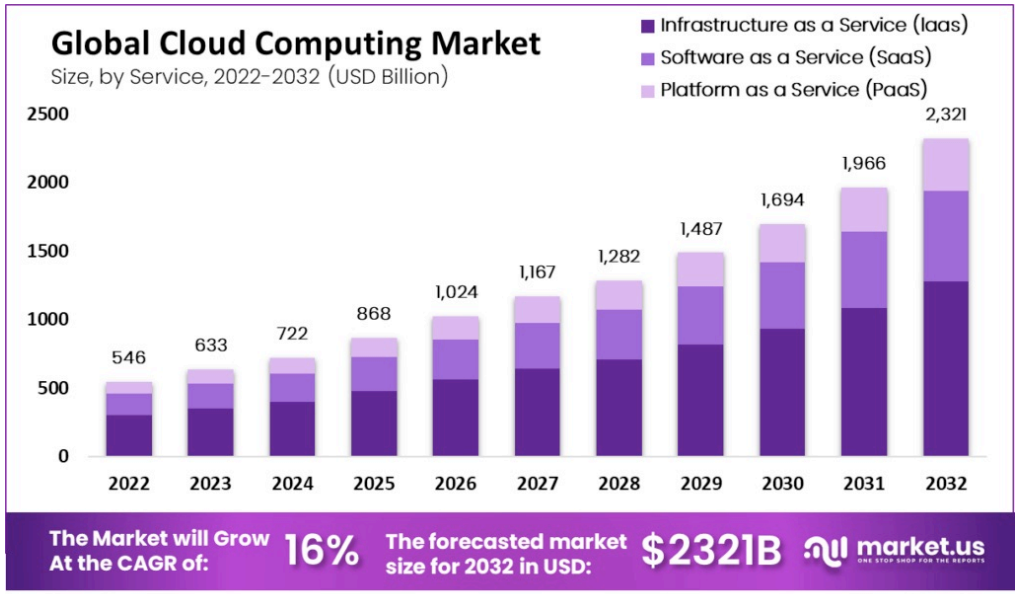
Azure

HUAWEI CLOUD

Tencent Cloud

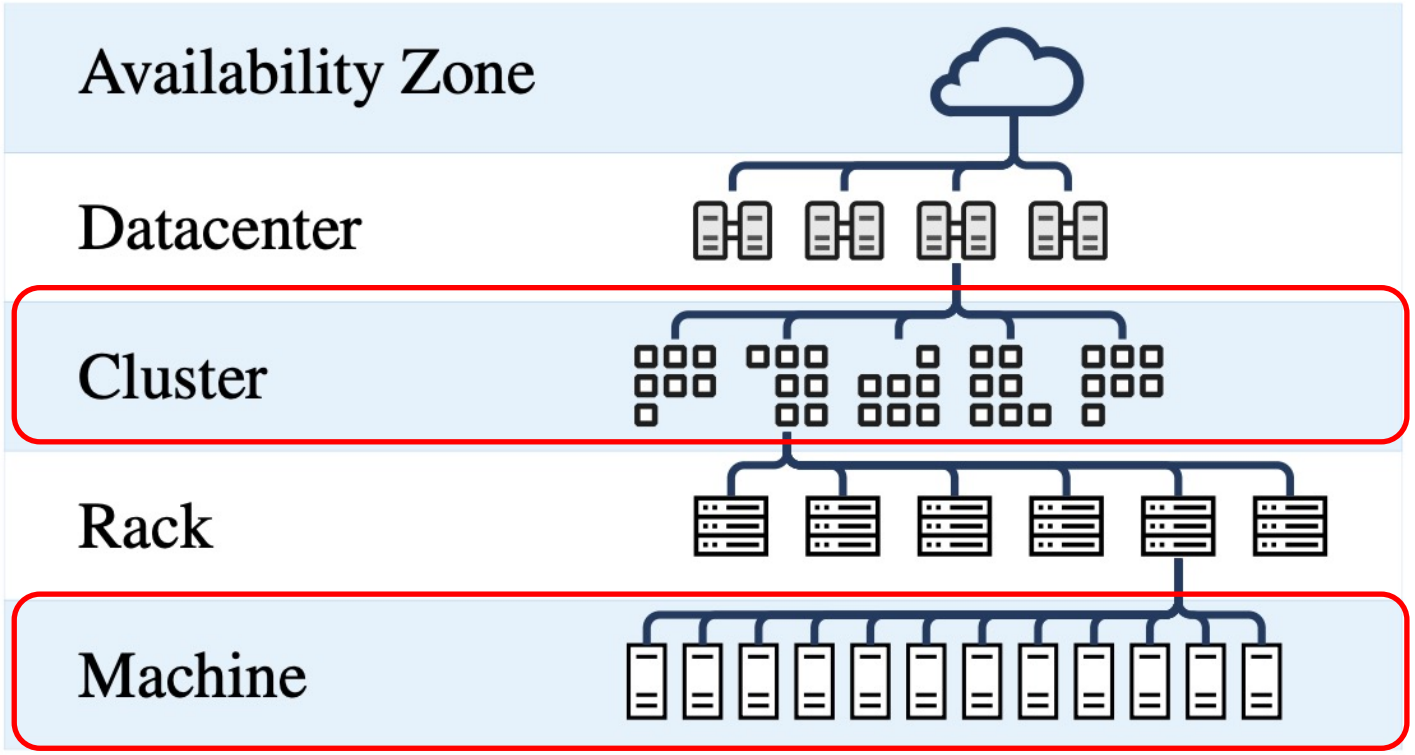
Alibaba Cloud

BAIDU AI CLOUD



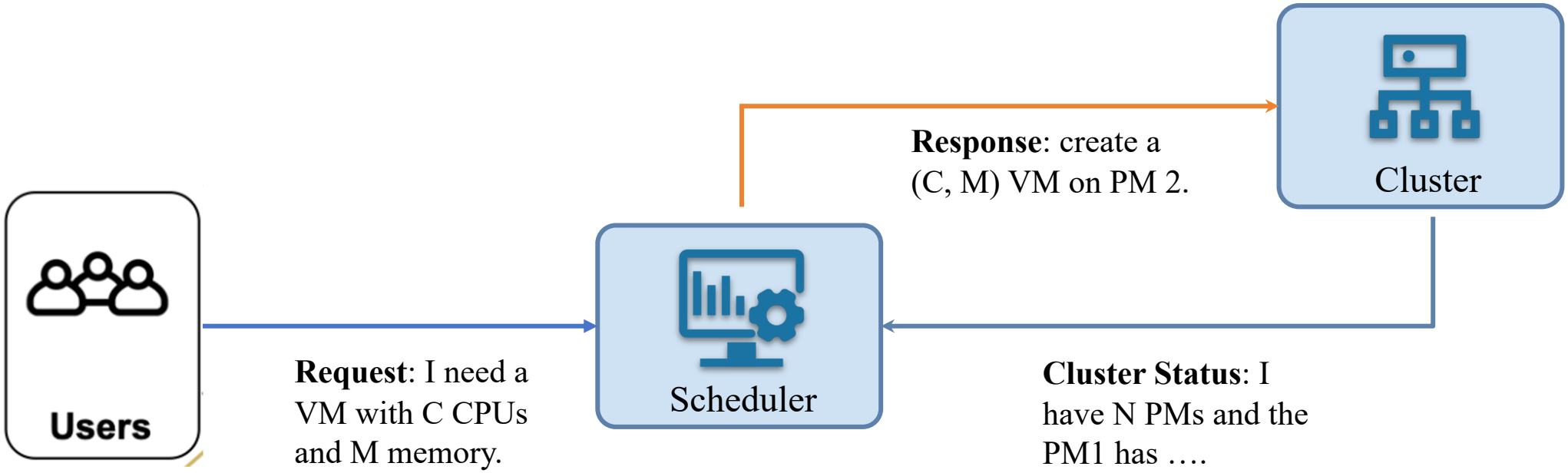
Cloud Resource Scheduling: Background

Cloud Service Providers hold physical resources in hierarchy.



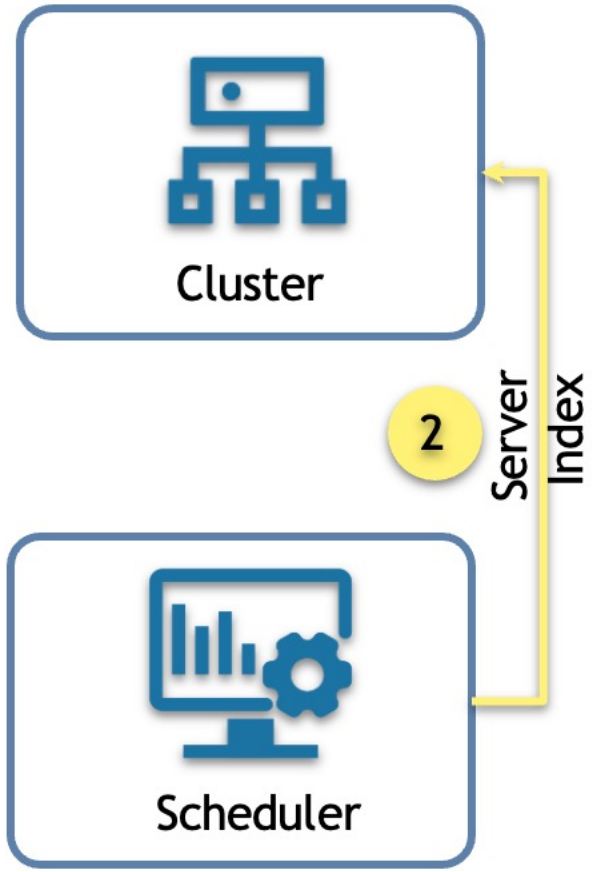
Cloud Resource Scheduling: Key Concepts

Cloud Resource Scheduling involves three components.



Cloud Resource Scheduling: Heuristic Methods

Heuristic Schedulers are prevailing in industries.



First Fit Method

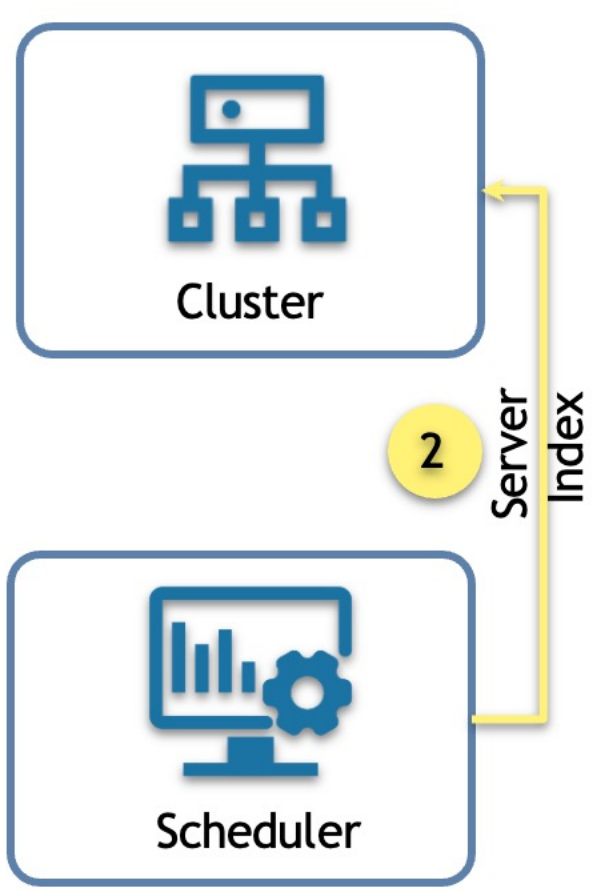
Use the first physical machine that can allocate the current request.

Best Fit Method

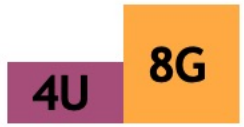
Use the physical machine that can allocate the current request while have smallest remaining computational resources.

Cloud Resource Scheduling: Heuristic Methods

Heuristic Schedulers are prevailing in industries.



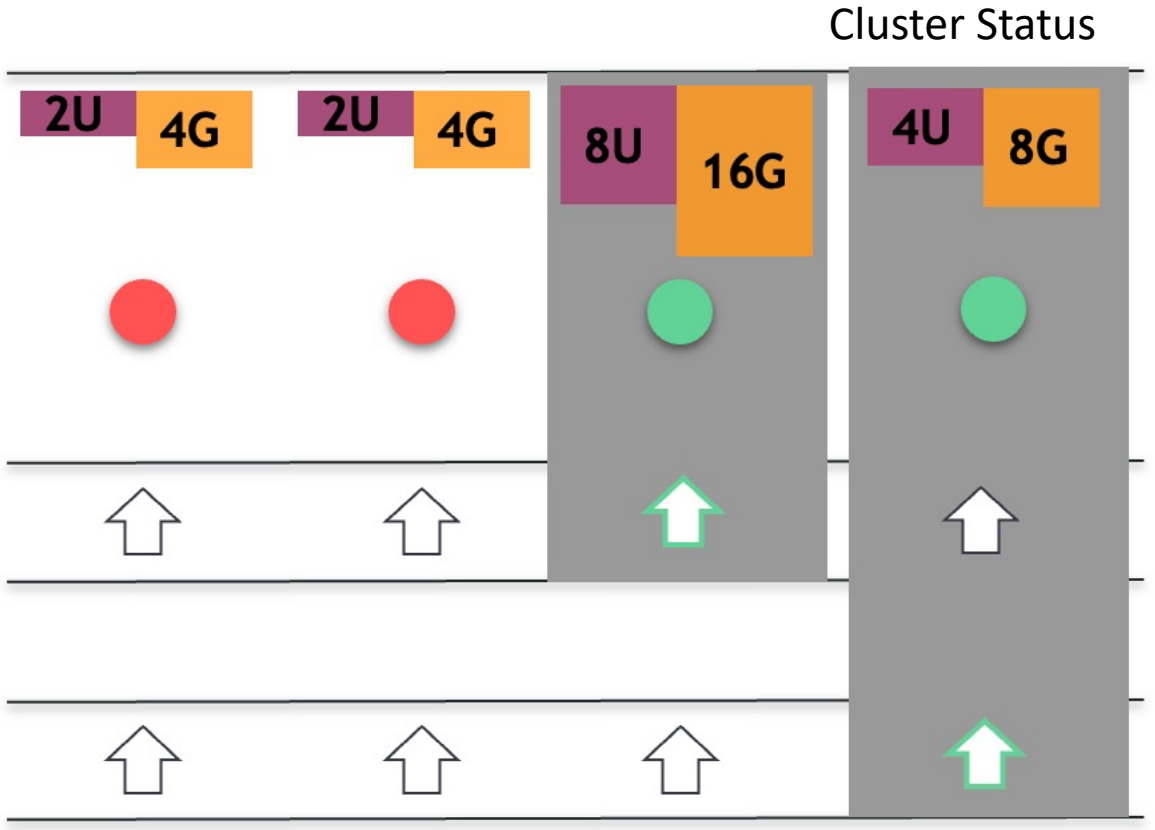
Request Status



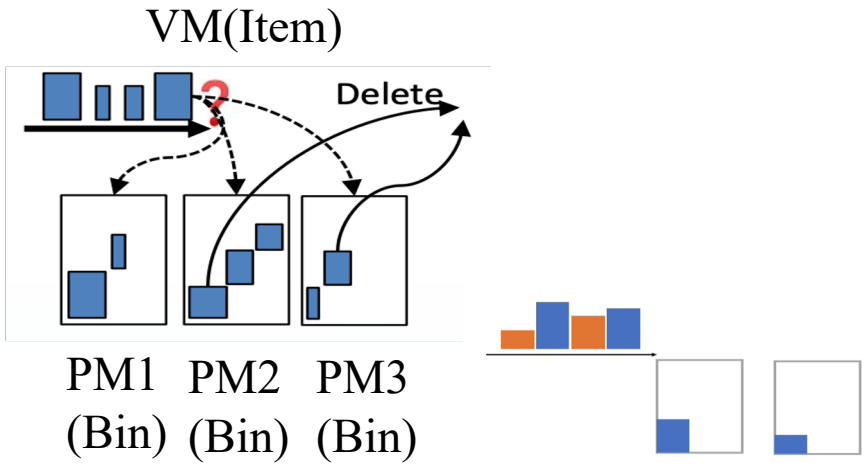
First Fit Method

Best Fit Method

Cluster Status



Cloud Resource Scheduling: Combinatorial Optimization



Can be treated as a **Bin-Packing** Problem.
Objective: maximize number of packed VMs

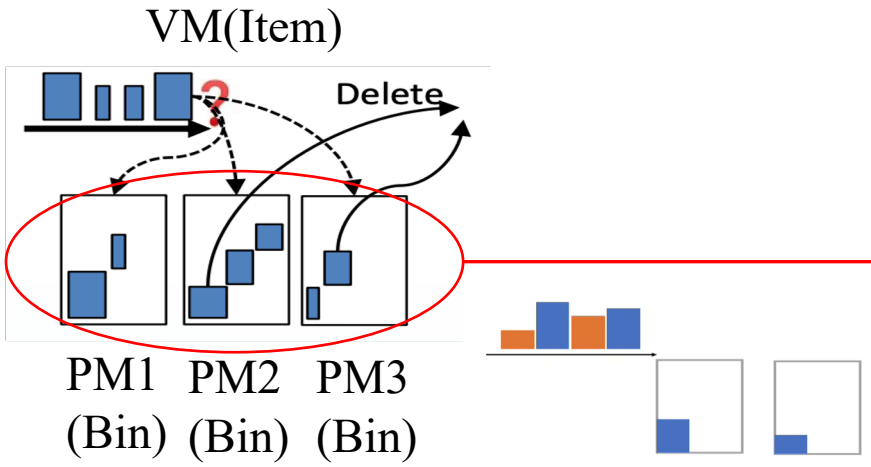
$$\begin{aligned}
\max_{\text{res},w} \quad & \|w\|_1, \\
\text{s.t.} \quad & \text{res} \in C_s \cap C_c, \\
& w(1) = \begin{cases} 0, & \text{if } \text{res}(1) = (0,0), \\ 1, & \text{otherwise}, \end{cases} \\
& w(t) = \begin{cases} 0, & \text{if } \text{res}(t) = (0,0) \text{ or } w(t-1) = 0, \\ 1, & \text{otherwise}, \end{cases} \qquad \text{for } 2 \le t \le T.
\end{aligned}$$

$$\begin{cases}
a(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=0,\text{res}(k,\cdot)=(i,j)} \\
\qquad\qquad + \frac{1}{M} \mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=1,\text{res}(k,\cdot)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]), \\
b(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=0,\text{res}(k)=(i,j)} \\
\qquad\qquad + \frac{1}{M} \mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=1,\text{res}(k)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]).
\end{cases}$$

$$\text{res} \in \mathcal{C}_s, \quad \mathcal{C}_s := \left\{ \text{res} \;\middle|\; \begin{array}{c} s(i,j,\cdot) - a(t,i,j) + b(t,i,j) \succeq 0, \\ \forall\, 0 \le i \le N, \forall\, 0 \le j \le M, \forall\, t \le T \end{array} \right\}.$$

$$\text{res} \in \mathcal{C}_c, \; \mathcal{C}_c := \left\{ \text{res} \;\middle|\; \text{res}(t,\cdot) = \text{res}(t',\cdot), \text{ if } \text{req}(t,1) = \text{req}(t',1), \; \forall\, t,t' \le T \right\}.$$

Cloud Resource Scheduling: Combinatorial Optimization



Can be treated as a **Bin-Packing** Problem.
Objective: maximize number of packed VMs
Constraints:

1. Capacity Constraint:

The remaining capacity of each PM must be non-negative.

$$\max_{\text{res},w} \quad \|w\|_1,$$

$$\text{s.t.} \quad \text{res} \in C_s \cap C_c,$$

$$w(1) = \begin{cases} 0, & \text{if res}(1) = (0,0), \\ 1, & \text{otherwise}, \end{cases}$$

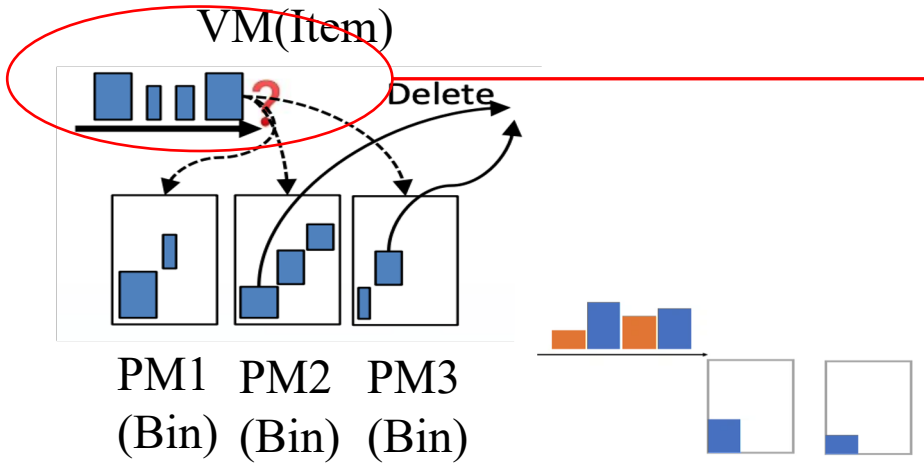
$$w(t) = \begin{cases} 0, & \text{if res}(t) = (0,0) \text{ or } w(t-1) = 0, \\ 1, & \text{otherwise}, \end{cases} \quad \text{for } 2 \le t \le T.$$

$$\begin{cases} a(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=0,\text{res}(k,\cdot)=(i,j)} \\ \qquad + \frac{1}{M} \mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=1,\text{res}(k,\cdot)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]), \\ b(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=0,\text{res}(k)=(i,j)} \\ \qquad + \frac{1}{M} \mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=1,\text{res}(k)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]). \end{cases}$$

$$\text{res} \in \mathcal{C}_s, \quad \mathcal{C}_s := \left\{ \text{res} \;\middle|\; \begin{array}{r} s(i,j,\cdot) - a(t,i,j) + b(t,i,j) \succeq 0, \\ \forall\, 0 \le i \le N, \forall\, 0 \le j \le M, \forall\, t \le T \end{array} \right\}.$$

$$\text{res} \in \mathcal{C}_c, \; \mathcal{C}_c := \left\{ \text{res} \;\middle|\; \text{res}(t,\cdot) = \text{res}(t',\cdot), \text{ if } \text{req}(t,1) = \text{req}(t',1), \; \forall\, t,t' \le T \right\}.$$

Cloud Resource Scheduling: Combinatorial Optimization



Can be treated as a **Bin-Packing** Problem.

Objective: maximize number of packed VMs

Constraints:

1. Capacity Constraint
2. Temporary Constraint

All VM requests must be handled sequentailly.

$$\max_{\text{res},w} \quad \|w\|_1,$$

$$\text{s.t.} \quad \text{res} \in C_s \cap C_c,$$

$$w(1) = \begin{cases} 0, & \text{if res}(1) = (0,0), \\ 1, & \text{otherwise}, \end{cases}$$

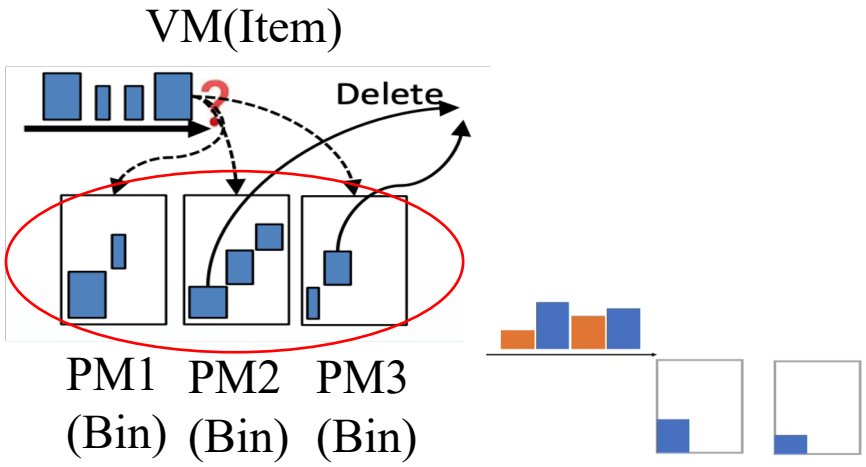
$$w(t) = \begin{cases} 0, & \text{if res}(t) = (0,0) \text{ or } w(t-1) = 0, \\ 1, & \text{otherwise}, \end{cases} \quad \text{for } 2 \le t \le T.$$

$$\begin{cases} a(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=0,\text{res}(k,\cdot)=(i,j)} \\ \qquad + \frac{1}{M} \mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=1,\text{res}(k,\cdot)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]), \\ b(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=0,\text{res}(k)=(i,j)} \\ \qquad + \frac{1}{M} \mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=1,\text{res}(k)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]). \end{cases}$$

$$\text{res} \in \mathcal{C}_s, \quad \mathcal{C}_s := \left\{ \text{res} \;\middle|\; \begin{array}{r} s(i,j,\cdot) - a(t,i,j) + b(t,i,j) \succeq 0, \\ \forall\, 0 \le i \le N, \forall\, 0 \le j \le M, \forall\, t \le T \end{array} \right\}.$$

$$\text{res} \in \mathcal{C}_c, \; \mathcal{C}_c := \left\{ \text{res} \;\middle|\; \text{res}(t,\cdot) = \text{res}(t',\cdot), \text{ if } \text{req}(t,1) = \text{req}(t',1), \; \forall\, t,t' \le T \right\}.$$

Cloud Resource Scheduling: Combinatorial Optimization



Can be treated as a **Bin-Packing** Problem.

Objective: maximize number of packed VMs

Constraints:

1. Capacity Constraint
2. Temporary Constraint

Challenges:

1. NP-Hard
2. Real time scheduling
3. Reqeust distribution is unkown

$$\max_{\text{res},w} \quad \|w\|_1,$$

$$\text{s.t.} \quad \text{res} \in C_s \cap C_c,$$

$$w(1) = \begin{cases} 0, & \text{if res}(1) = (0,0), \\ 1, & \text{otherwise}, \end{cases}$$

$$w(t) = \begin{cases} 0, & \text{if res}(t) = (0,0) \text{ or } w(t-1) = 0, \\ 1, & \text{otherwise}, \end{cases} \qquad \text{for } 2 \le t \le T.$$

$$\begin{cases} a(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=0,\text{res}(k,\cdot)=(i,j)} \\ \qquad\qquad + \frac{1}{M}\mathbb{1}_{\text{req}(k,2)=1,\text{req}(k,3)=1,\text{res}(k,\cdot)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]), \\ b(t,i,j) = \sum_{k=1}^{t} (\mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=0,\text{res}(k)=(i,j)} \\ \qquad\qquad + \frac{1}{M}\mathbb{1}_{\text{req}(k,2)=0,\text{req}(k,3)=1,\text{res}(k)=(i,M+1)}) \cdot \text{req}(k,[4:d+3]). \end{cases}$$

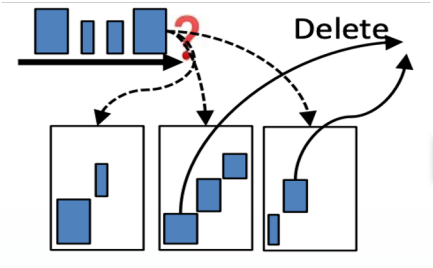
$$\text{res} \in \mathcal{C}_s, \quad \mathcal{C}_s := \left\{ \text{res} \;\middle|\; \begin{array}{c} s(i,j,\cdot) - a(t,i,j) + b(t,i,j) \succeq 0, \\ \forall\, 0 \le i \le N, \forall\, 0 \le j \le M, \forall\, t \le T \end{array} \right\}.$$

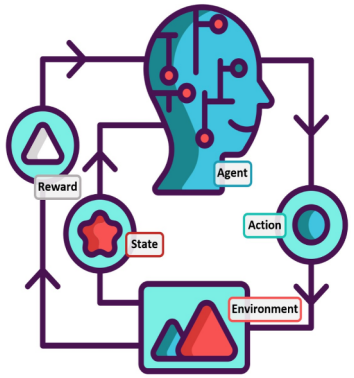
$$\text{res} \in \mathcal{C}_c, \; \mathcal{C}_c := \left\{ \text{res} \;\middle|\; \text{res}(t,\cdot) = \text{res}(t',\cdot), \text{ if } \text{req}(t,1) = \text{req}(t',1), \; \forall\, t,t' \le T \right\}.$$

Why Reinforcement Learning

Combinatorial Optimization	Heuristic Methods
➔ Optimal solutions	➔ Online scheduling
➔ Offline Scheduling	➔ Low computational complexity
➔ High computational complexity	➔ Sub-Optimal solutions

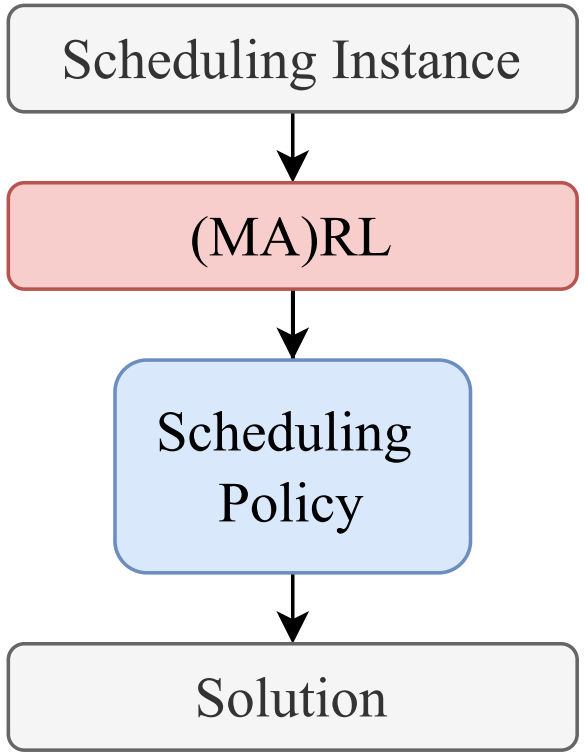
How can we achieve near-optimal solutions while implementing online scheduling with low computational complexity?



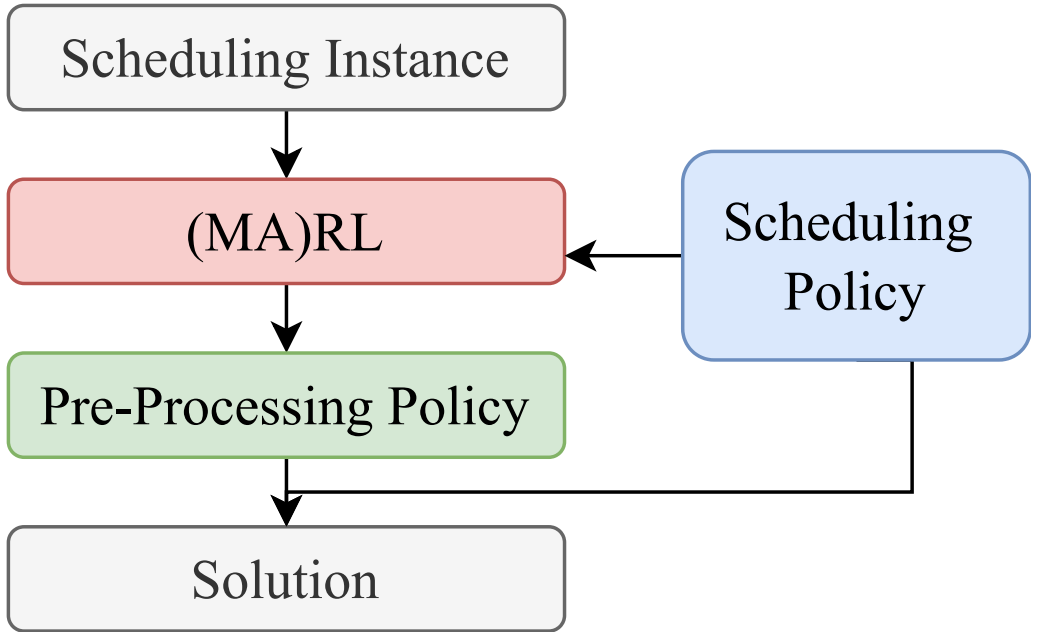


Reinforcement Learning

RL Methods For Cloud Resource Scheduling



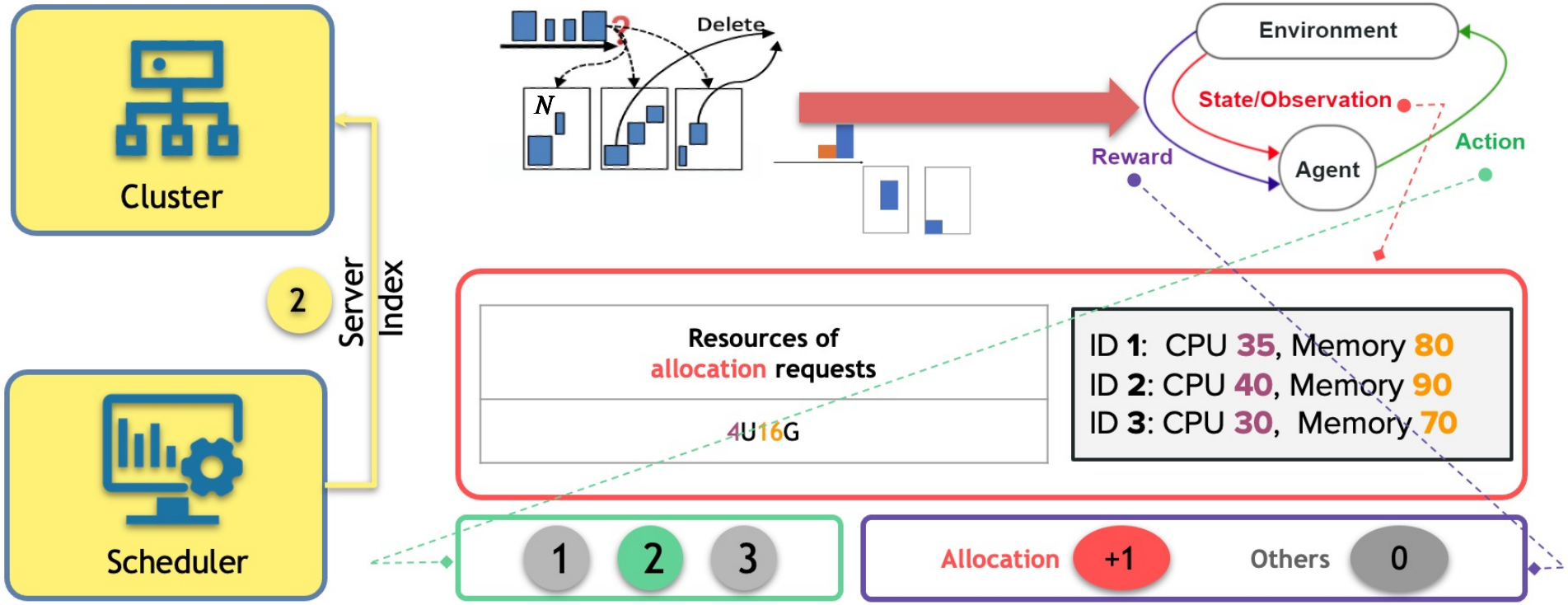
RL For (Direct) VM Scheduling



RL For Indirect VM Scheduling

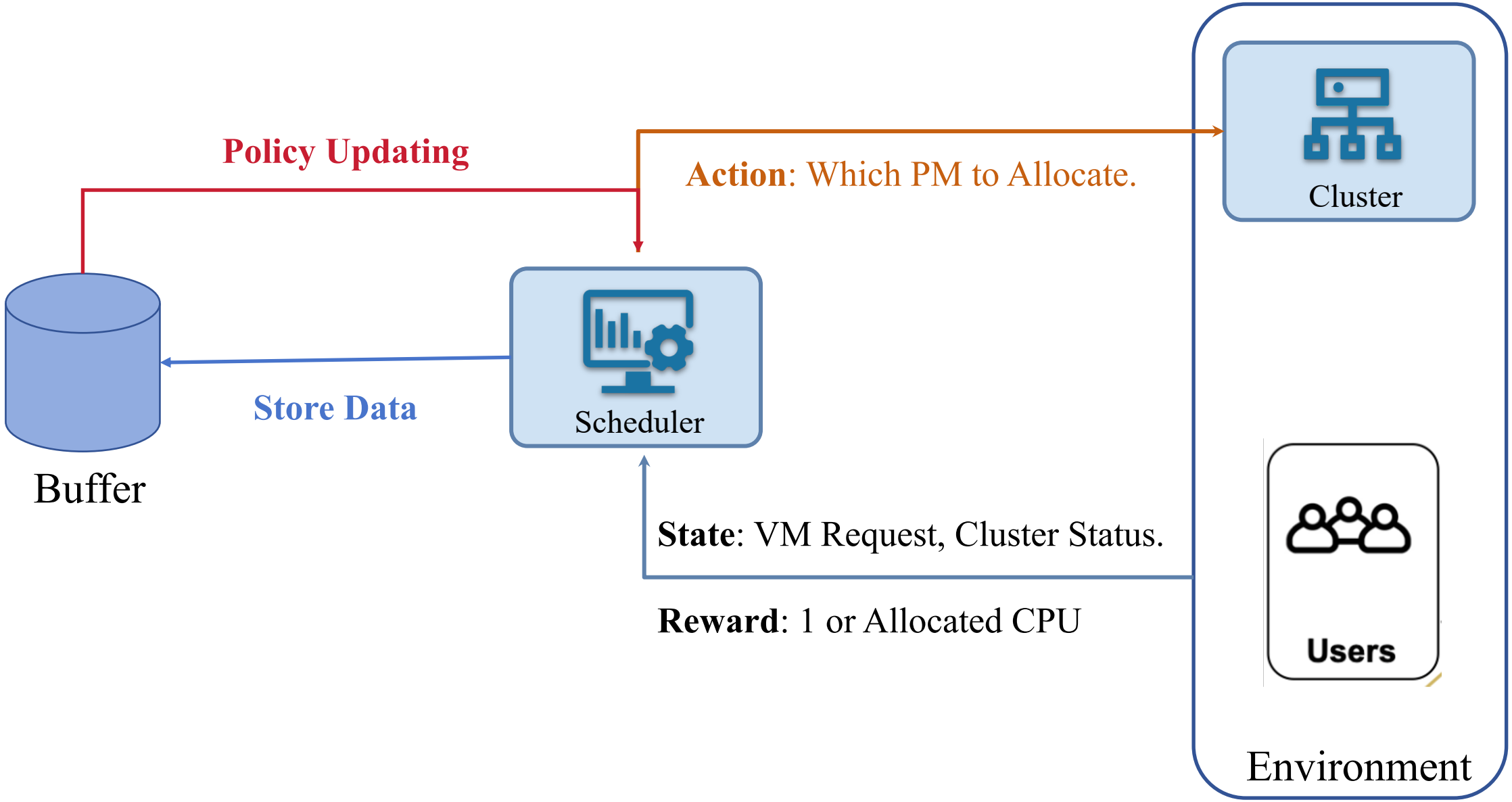
Reinforcement Learning For VM Scheduling

- ✓ State (Observation) Space: *N* servers with resources(CPU/Memory) and creation requests
- ✓ Action Space: server numbers {*1, 2, …, N*}
- ✓ State Transition: Transitioning from the current state *s* to the next state *s`* after placing a request
- ✓ Reward Setting: Successful placement +1, unsuccessful placement 0 (Is a simple reward setup appropriate?)

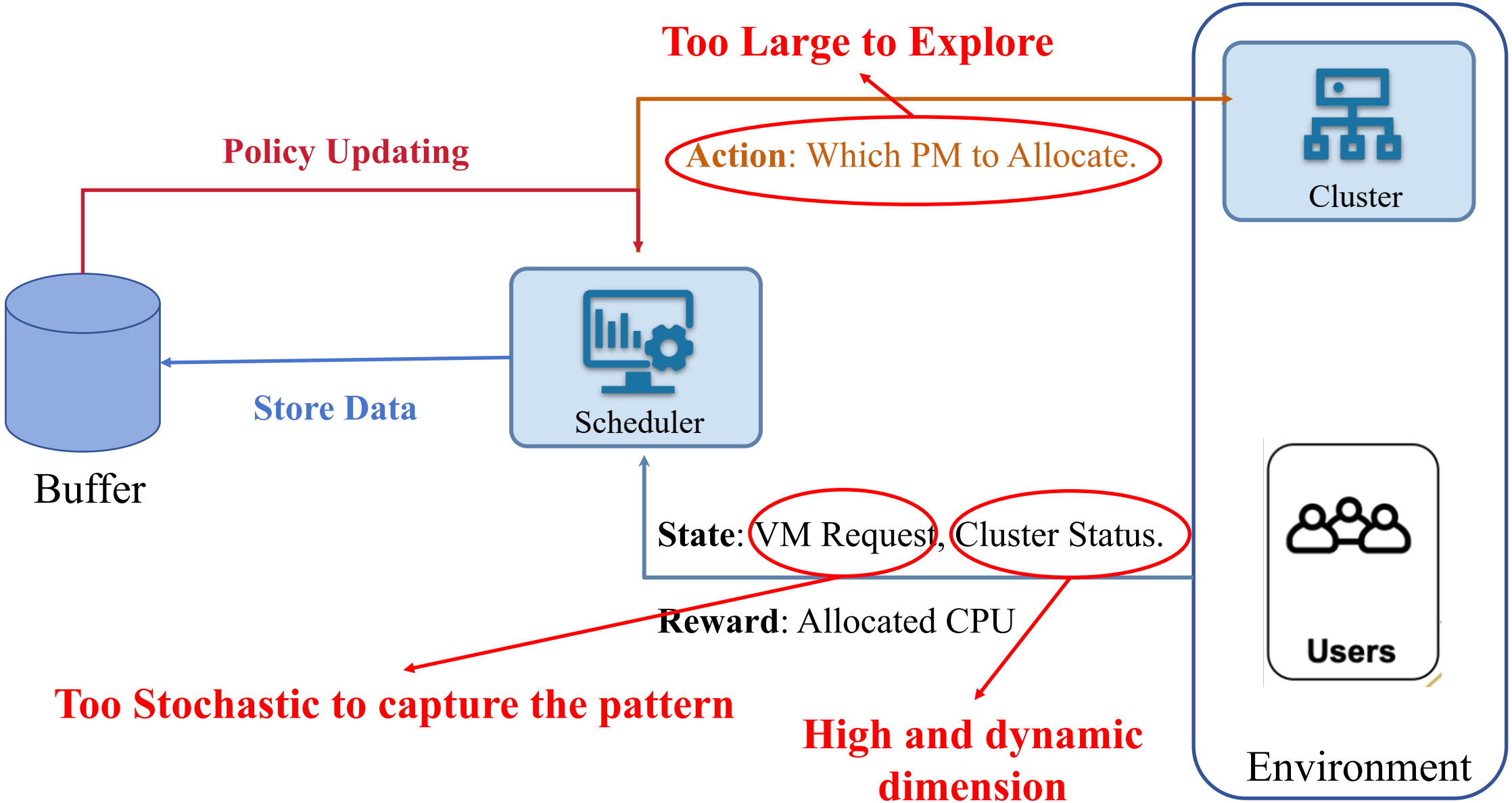


J Sheng, and etc., Learning to Schedule Multi-NUMA Virtual Machines via Reinforcement Learning, ***Pattern Recognition*** 121, 108254, 2022.

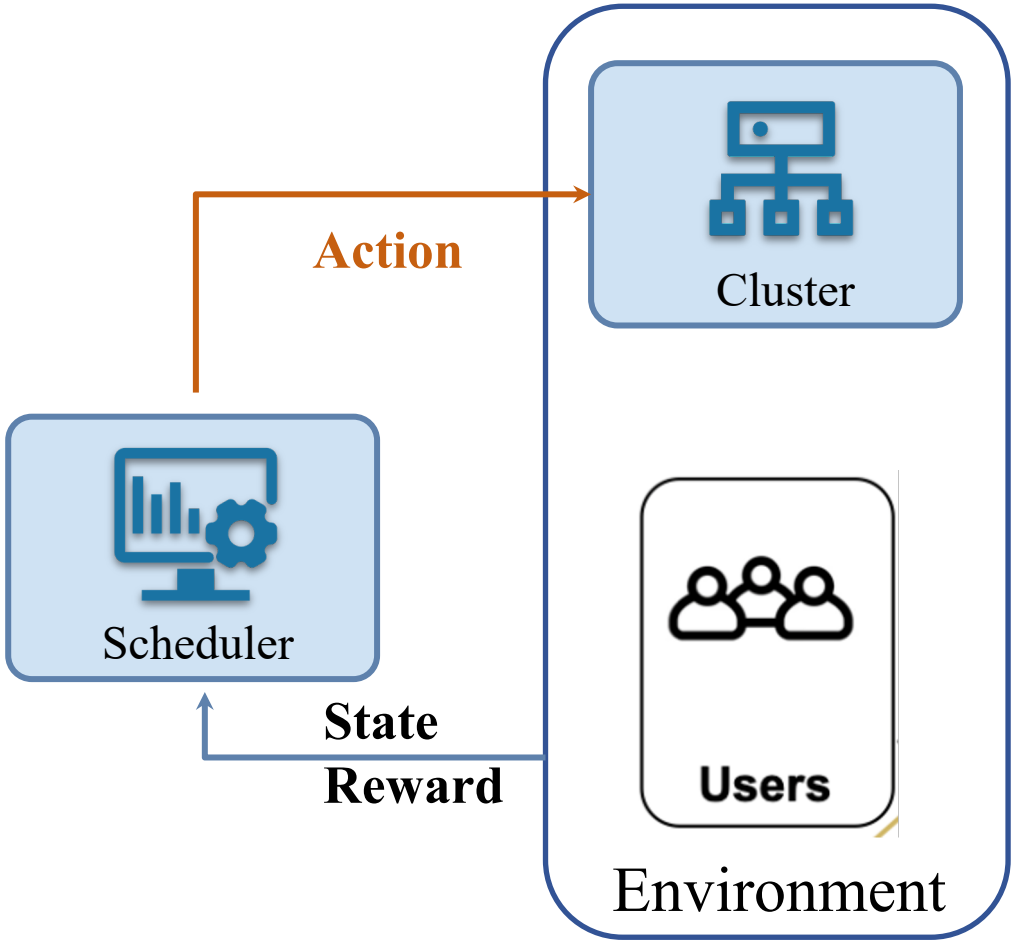
Reinforcement Learning For VM Scheduling



Reinforcement Learning For VM Scheduling



Reinforcement Learning For VM Scheduling



- + Uncertainty Handling?

Hindsight Learning (Sinclair et al. 2023)

- + Efficient Exploration?

SchedRL (Sheng et al. 2020)

- + Scalable Representation?

CVD-RL (Sheng et al. 2024)

- + Explainable Policy?

FunSearch (Romera-Paredes et al. 2023)

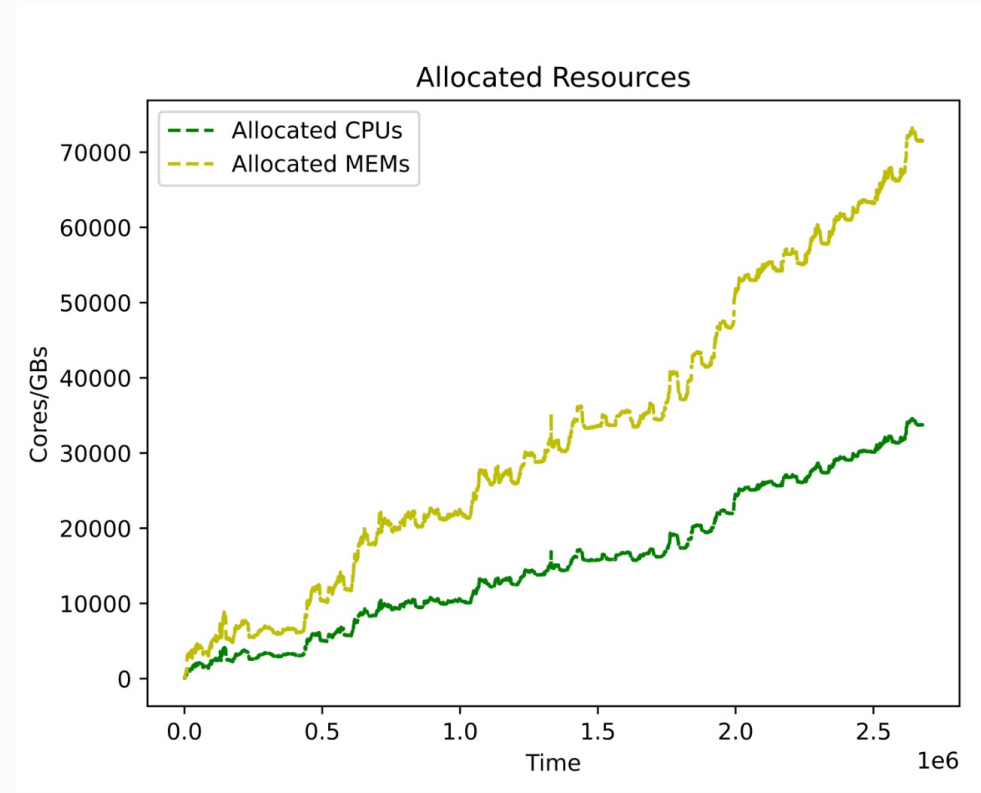
RL For VM Scheduling: Hindsight

Can we explicitly handle the uncertainty?

RL For VM Scheduling: Hindsight

Hindsight → Efficient Exploration → Representation → Explainable Policy

- The stochastic in VM scheduling problem is due to unknown VM request sequence.

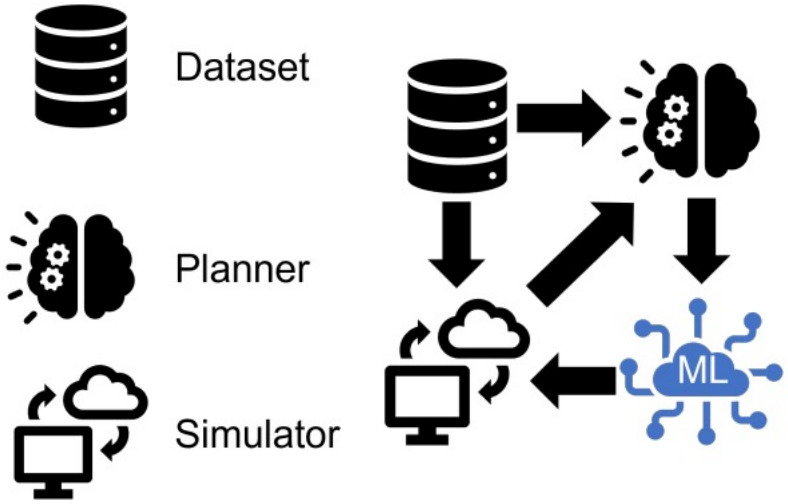


- Sinclair, Sean R., et al. "Hindsight learning for mdps with exogenous inputs." International Conference on Machine Learning. PMLR, 2023.

RL For VM Scheduling: Hindsight

Hindsight → Efficient Exploration → Representation → Explainable Policy

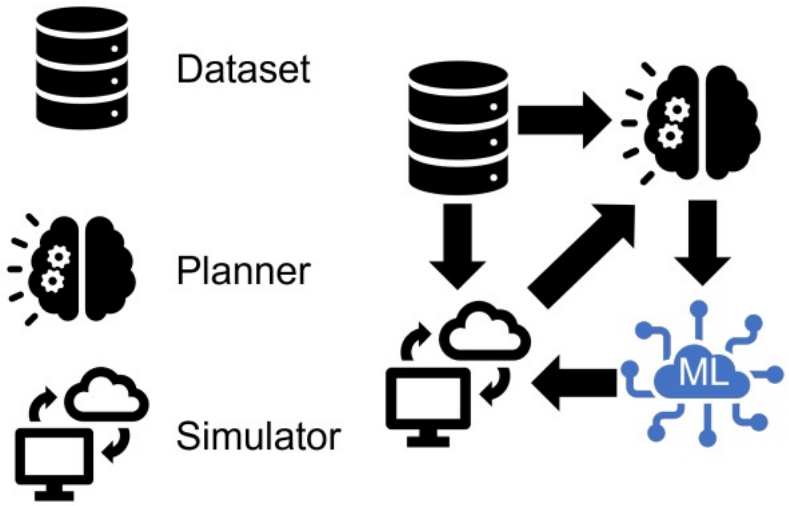
Sample: Sample traces of future request sequence $\xi_{>t}$



- Sinclair, Sean R., et al. "Hindsight learning for mdps with exogenous inputs." International Conference on Machine Learning. PMLR, 2023.

RL For VM Scheduling: Hindsight

Hindsight → Efficient Exploration → Representation → Explainable Policy



Sample: Sample traces of future request sequence $\xi_{>t}$

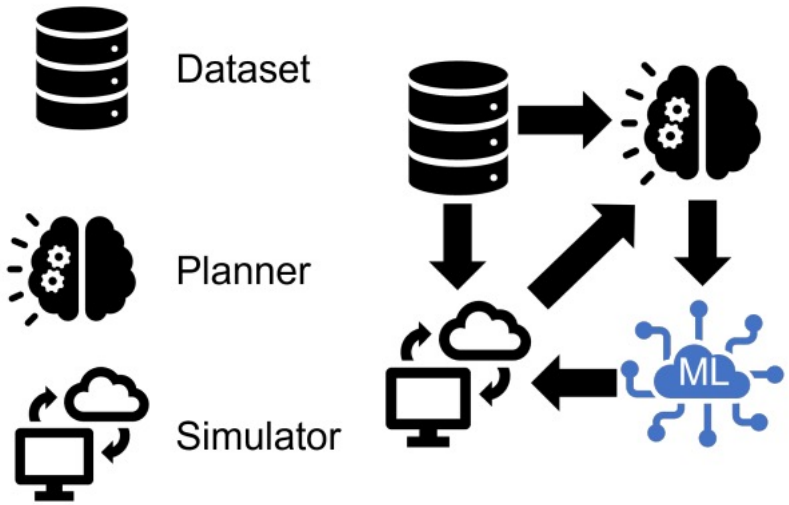
Label: Label data with hindsight heuristic scheduler on traces

$$\hat{Q}(s_t, a_t) = \hat{\mathbb{E}}_{\xi_{>t}}[\max_{a_{t+1},\ldots,a_T} \sum r(s_\tau, a_\tau, \xi_\tau)]$$

- Sinclair, Sean R., et al. "Hindsight learning for mdps with exogenous inputs." International Conference on Machine Learning. PMLR, 2023.

RL For VM Scheduling: Hindsight

Hindsight → Efficient Exploration → Representation → Explainable Policy



Sample: Sample traces of future request sequence $\xi_{>t}$

Label: Label data with hindsight heustic scheduler on traces

$$\hat{Q}(s_t, a_t) = \hat{\mathbb{E}}_{\xi_{>t}}[\max_{a_{t+1},\ldots,a_T} \sum r(s_\tau, a_\tau, \xi_\tau)]$$

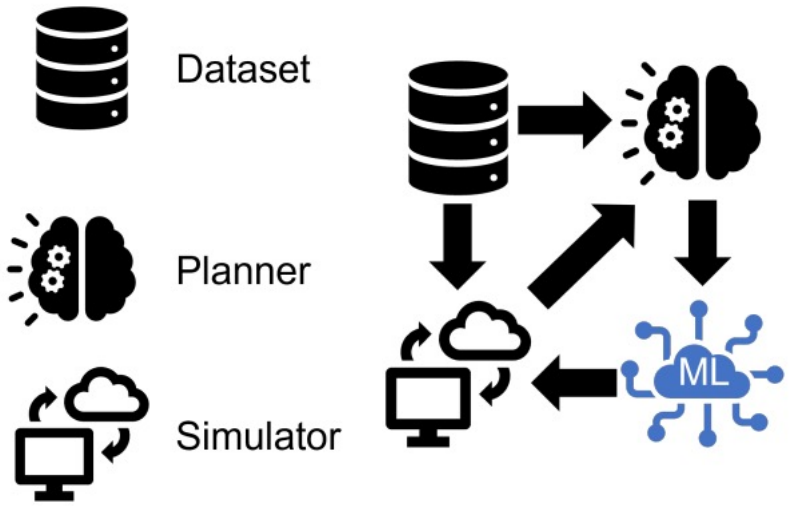
Generalize: Imitation Learning

$$\ell(\pi_\theta) = \mathbb{E}_{\boldsymbol{\xi}}[\sum_{t=1}^{T} \mathbb{E}_{S_t \sim \Pr_t^\pi}[\sum_{a \in \mathcal{A}} (Q_t^\theta(S_t, a) - Q_t^\dagger(S_t, a, \boldsymbol{\xi}_{\geq t}))^2]].$$

- Sinclair, Sean R., et al. "Hindsight learning for mdps with exogenous inputs." International Conference on Machine Learning. PMLR, 2023.

RL For VM Scheduling: Hindsight

Hindsight → Efficient Exploration → Representation → Explainable Policy



Algorithm 3 Hindsight Heuristic.

Input: A cluster state s, sequence of VM requests $\boldsymbol{\xi}_{t:T}$.
Sort requests in descending order of their lifetimes
for Each request ξ **do**
    Allocate to the feasible PM where ξ is the only live VM on it for the least amount of time
end for

- Sinclair, Sean R., et al. "Hindsight learning for mdps with exogenous inputs." International Conference on Machine Learning. PMLR, 2023.

RL For VM Scheduling: Hindsight

Hindsight → Efficient Exploration → Representation → Explainable Policy

Algorithm	Performance $r = -1/\texttt{Packing Density}$	Packing Density
Performance Upper Bound	0.66 ± 0.29	$.09\% \pm 0.03\%$
Best Fit	0.0	0.0
Bin Packing	-64.44 ± 2.49	$-5.34\% \pm 0.2\%$
Round Robin	-56.36 ± 2.65	$-4.67\% \pm 0.22\%$
Random	-48.94 ± 2.33	$-4.08\% \pm 0.19\%$
DQN	-1.00 ± 0.41	$-0.05\% \pm 0.04\%$
MAC	-0.38 ± 0.033	$-0.03\% \pm 0.00\%$
AC	-2.94 ± 0.61	$-0.21\% \pm 0.06\%$
Policy Gradient	-1.03 ± 0.39	$-0.06\% \pm 0.04\%$
Hindsight MAC	$\mathbf{0.18 \pm 0.093}$	$\mathbf{0.05\% \pm 0.00\%}$
HINDSIGHT Q-DISTILLATION	0.08 ± 0.32	$0.04\% \pm 0.029\%$

- Sinclair, Sean R., et al. "Hindsight learning for mdps with exogenous inputs." International Conference on Machine Learning. PMLR, 2023.

RL For VM Scheduling: Exploration

Can we improve exploration efficiency?

RL For VM Scheduling: Exploration

Hindsight → **Efficient Exploration** → Representation → Explainable Policy

Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	1	FIRE	2	UP
3	RIGHT	4	LEFT	5	DOWN
6	UPRIGHT	7	UPLEFT	8	DOWNRIGHT
9	DOWNLEFT	10	UPFIRE	11	RIGHTFIRE
12	LEFTFIRE	13	DOWNFIRE	14	UPRIGHTFIRE
15	UPLEFTFIRE	16	DOWNRIGHTFIRE	17	DOWNLEFTFIRE

Atari Action Space

* 0-N where N is the num of PM.
* It can be from 5 to 1000.

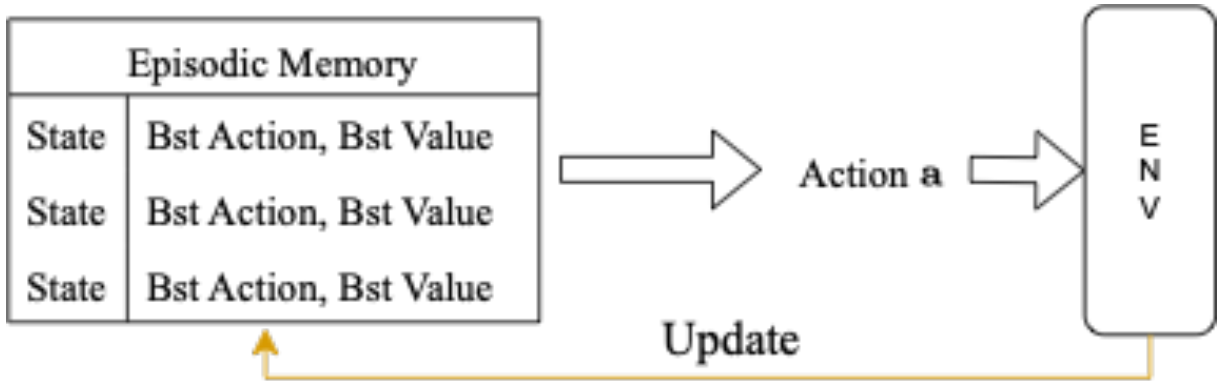
VM Scheduling Action Space

- Junjie, Sheng, et al "Learning to schedule multi-NUMA virtual machines via reinforcement learning." *Pattern Recognition*, 2022

RL For VM Scheduling: Exploration

Hindsight → **Efficient Exploration** → Representation → Explainable Policy

1. **Episodic Sampling**
   Gradient based updating is slow which hinders the exploration effectiveness-->
   add fast thinking (episodic sampling)

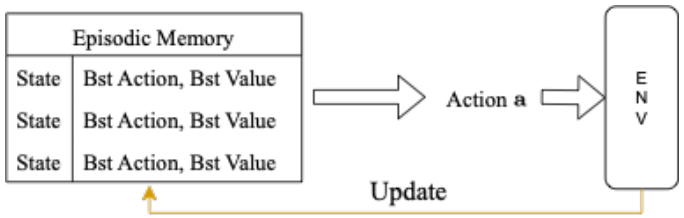


- Junjie, Sheng, et al "Learning to schedule multi-NUMA virtual machines via reinforcement learning." *Pattern Recognition*, 2022

RL For VM Scheduling: Exploration

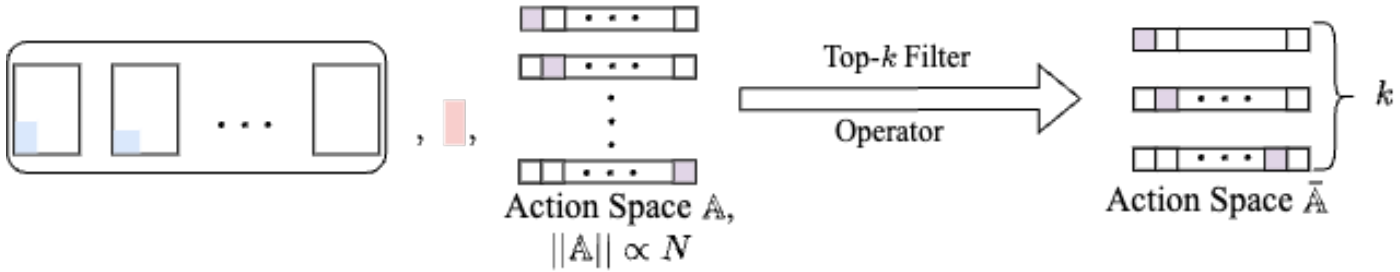
Hindsight → **Efficient Exploration** → Representation → Explainable Policy

1. Episodic Sampling



2. Top-K Filter

$$\mathcal{K}(f, s) = \{x | x \in \mathcal{A} \text{ and } \mathrm{f(x, s) \text{ is among the Top } } K \text{ scores in } \mathcal{A}\}$$



- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Representation

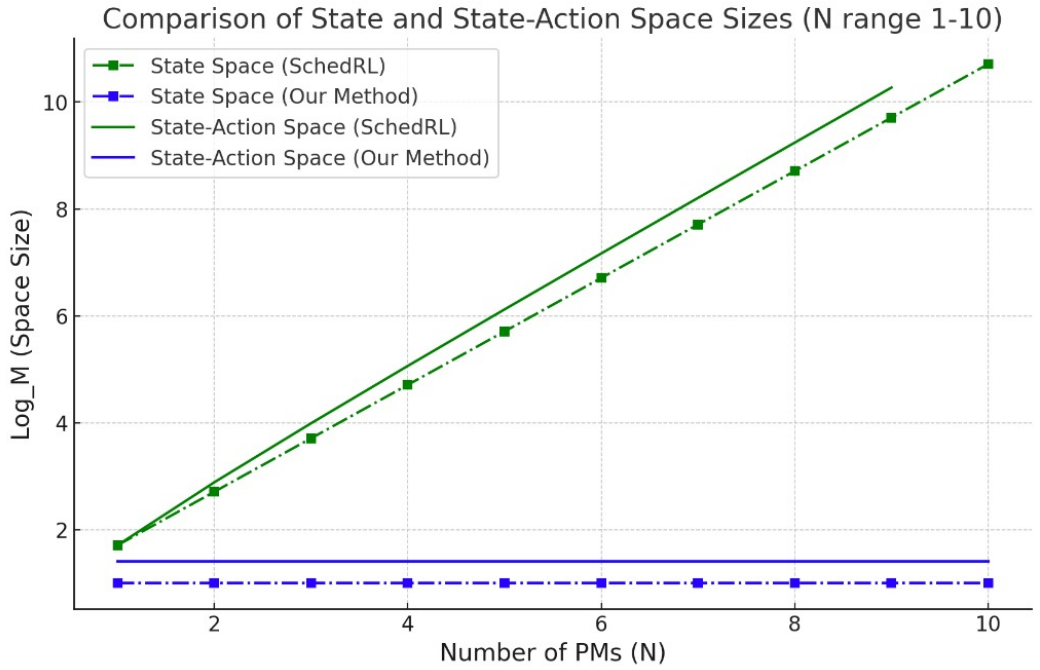
Can we represent the value function effectively?

RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

State Space: LM^N, where L is the number of VM types, M is the number of PM status types, and N is the number of PMs.

----> Representation Challenge: curse of dimensionality



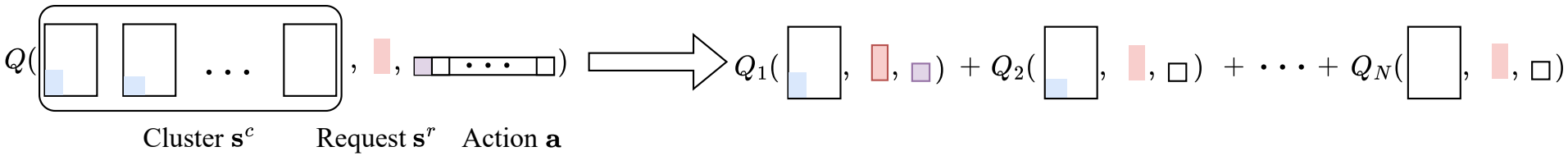
RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

Asssumption: *the overall performance of the cluster can be viewed as the sum of performances of individual PMs in the cluster*.

1. **Decoposition Operator:**

$$Q\left(s_t^c, s_t^v, a; \theta\right) = \sum_i Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta_i\right),$$



- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

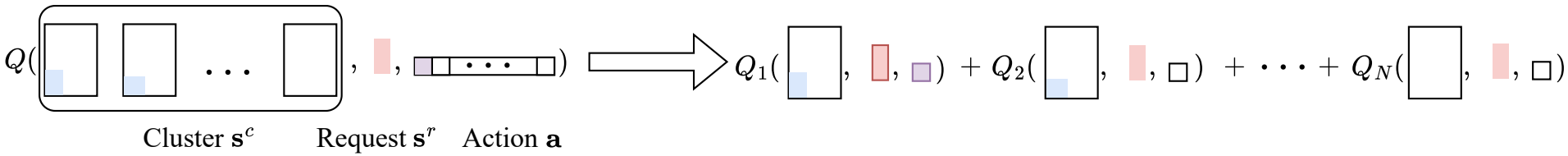
RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

Asssumption: *the overall performance of the cluster can be viewed as the sum of performances of individual PMs in the cluster*.

1. **Decoposition Operator:**

$$Q\left(s_t^c, s_t^v, a; \theta\right) = \sum_i Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta_i\right),$$



Bestfit is in the representation space.

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

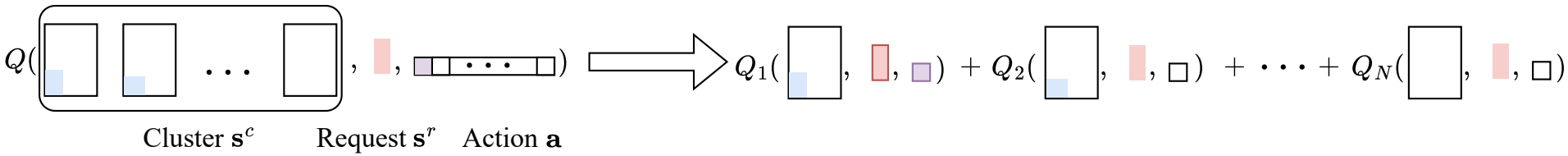
RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

Asssumption: *the overall performance of the cluster can be viewed as the sum of performances of individual PMs in the cluster*.

1. **Decoposition Operator:**

$$Q\left(s_t^c, s_t^v, a; \theta\right) = \sum_i Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta_i\right),$$



Bestfit is in the representation space.

Further due to homogenous feature of each PM, we share parameters among them and obtain state space as: LM .

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

1. **Decoposition Operator:**

$$Q\left(s_t^c, s_t^v, a; \theta\right) = \sum_i Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta_i\right),$$

State Space: LM^N → LM.

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

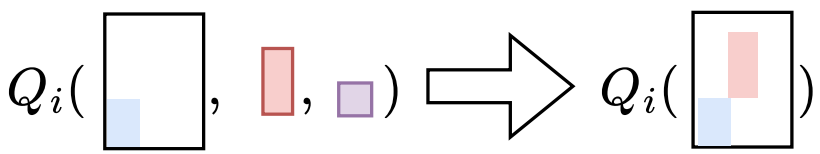
1. **Decoposition Operator:**

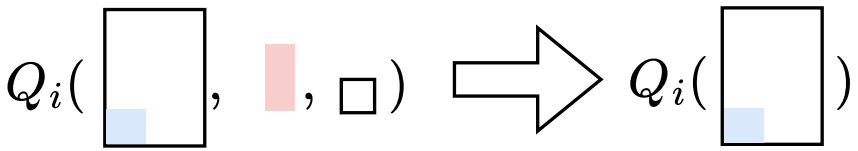
$$Q\left(s_t^c, s_t^v, a; \theta\right) = \sum_i Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta_i\right),$$

State Space: LM^N → LM.

2. **Look Ahead Operator**

$$Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta\right) = Q_i\left(s_{t,i}^c - s_{t,a_i-1}^v; \theta\right), \text{ if } a_i \neq 0.$$





- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

1. **Decoposition Operator:**

$$Q\left(s_t^c, s_t^v, a; \theta\right) = \sum_i Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta_i\right),$$

LM^N → LM.

2. **Look Ahead Operator**

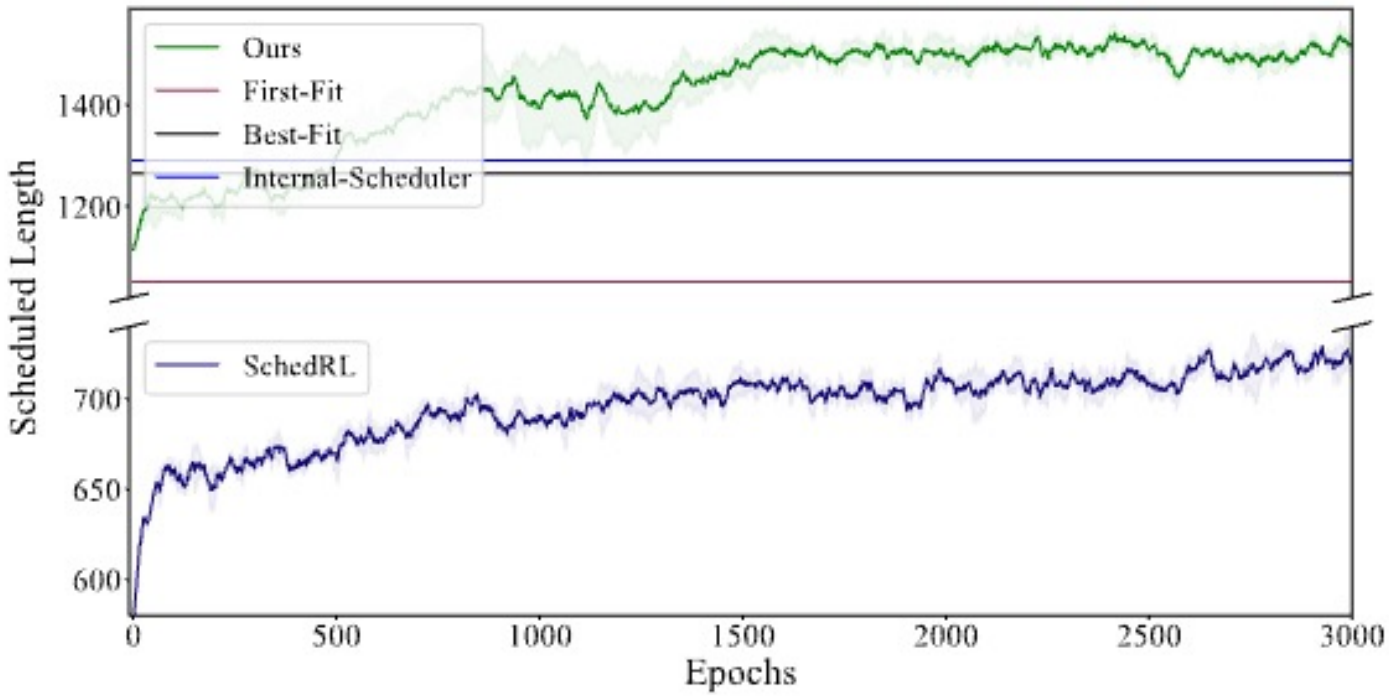
$$Q_i\left(s_{t,i}^c, s_t^v, a_i; \theta\right) = Q_i\left(s_{t,i}^c - s_{t,a_i-1}^v; \theta\right), \text{ if } a_i \neq 0.$$

LM^N → LM → M.

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Exploration

Hindsight → **Efficient Exploration** → **Representation** → Explainable Policy



- Training in a cluster with 50 PMs, our method outperforms baselines.

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

Scenario	Server Num	Ratio	Best-Fit			Internal			CVD-RL		
			Length	Income	cpuAll	Length	Income	cpuAllo	Length	Income	cpuAllo
Non-Expansion	50	0%	1386.8	7880.5	41.7	1398.6	8191.5	41.6	**1464.6** (±23.8)	**8680.2** (±119.7)	**43.2** (±0.4)
		30%	1149.7	7157.4	50.4	1144.2	7751.7	50.6	**1224.3** (±37.5)	**7883.6** (±317.8)	**51.4** (±0.2)
		40%	1010.1	6244.2	53.9	1004.5	6428.6	53.8	**1061.9** (±8.5)	**6789.3** (±267.8)	**54.5** (±0.3)
		50%	852.8	5295.4	57.4	844.0	5112.9	57.5	**939.3** (±23.8)	**6273.4** (±5.0)	**58.4** (±0.2)
		60%	669.2	3947.8	63.2	661.9	4391.4	63.3	**712.4** (±3.2)	**4441.9** (±191.9)	**63.7** (±0.0)
Non-Expansion	100	0%	3812.0	33110.3	45.0	3824.6	31975.9	45.0	**3855.4** (±45.4)	**32912.5** (±1090.9)	**45.2** (±0.5)
		30%	2965.9	27289.1	53.5	3003.8	**26935.9**	54.0	**3019.1** (±34.1)	26649.6 (±161.0)	**54.1** (±0.2)
		40%	2565.1	**22768.9**	57.9	**2604.0**	22448.6	**58.5**	2588.1 (±10.2)	22290.1 (±128.5)	58.3 (±0.1)
		50%	2078.4	17456.7	63.1	2106.3	17296.8	63.5	**2181.8** (±9.1)	**19108.4** (±75.3)	**63.9** (±0.1)
		60%	1502.9	11009.4	67.9	1589.7	12258.0	68.5	**1602.1** (±5.7)	**12417.8** (±24.5)	**68.6** (±0.1)

Generalization Ability:

- In 50 PMs setting, our method outperforms baselines under different start ratio.

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Representation

Hindsight → Efficient Exploration → **Representation** → Explainable Policy

Scenario	Server Num	Ratio	Best-Fit			Internal			CVD-RL		
			Length	Income	cpuAll	Length	Income	cpuAllo	Length	Income	cpuAllo
Non-Expansion	50	0%	1386.8	7880.5	41.7	1398.6	8191.5	41.6	**1464.6** (±23.8)	**8680.2** (±119.7)	**43.2** (±0.4)
		30%	1149.7	7157.4	50.4	1144.2	7751.7	50.6	**1224.3** (±37.5)	**7883.6** (±317.8)	**51.4** (±0.2)
		40%	1010.1	6244.2	53.9	1004.5	6428.6	53.8	**1061.9** (±8.5)	**6789.3** (±267.8)	**54.5** (±0.3)
		50%	852.8	5295.4	57.4	844.0	5112.9	57.5	**939.3** (±23.8)	**6273.4** (±5.0)	**58.4** (±0.2)
		60%	669.2	3947.8	63.2	661.9	4391.4	63.3	**712.4** (±3.2)	**4441.9** (±191.9)	**63.7** (±0.0)
Non-Expansion	100	0%	3812.0	33110.3	45.0	3824.6	31975.9	45.0	**3855.4** (±45.4)	**32912.5** (±1090.9)	**45.2** (±0.5)
		30%	2965.9	27289.1	53.5	3003.8	**26935.9**	54.0	**3019.1** (±34.1)	26649.6 (±161.0)	**54.1** (±0.2)
		40%	2565.1	**22768.9**	57.9	**2604.0**	22448.6	**58.5**	2588.1 (±10.2)	22290.1 (±128.5)	58.3 (±0.1)
		50%	2078.4	17456.7	63.1	2106.3	17296.8	63.5	**2181.8** (±9.1)	**19108.4** (±75.3)	**63.9** (±0.1)
		60%	1502.9	11009.4	67.9	1589.7	12258.0	68.5	**1602.1** (±5.7)	**12417.8** (±24.5)	**68.6** (±0.1)

Generalization Ability:

- In 50 PMs setting, our method outperforms baselines under different start ratio.
- When directly apply the learnt policy to 100 PMs setting, it still outperforms baselines, showing the generalization ability to different num of PMs.

- Junjie, Sheng, et al. Scalable Reinforcement Learning for Virtual Machine Scheduling

RL For VM Scheduling: Explainable policy

Can we search in the explainable function space?

RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**

1. **Code Generation:**
   Code is an explainable representation of a policy.

```
def priority(bin, item):
    score = - (bin - item[0])
    return score
```

BestFit

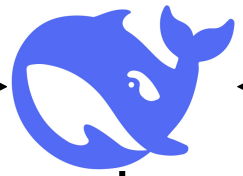
RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**

1. Code Generation:

```
def priority(bin, item):
    score = - (bin[0] - item[0])
    return score
```

BestFit



Given the existing priority_v0 function, please generate an optimized version named priority_v*. This new version should be more complex and efficient, incorporating multiple conditional logic and loops as necessary. The function should calculate priorities for items to be added to bins, considering the item size and bin capacities.

Prompt

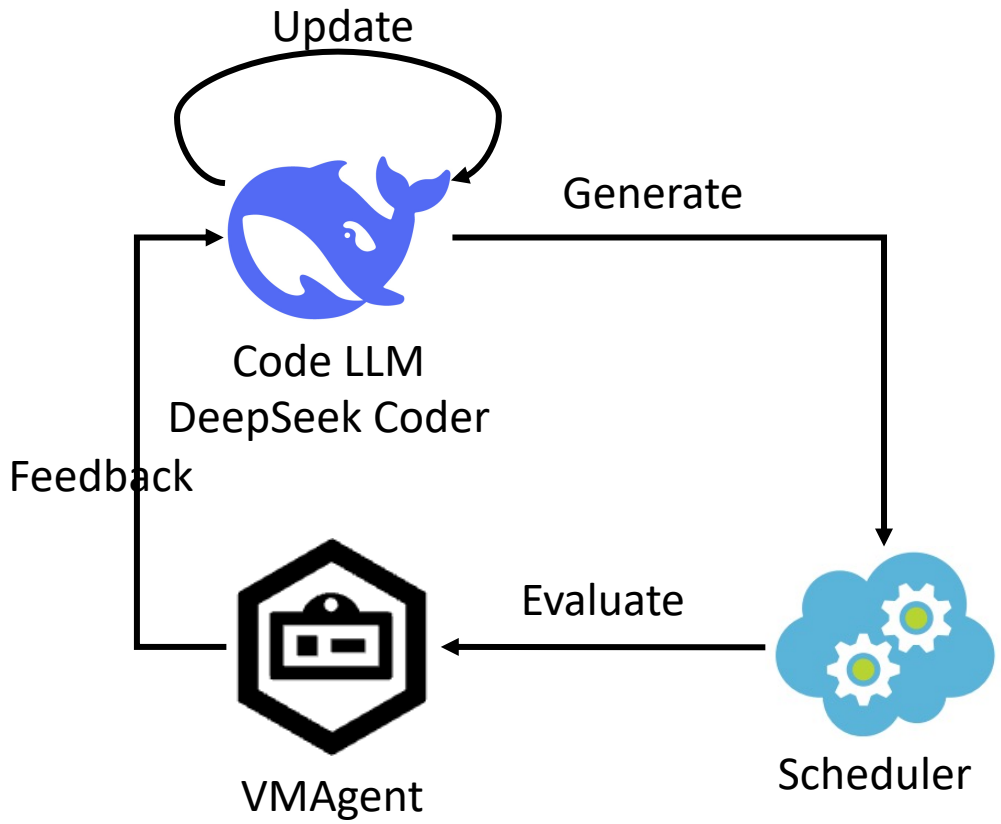
```
def priority(bin, item):
    score = 0
    for i in range(len(bin)):
        diff = bin[i] - item[i]
        score += diff if diff > 0  else bin[i] + item[i]
        score *= 2 if diff < -5 else 1
        score += np.sin(score) if score > 100 else 0
    ...
    return score
```

BestFit*

RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**

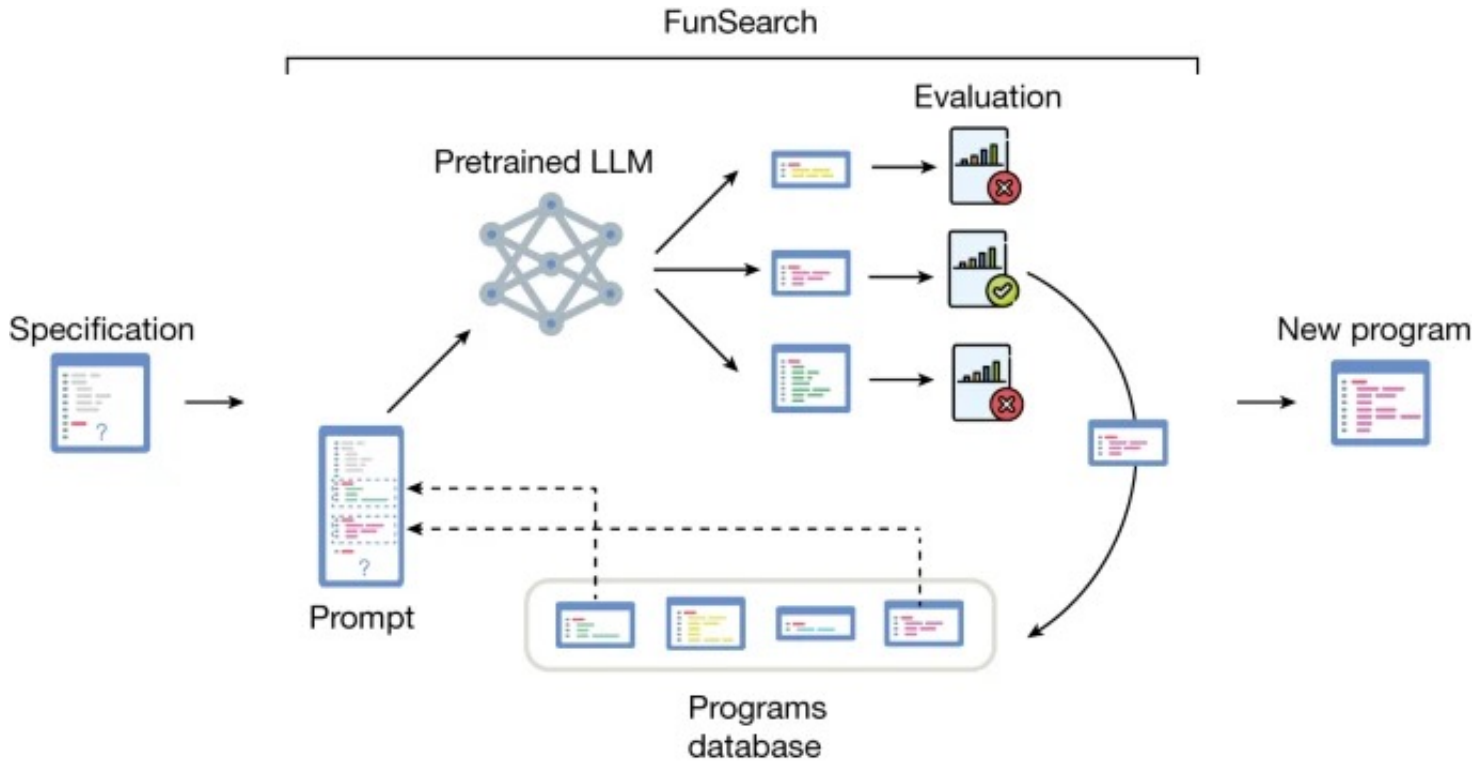
1. **Code Generation**
2. **Evolve with feedback:**



RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**

1. **Code Generation**
2. **Evolve with feedback:**



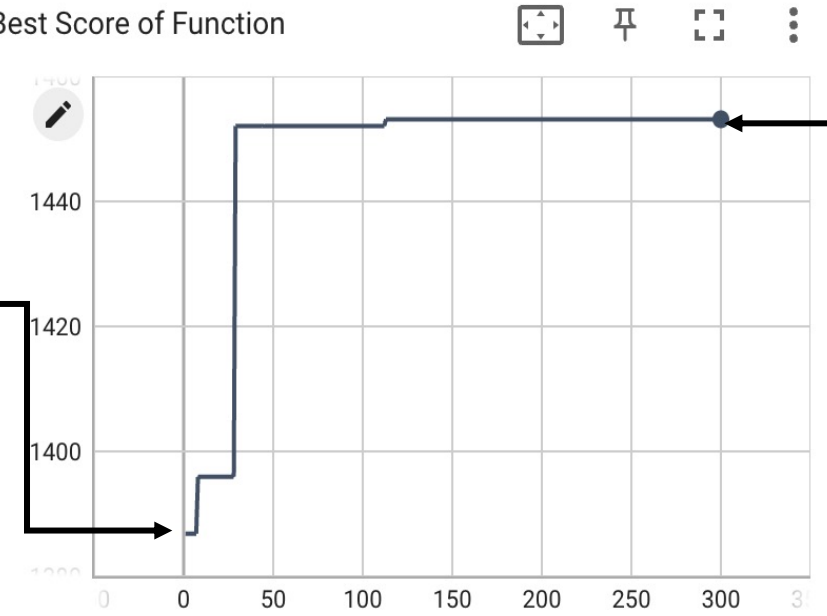
RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**

```
def priority(bin, item):
    """
    Returns priority with which we want to add item to each bin.
    Args:
        bin (int): Total available CPU resources.
        item (int): Item need to be placed.

    Returns:
        int: The total score for current bin.

    """
    score = - (bin[0] - item[0])
    return score
```




```
def priority(bin, item):
    score = 0
    for i in range(len(bin)):
        diff = bin[i] - item[i]
        score += diff if diff > 0 else bin[i] + item[i]
        score *= 2 if diff < -5 else 1
        score += np.sin(score) if score > 100 else 0
    score = -abs(score + 10)
    score += 5 if score % 2 == 0 else -5
    score += sum(range(100))
    score += item[-1]
    return score
```

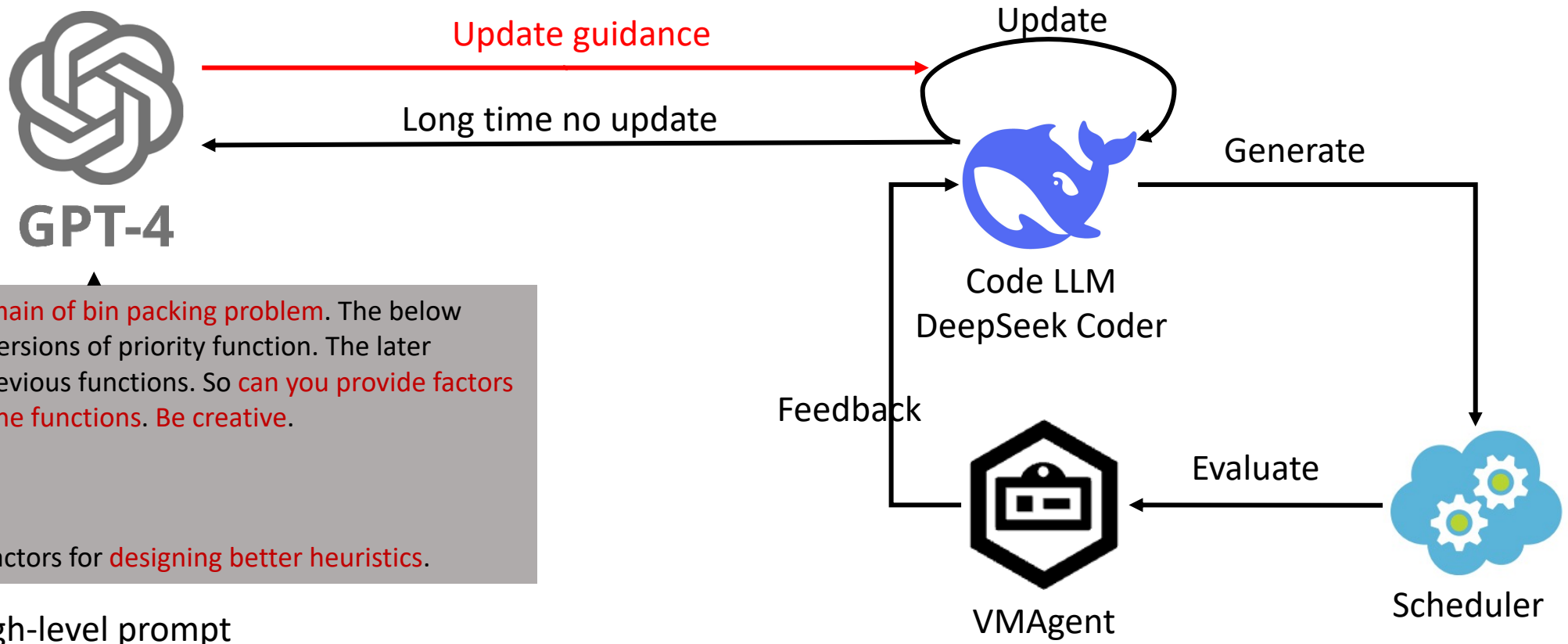
Advantage
- Efficiency
- Emergence
- Generalization
- Explainable

Scenarios	Size	BestFit	BestFit*	Training
Testing	30	566.9	**572.1333**	
Training	50	1386.8	**1453.133**	**A100 80G**
Testing	100	3811.5	**3823.633**	**300 epochs**
Testing	150	6256.266	**6306.133**	**7~8 hours**
Testing	200	8366.466	**8447.9**	

RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**

1. **Code Generation**
2. **Evolve with feedback**
3. **Guide to Evolve**



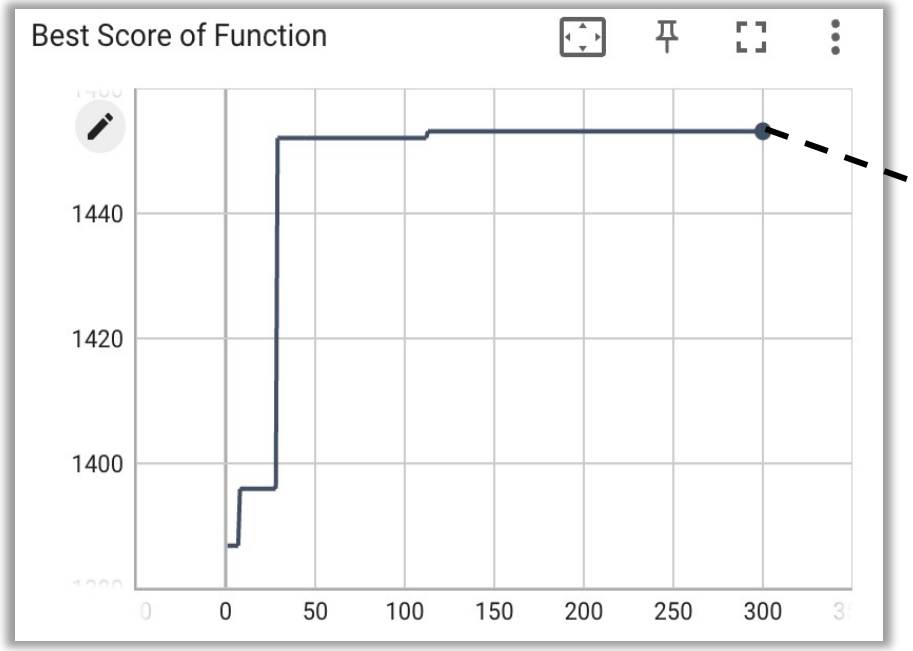
You are an expert in the domain of bin packing problem. The below functions are the iteration versions of priority function. The later functions are better than previous functions. So can you provide factors which may help improving the functions. Be creative.
priority_1()
priority_2()

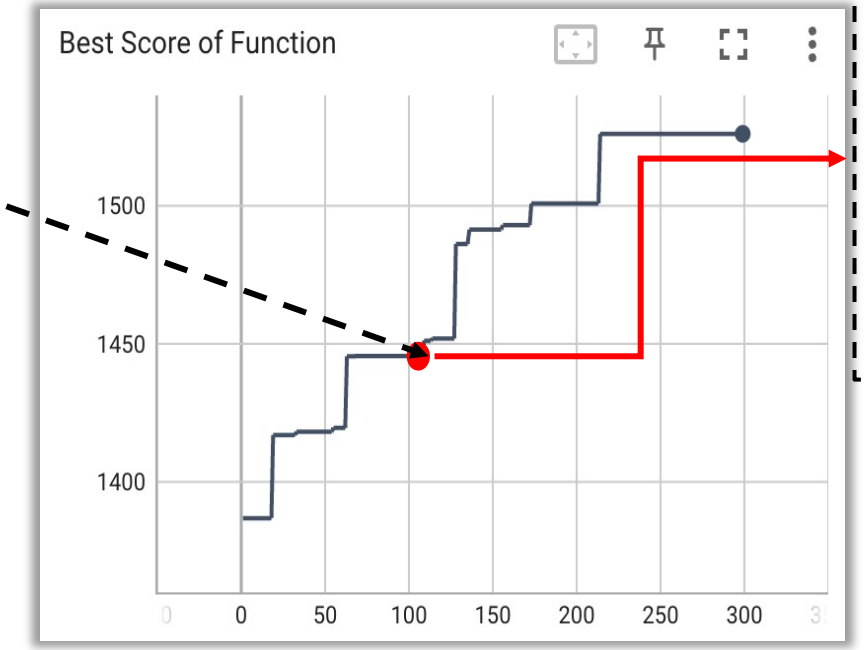
… Write constructive code factors for designing better heuristics.

High-level prompt

RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**





1. Normalize score

```
for score in scores:
    score /= max_possible_score
```

2. Exponential Bonus:

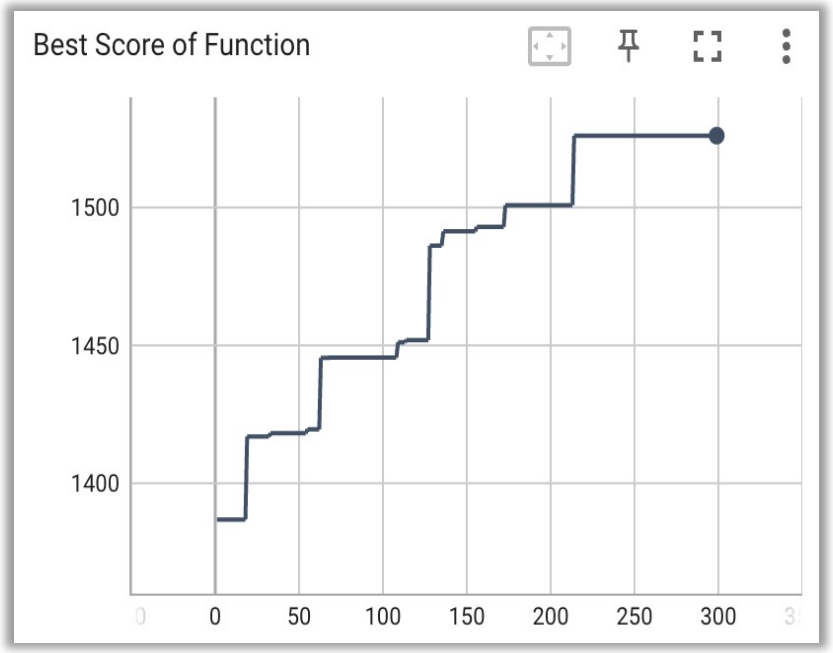
```
for score in scores:
    scores.append(np.exp(score))
total_score = sum(scores)
```

Update guidance

Size	BestFit	BestFit*	BestFit**	Traing
30	566.9	572.1333	**572.6333**	
50	1386.8	1453.133	**1526.066**	**A100 80G**
100	3811.5	3823.633	**3874.966**	**300 epochs**
150	6256.266	6306.133	**6318.166**	**7~8 hours**
200	8366.466	**8447.9**	8403.366	

RL For VM Scheduling: Explainable policy

Hindsight → Efficient Exploration → Representation → **Explainable Policy**



Empty Bin Bonus: If the bin is empty, add a predefined bonus:

$$\text{score}_{\text{empty}} = \text{EMPTY\_BIN\_BONUS} \times (\text{bin} == 0)$$

Proportion Bonus: Add a bonus if the proportion of item to bin sum is below a threshold:

$$\text{score}_{\text{proportion}} = \text{PROPORTION\_BONUS} \times (\frac{\sum \text{item}}{\sum \text{bin}} < \text{THRESHOLD\_1})$$

Proximity Bonus: Add a bonus if the bin is close to reaching maximum capacity:

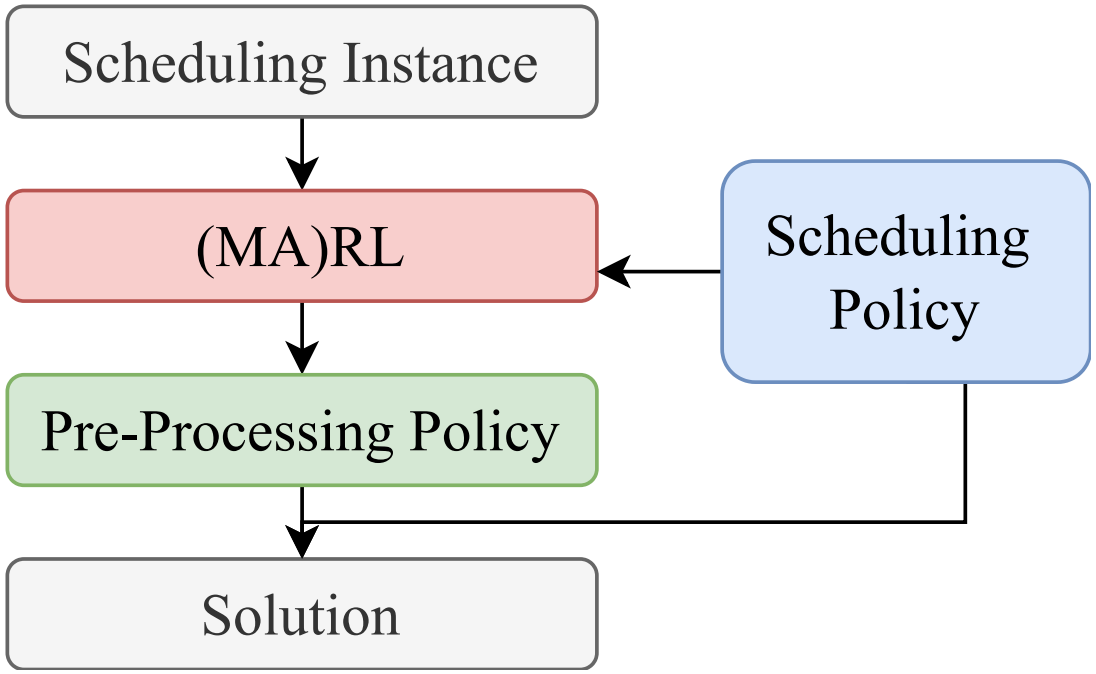
$$\text{score}_{\text{proximity}} = \text{PROXIMITY\_BONUS} \times (|\sum \text{bin} - \text{MAX\_BIN\_CAPAC}$$

Half of Bonus: Add a bonus if any bin is less than half full:

$$\text{score}_{\text{half}} = \text{HALF\_OF\_BONUS} \times (\text{any}(\text{bin} < \frac{\text{MAX\_BIN\_CAPACITY}}{2}))$$

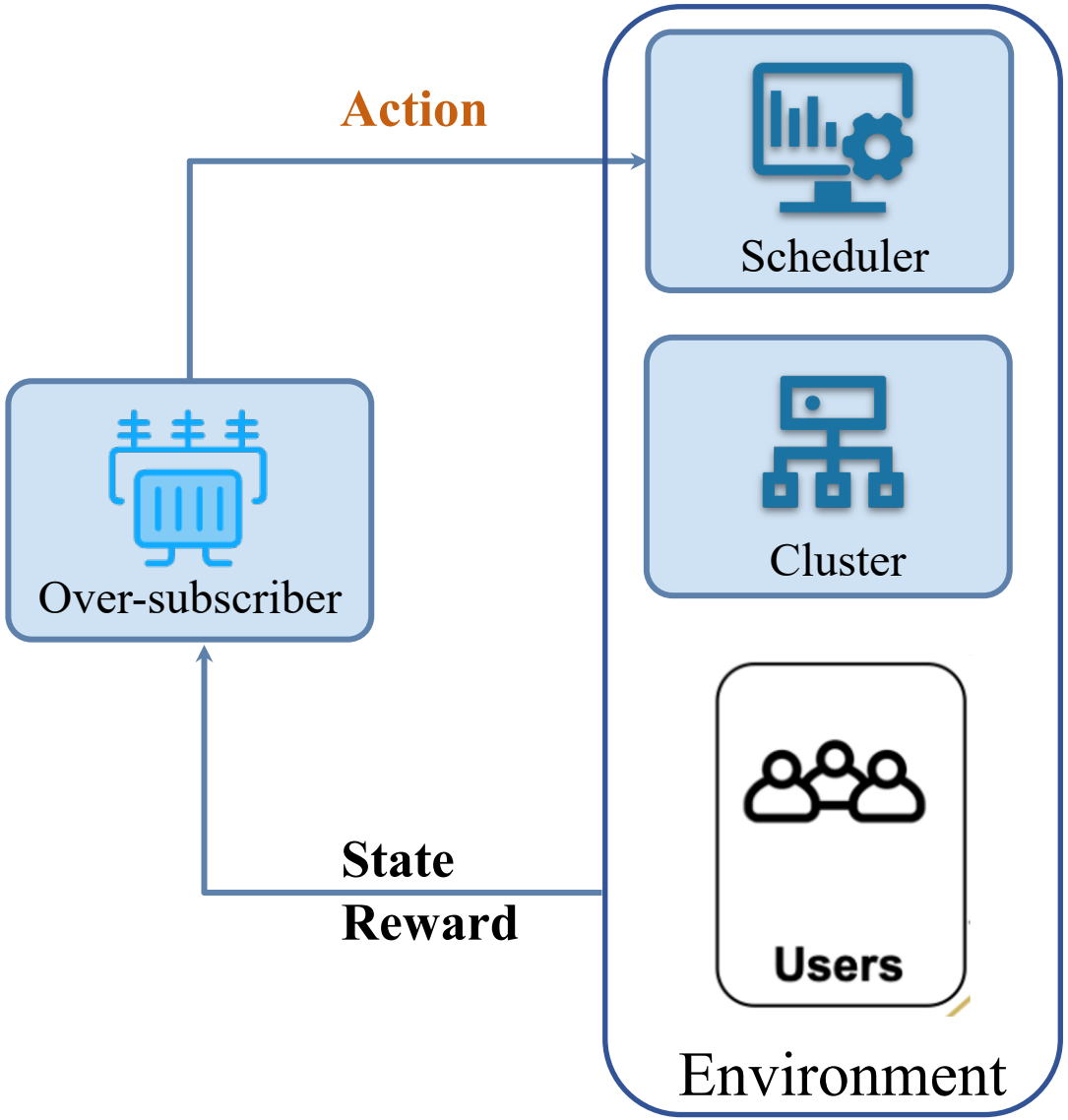
The final priority function that considers multiple factors

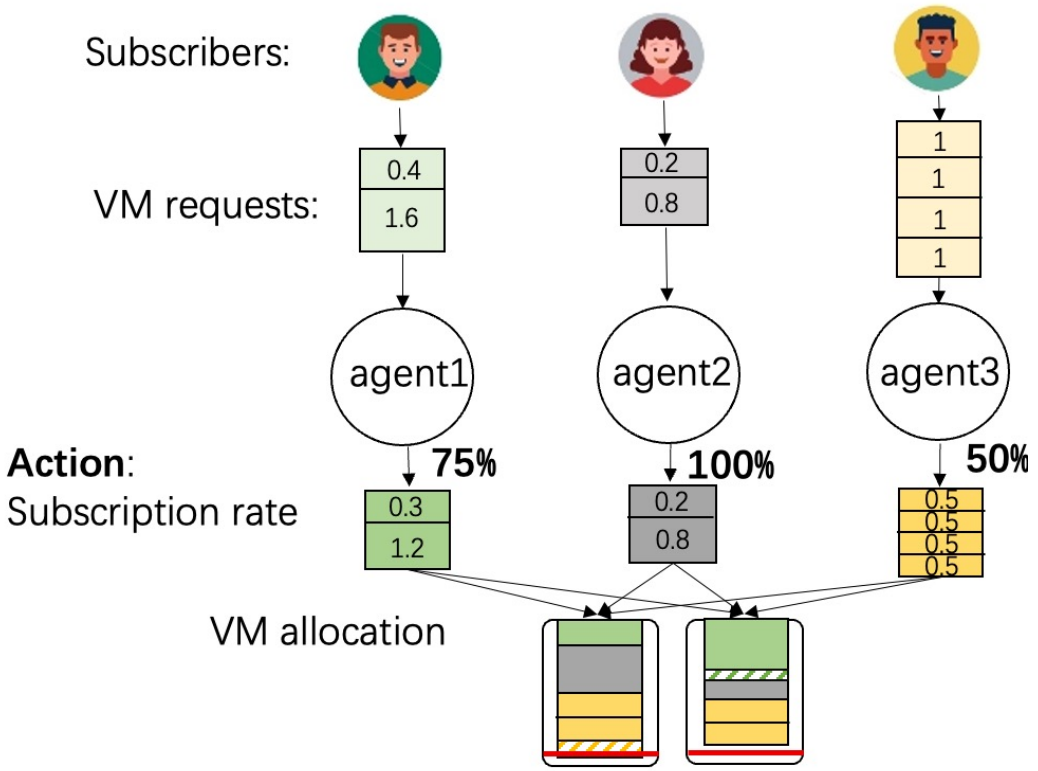
RL Methods For Cloud Resource Scheduling



RL For Indirect VM Scheduling

RL For Indirect VM Scheduling: oversubscription

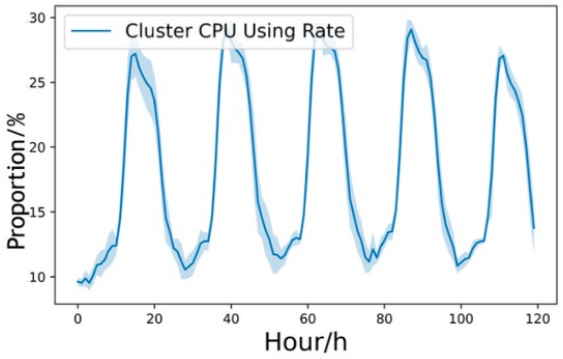




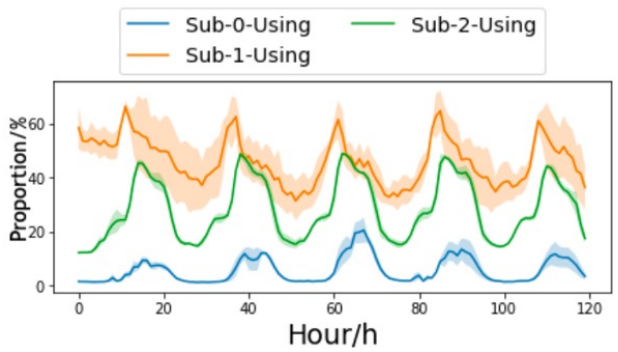
RL For Indirect VM Scheduling: oversubscription

- Cloud Computing: Subscribers estimate their resource usages and request a number of Virtual Machines (VMs) accordingly.
- Subscribers usually over-estimate their usages and the cloud tends to have low utilization.

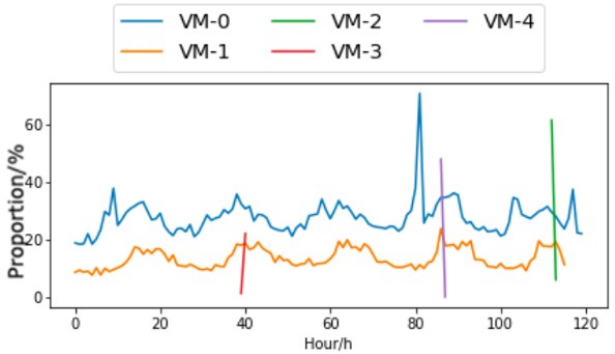
Oversubscription is an effective way to improve resource utilization.



(a) CPU Usage in Cluster Level



(b) CPU Usage in Subscriber Level

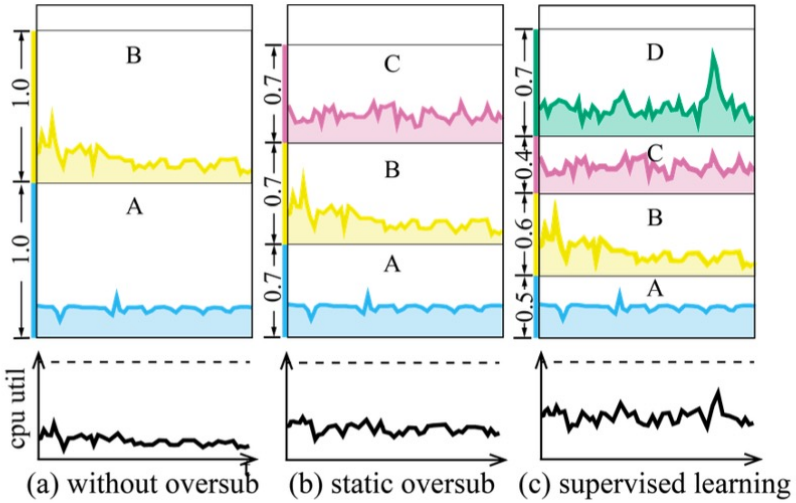


(c) CPU Usage in VM Level

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription

- Oversubscription: For each VM on a Physical Machine (PM), oversubscription assigns less than the requested resources to the VM.
- Goal: reduces the most unused resources while having a low probability of causing hot machines.



- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription

- The number of chance constraints increases linearly with the number of PMs,
- The transition function is not available to the agents,
- It is a non-convex optimization problem.

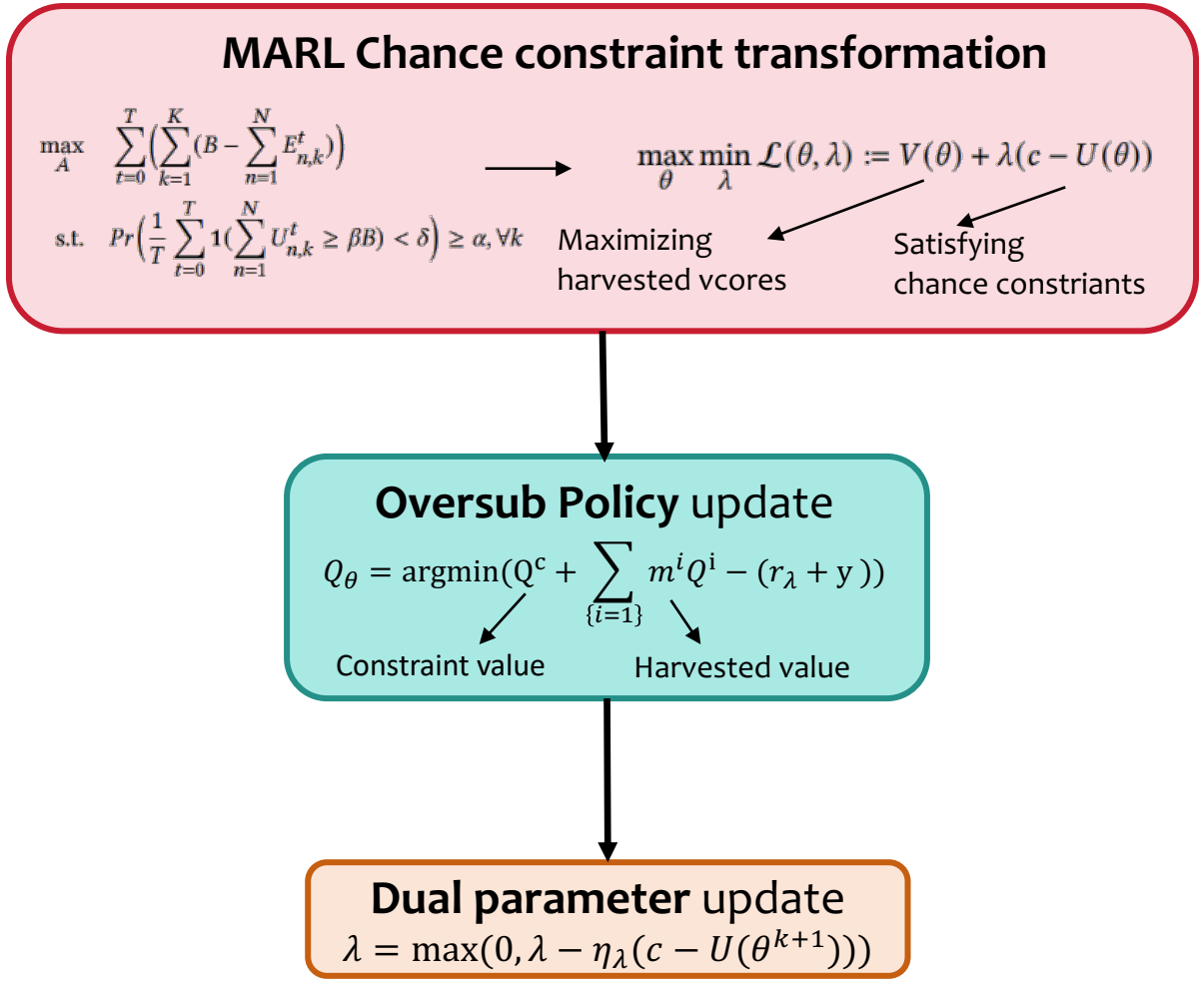
$$\max_{\pi} \quad J(\pi) = \mathbb{E}_{s_0}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$$

$$\text{s.t.} \quad \Pr\left(\frac{1}{T}\sum_{t=1}^{T} C_k(s_t) < \delta | \pi\right) \geq \alpha, \quad \forall k$$

$$s_{t+1} = p(s_t, a_t);\ a_t^i = \pi^i(\cdot|o_t^i);\ o_t = O(s_t),\ \forall t, i$$

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription



- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription

- We transform the chance constrained to a cumulative constraint through inverse Markovian

$$\begin{aligned} \max_{\theta} \quad & V(\theta) := \mathbb{E}[\sum_{t=0}^{T} r(s_t, a_t)|\pi_\theta] \\ \text{s.t.} \quad & U(\theta) < c, U(\theta) := \frac{1}{T}\sum_{t=0}^{T} \Pr(C_c(s_t)|\pi_\theta) \end{aligned} \tag{5}$$

Intuitively, when c is small enough, the feasible policy in Eq. (5) will avoid hot cluster and further be feasible in Eq. (3). When set c as $(1-\alpha)\delta$, the feasible policy in Eq. (5) achieves $\alpha - \delta$ cluster safety.

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription

- Then we reformulate it as a primal-dual problem.

$$\max_{\theta} \quad V(\theta) := \mathbb{E}[\sum_{t=0}^{T} r(s_t, a_t)|\pi_\theta]$$

$$\text{s.t.} \quad U(\theta) < c, U(\theta) := \frac{1}{T}\sum^{T} \Pr(C_c(s_t)|\pi_\theta)$$

$$\max_{\theta} \min_{\lambda} \mathcal{L}(\theta, \lambda) := V(\theta) + \lambda(c - U(\theta)).$$

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription

- Primal:
  - Encode to the reward.

$$\max_{\theta} \quad \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_\lambda(s_t, a_t)]$$
$$\text{s.t.} \quad r_\lambda(s, a) = r(s, a) + \lambda(c - C_c(s)))$$

  - Value Decomposition Network:

$$Q(s_t, a_t; \theta) = Q^c(s_t^c; \theta^c) + \sum_{i}^{N} m_t^i Q^i(o_t^i; \theta^i),$$

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription

- Primal:
  - Loss Function.

$$L = \arg\min_{\theta} ||Q^c(s_t^c;\theta^c) + \sum_{i=1}^{N} m_t^i Q^i(o_t^i;\theta^i) - (r_\lambda + y)||_2$$

$$\text{s.t.} \quad y = Q^c(s_{t+1}^c;\bar{\theta}^c) + \sum_{i=1}^{N} Q^i(\max_a Q^i(a, o_{t+1}^i;\theta^i), o_{t+1};\bar{\theta}^i)$$

- Dual update:

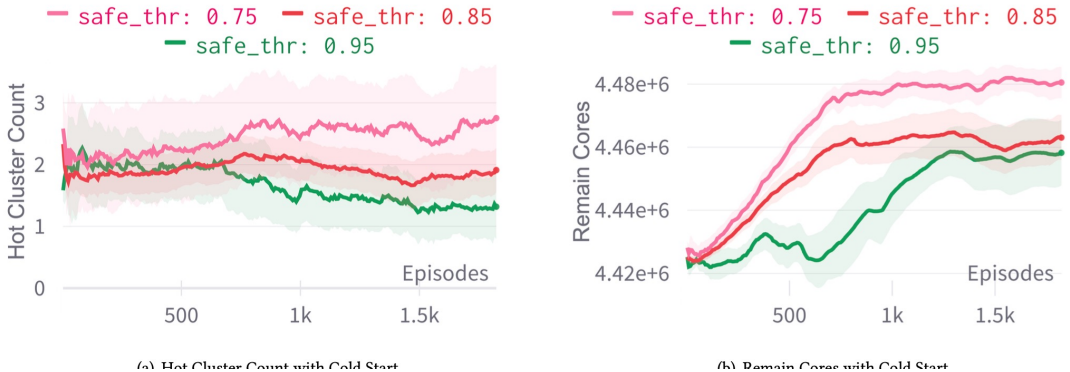
$$\lambda = \max(0, \lambda - \eta_\lambda(c - U(\theta^{k+1})))$$

Algorithm 1 C2MARL: Chance-Constrained Multi-Agent Reinforcement Learning

Initialization: the number of over-subscribers N, the cluster value network parameters θ^c, each over-subscriber i's state action value network parameters θ^i, discount factor γ, target network parameters $\bar{\theta}$, dual learning rate η_λ, safety threshold α, hot cluster indicator h'.
for Episode = $1, \cdots, M$ **do**
  Reset state and obtain the initial state $s_1 := [s_1^c, o_1^1, \ldots, o_1^N]$
  for $t = 1, \cdots, T$ and $s_t \neq$ terminal **do**
    for each agent i **do**
      With probability ϵ pick a random action a_t^i else choose action with the largest value in $Q_t^i(a_t^i, o_t^i)$;
    end for
    Execute global actions and get global reward r_t, constraint value c_t, next state $s_{t+1} := [s_{t+1}^c, o_{t+1}^1, \ldots, o_{t+1}^N]$;
    Store $(s_t, a_t, r_t, c_t, s_{t+1})$ to $\mathcal{R}$;
  end for
  for $k = 1, \cdots, K$ **do**
    Sample a random mini-batch transitions from $\mathcal{R}$;
    Update θ by minimizing Eq. 10;
    Update target Q network by soft update manner;
    Update dual parameter by Eq. 11;
  end for
end for

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

RL For Indirect VM Scheduling: oversubscription



Method	PM-Hot-R	S-Cores	0.75	0.85	0.95
Grid-0.2	30.9	80	✗	✗	✗
Grid-0.4	22.7	60.0	✓	✗	✗
Grid-0.6	0.0	40.0	✓	✓	✓
MA	0	49.2	✓	✓	✓
SL	0.2 ± 0.3	49.9 ± 3.3	✓	✓	✓
Our-0.75	9.8 ± 0.6	66.1 ± 6.7	✓	–	–
Our-0.85	6.6 ± 0.6	61.7 ± 9.2	✓	✓	–
Our-0.95	3.5 ± 0.5	59.9 ± 1.9	✓	✓	✓

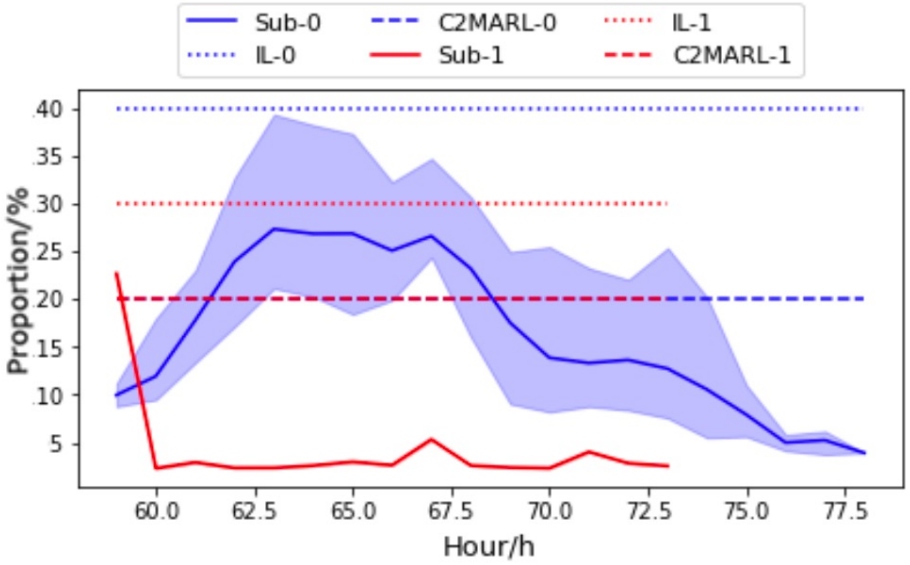


Figure 5: Staggered Peaks. The dashed and dotted lines are the action of C2MARL and SL agents. The solid line is the actual usage, and different colors represent different subscribers.

- Junjie, Sheng, et al. "Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning." Proceedings of the ACM Web Conference 2023.

Take Away