A Bandit Approach to Online Pricing for Heterogeneous Edge Resource Allocation

Jiaming Cheng*[1], Duong Thuy Anh Nguyen*[2], Lele Wang[1], Duong Tung Nguyen[2], Vijay Bhargava[1]

1. University of British Columbia, Vancouver, Canada
2. Arizona State University, Tempe, AZ, USA

Authors with * contribute equally to the work

Background

THE UNIVERSITY OF BRITISH COLUMBIA

Sellers

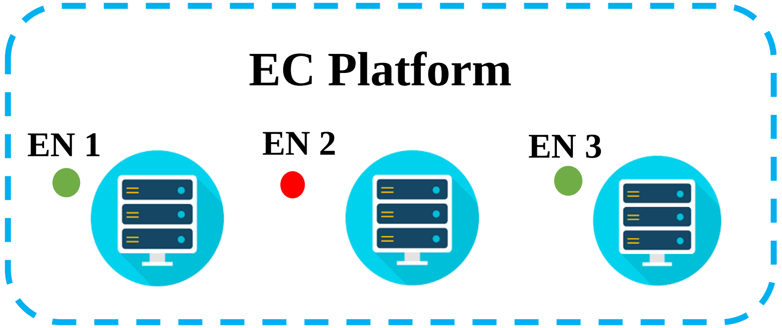
- Telcom central offices
- Servers at base stations (BSs)
- Idle machines in research lab
- Idle micro-DCs in campus buildings

- Provide available edge resources for customers

Background

- Telcom central offices
- Servers at base stations (BSs)
- Idle machines in research lab
- Idle micro-DCs in campus buildings

- Provide available edge resources for customers



- An entity manages a set of computing resources, i.e., in the form of VMs

- Price those available resources

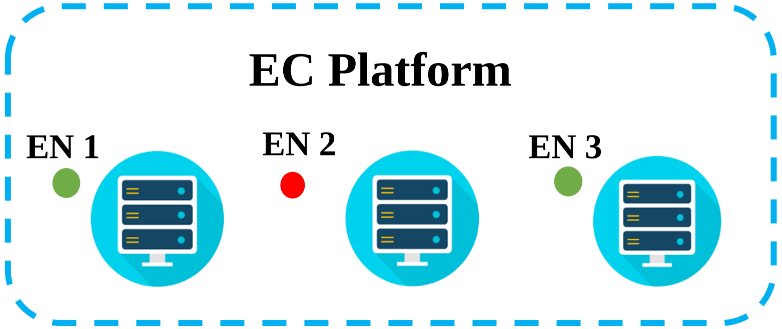
Background

THE UNIVERSITY OF BRITISH COLUMBIA

Sellers

- Telcom central offices
- Servers at base stations (BSs)
- Idle machine in research lab
- Idle micro-DC in campus buildings

- Provide available edge resources for customers



- An entity manages a set of computing resources, i.e., in the form of VMs

- Price those available resources to maximize profit

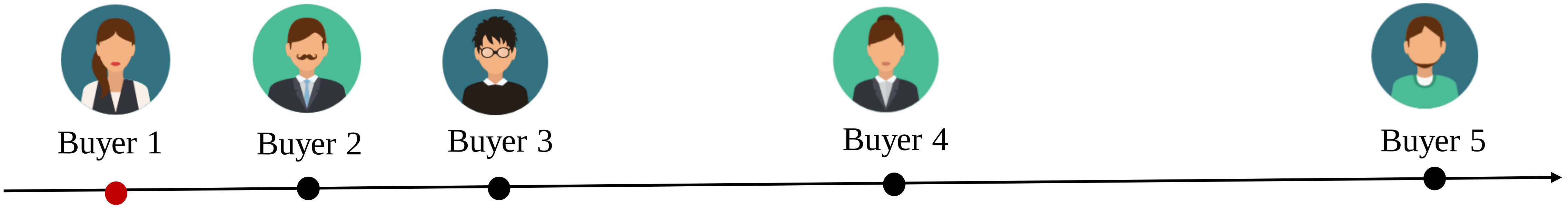
Buyers

- **Application/service providers**: Netflix; Online gaming company
- **Developers/individuals**: run some intensive computational tasks

- Lease/rent edge resources from EC platform
- Enhance quality of service (QoS) – reduce network delay

Problem statement - Multi-VM pricing scheme

THE UNIVERSITY OF BRITISH COLUMBIA

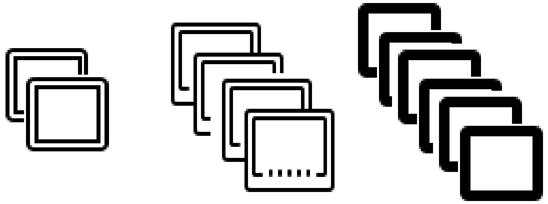


Platform

- Offers at most ***one unit*** of each VM (i) at EN (j) with a price $p_{i,j}^{t}$, where price vector : $(p_{i,j}^{t}) \in [0,1]$
- The platform can offer **<u>different types of VM</u>** for buyers to choose

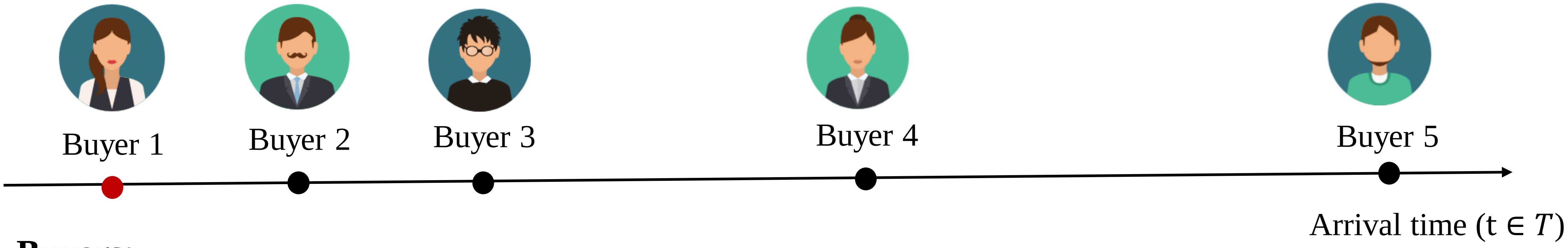
Hardware specification sheet

	Type	VM_1	VM_2	VM_3
	Type(i)	i=1	i=2	i=3
r=1	CPU (2-GHz)	1	2	4
r=2	RAM (GB)	8	16	32
r=3	Storage (TB)	0.5	1	2



cores

Problem statement - Multi-VM pricing scheme



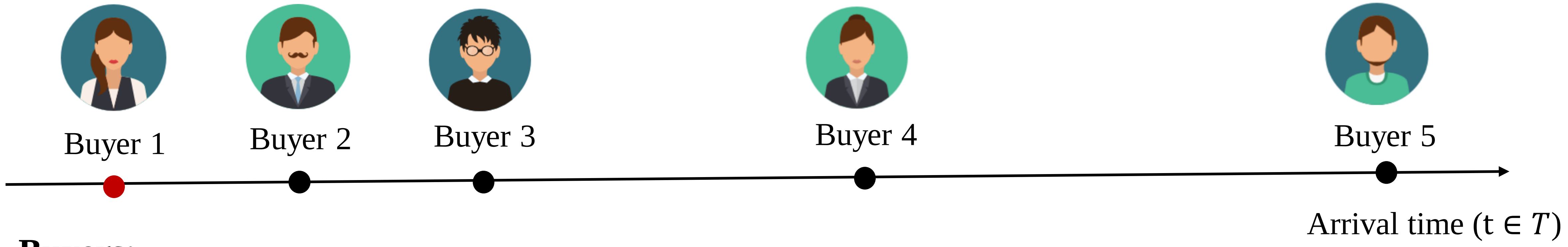
Buyers:

- There are **T** potential buyers' request arriving the platform sequentially
- Each buyer choose to procure a subset of products based on their valuations
- We define $v_{i,j}^{t}$ as the valuation for VM (i, j) with listed price $p_{i,j}^{t}$

EC platform

Type (i)	Location (j)	Price ($p_{i,j}^{t}$: \$/unit)
VM1	EN 1	$p_{1,1}^{t}$
VM2	EN 1	$p_{2,1}^{t}$
VM1	EN 3	$p_{1,3}^{t}$

Problem statement - Multi-VM pricing scheme

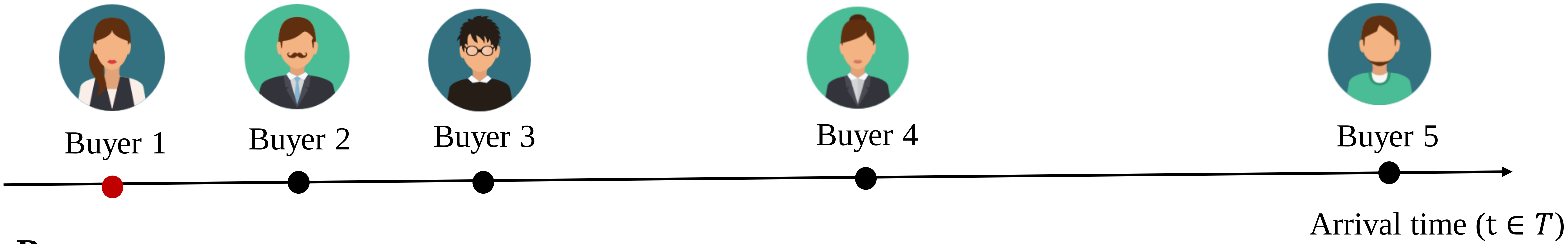


Buyers:

- There are **T** potential buyers' request arriving the platform sequentially
- Each buyer choose to procure a subset of products based on their valuations
- We define $v_{i,j}^{t}$ as the valuation for VM (i, j) with listed price $p_{i,j}^{t}$

The goal of the platform is to determine the price vector $p_{i,j}^{t}$ to maximize its revenue

Problem statement - Multi-VM pricing scheme



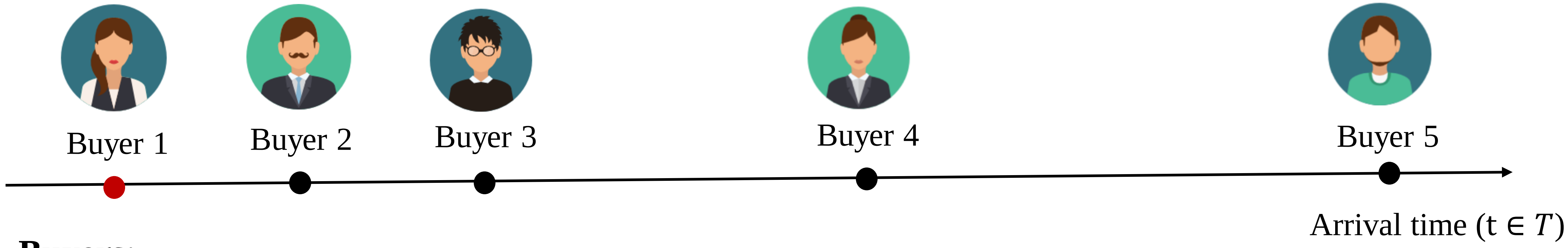
Buyers:

- There are **T** potential buyers' request arriving the platform sequentially
- Each buyer choose to procure a subset of products based on their valuations at the same time
- We define $v_{i,j}^{t}$ as the valuation for VM (i, j) with listed price $p_{i,j}^{t}$

$$\begin{cases} v_{i,j}^{t} \geq p_{i,j}^{t} & \text{accept} \\ v_{i,j}^{t} \leq p_{i,j}^{t} & \text{reject} \end{cases}$$

Problem statement - Multi-VM pricing scheme

THE UNIVERSITY OF BRITISH COLUMBIA



Buyers:

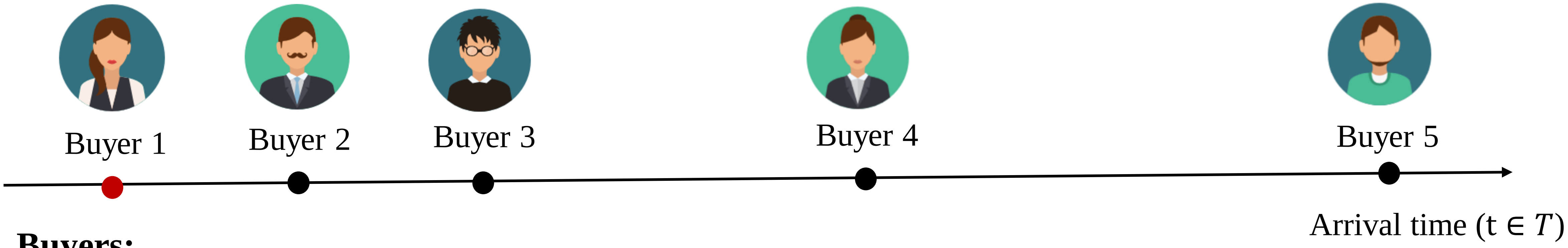
- There are **T** potential buyers' request arriving the platform sequentially
- Each buyer choose to procure a subset of products based on their valuations
- We define $v_{i,j}^t$ as the valuation for VM (i, j) with listed price $p_{i,j}^t$
- $v_{i,j}^t$ is called valuation function, which is unknown to the platform

$$c_{i,j}^t = \begin{cases} 1 & \text{If } v_{i,j}^t \geq p_{i,j}^t \\ 0 & \text{If } 0 \leq v_{i,j}^t < p_{i,j}^t \end{cases}$$

- Vector $c_{i,j}^t$ is denoted as resource consumption vector

$$c^t = (c_{1,1}^t, \dots, c_{1,N}^t, c_{2,1}^t, \dots, c_{M,N}^t) \in \{0,1\}^{MN}$$

Problem statement - Multi-VM pricing scheme



Buyers:

- There are **T** potential buyers' request arriving the platform sequentially
- Each buyer choose to procure a subset of products based on their valuations
- We define $v_{i,j}^{t}$ as the valuation for VM (i, j) with listed price $p_{i,j}^{t}$
- $v_{i,j}^{t}$ is called valuation function, which is unknown to the platform

$$c_{i,j}^{t} = \begin{cases} 1 & \text{If } v_{i,j}^{t} \geq p_{i,j}^{t} \\ 0 & \text{If } 0 \leq v_{i,j}^{t} < p_{i,j}^{t} \end{cases}$$

- The **utility of buyer *t*** can be expressed as

$$u_t = \sum_{i}^{M} \sum_{j}^{N} (v_{i,j}^{t} - p_{i,j}^{t}) c_{i,j}^{t}$$

The goal of each buyer is to maximize their own utility

Fixed pricing mechanism

- The platform determines a vector of *static price* that does not change over time
- Each buyer arrives and compare this static price ($\bar{p}_{i,j}^{t}$) with her own valuation

$$c_{i,j}^{t} = \begin{cases} 1 & \text{If } v_{i,j}^{t} \geq \bar{p}_{i,j}^{t} \\ 0 & \text{If } 0 \leq v_{i,j}^{t} < \bar{p}_{i,j}^{t} \end{cases}$$

However, the static price, i.e., $\bar{p}_{i,j}^{t}$, is hard to obtain since unknown information about $\boldsymbol{v}$

Fixed pricing mechanism

- The platform determines a vector of *static price* that does not change over time
- Each buyer arrives and compare this static price ($\bar{p}_{i,j}^{t}$) with her own valuation

$$c_{i,j}^{t} = \begin{cases} 1 & \text{If } v_{i,j}^{t} \geq \bar{p}_{i,j}^{t} \\ 0 & \text{If } 0 \leq v_{i,j}^{t} < \bar{p}_{i,j}^{t} \end{cases}$$

However, the static price, i.e., $\bar{p}_{i,j}^{t}$, is hard to obtain since unknown information about $\boldsymbol{v}$

Weakness:

- Different buyers may have varying attitudes even towards the same type of VM
- Static pricing scheme ignores the value of past observations.
- The determined price is less likely to maximize the total reward

Fixed pricing mechanism

- The platform determines a vector of *static price* that does not change over time
- Each buyer arrives and compare this static price ($\bar{p}_{i,j}^t$) with her own valuation

$$c_{i,j}^t = \begin{cases} 1 & \text{If } v_{i,j}^t \geq \bar{p}_{i,j}^t \\ 0 & \text{If } 0 \leq v_{i,j}^t < \bar{p}_{i,j}^t \end{cases}$$

However, the static price, i.e., $\bar{p}_{i,j}^t$, is hard to obtain since unknown information about $\boldsymbol{v}$

Weakness:

- Different buyers may have varying attitudes even towards the same type of VM
- Static scheme ignores the value of past observations.
- The determined price is less likely to maximize the total reward

Motivation:

- The platform aims to determine a “policy” to associate its decisions on price $p_{i,j}^t$ using past observations ((p^1, c^1),(p^2, c^2),…,(p^{t-1}, c^{t-1}))

Online posted pricing scheme

Goal: design online posted pricing scheme that allows the platform to make online decisions with performance guarantees.

- **Dynamic protocol**:
  1. The platform must determine the price of each VM *according to metric*
  2. Each buyer t arrives the platform one by one and compares its price for each VM with her valuation
     - If $p_{i,j}^t < v_{i,j}^t$, buyer (t) will purchase one unit of the VM, otherwise, reject;
  3. Update the cumulative reward and resource consumption c^t only if making a sale;

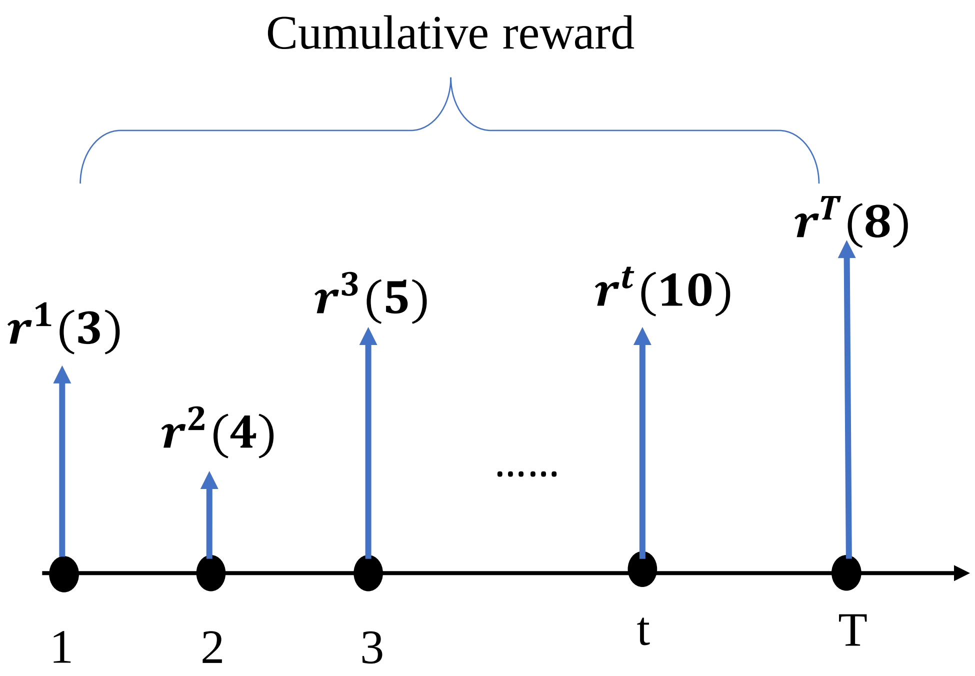
- **Solution**: cast this dynamic edge resource pricing as a Multi-armed bandit problem

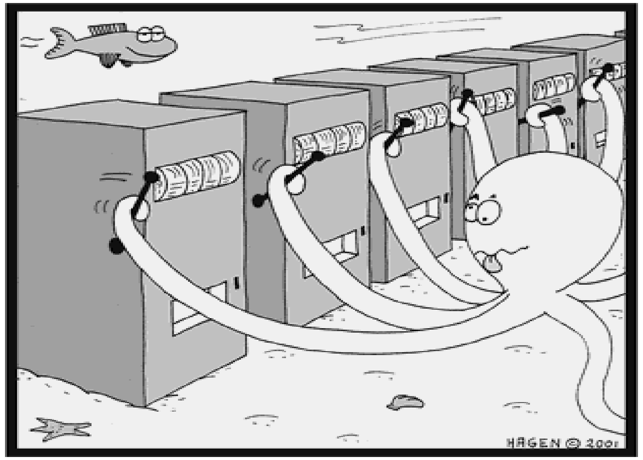
Recap:

THE UNIVERSITY OF BRITISH COLUMBIA

- $\boldsymbol{p_{i,j}^t} \in \{p_{i,j}^{t,1}, p_{i,j}^{t,2}, p_{i,j}^{t,3}, \dots \dots, p_{i,j}^{t,V}\}$: finite action space for the price, where v represents the different price options/levels
- $\boldsymbol{r^t} \in [0,1]$: in round t/when buyer t arrives.

$$\boldsymbol{r^t} = \sum_{i}^{M} \sum_{j}^{N} p_{i,j}^t \, c_{i,j}^t$$





- Goal: maximize the expected cumulative reward
- Question: which arm to select in each round ?
- Challenge: exploitation vs. exploration?

Regret Optimal Algorithm

- Reward maximization $\Longleftrightarrow$ Regret minimization

$$\max \mathrm{E}[r^t(p,\mu]) \Longleftrightarrow \min\left\{\sum_t^T \mathrm{E}[[r^t(p^*,\mu]] - \mathrm{E}\left[\sum_t^T r^t(p,\mu)\right]\right\}$$

Regret: Reg_T

- Lower bound: $Reg_T = \Omega(logT)$ [Lai & Robbins ‘85]

Regret Optimal Algorithm

- Reward maximization $\Longleftrightarrow$ Regret minimization

$$\max \mathrm{E}[r^t(p,\mu]) \Longleftrightarrow \min \left\{ \underbrace{\sum_t^T \mathrm{E}[[r^t(p^*,\mu]] - \mathrm{E}\left[\sum_t^T r^t(p,\mu)\right]}_{\text{Regret: } Reg_T} \right\}$$

- Lower bound: $Reg_T = \Omega(logT)$ [Lai & Robbins '85]
- Regret optimal algorithms ($Reg_T = \Theta\ (logT)$)
  - ➢ Upper-Confidence-Bound (UCB) [Lai & Robbins '85, Auer et al. '02]
  - ➢ Epsilon greedy [Lai & Robbins '85, Sutton & Barto '98]
  - ➢ Thompson sampling [Thompson '33, Agrawal & Goyal '12]

Recap: UCB algorithm

- $n_t(p)$: number of times price vector p is selected up to round t
- $\hat{r}_t(p)$: empirical (mean) reward of price vector p at round t (vs. $r(p)$ true (mean) reward)

UCB Algorithm: In round t, pull the arm that maximizes the following

- UCB index $\hat{r}_t(p) + \sqrt{\dfrac{2\log t}{n_t(p)}}$
- Why UCB works?
  - Chernoff – Hoeffding inequality states: $\mathrm{p}\left(\left|r_p - \hat{r}_p\right| < \epsilon\right) < e^{-2n_t\epsilon^2}$

$$\Longrightarrow \quad \mathrm{p}\left(r_p > \hat{r}_p + \sqrt{\frac{2\log t}{n_t(p)}}\right) < 1 - \frac{1}{t^4}$$

Contributions

- **Modeling:**
  1. Online detail-free posted pricing mechanism for multi-VM problem. (vs. single item)
  2. Consider computing power of VM and their geographical locations. (consider price as price)
- **Algorithm:**
  1. Derive
  2. Eliminate the need for prior knowledge of the demand distribution (distribution-free)
  3. Ensure truthfulness as the online posted price is independent of the newly arrived buyer's valuation
- **Simulations:**
  1. Performance comparison over three different scenarios
  2. Run-time analysis and comparison

KL-UCB Algorithm

Weakness:

- The UCB algorithm can perform suboptimally when r.v is not sub-Guassian

KL-UCB

- The algorithm selects the available price vector $\boldsymbol{p}$ with the highest $\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{KL}}$

$$\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{KL}} = \max\{q \in [0,1]: d(\hat{r}^t(p), q)\ \ n_t(p) \le f(t)\}$$

KL-UCB Algorithm

THE UNIVERSITY OF BRITISH COLUMBIA

Weakness:

- The UCB algorithm can perform suboptimally when r.v is not sub-Guassian

KL-UCB

- The algorithm selects the available price vector $\boldsymbol{p}$ with the highest $\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{KL}}$

$$\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{KL}} = \max\{q \in [0,1]: d(\hat{r}^t(p), q)\ \ n_t(p) \leq f(t)\}$$

$d(\hat{r}^t(p), q)$: KL divergence between two probability distributions

- Bernoulli distributions with parameters (u, v)

$$d(u,v) = u\ log\ \frac{u}{v} + (1-u)log\frac{1-u}{1-v}$$

- Exponential distribution: with parameters (u, v)
- $d(u,v) = \frac{u}{v} - 1 - \log(\frac{u}{v})$

KL-UCB Algorithm

Weakness:

- The UCB algorithm can perform suboptimally when r.v is not sub-Guassian

KL-UCB

- The algorithm selects the available price vector $\boldsymbol{p}$ with the highest $\mathbf{UCB}^{\mathbf{KL}}_{\mathbf{p,t}}$

$$\mathbf{UCB}^{\mathbf{KL}}_{\mathbf{p,t}} = \max\{q \in [0,1]: d(\hat{r}^t(p), q)\ \ n_t(p) \leq f(t)\}$$

Observe the consumption vector c^t and reward r^t

Min-max optimal strategy (MOSS)

MOSS

- The algorithm selects the available price vector $\boldsymbol{p}$ with the highest $\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}}$

$$\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}} = \hat{r}_t(p) + \sqrt{\frac{\max\{\log\left(\frac{T}{K n_t(p)}\right), 0\}}{n_t(p)}}$$

Min-max optimal strategy (MOSS)

THE UNIVERSITY OF BRITISH COLUMBIA

MOSS

- The algorithm selects the available price vector $\boldsymbol{p}$ with the highest $\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}}$

$$\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}} = \hat{r}_t(p)$$

- The index of an arm that has been drawn more than $\frac{n}{K}$ times is simply the empirical mean of the reward;

$$\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}} = \hat{r}_t(p) + \sqrt{\frac{\log\left(\frac{T}{Kn_t(p)}\right)}{n_t(p)}}$$

- For the others, their index is an UCB on their mean reward.

Min-max optimal strategy (MOSS)

MOSS

- The algorithm selects the available price vector $\boldsymbol{p}$ with the highest $\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}}$

$$\mathbf{UCB}_{\mathbf{p,t}}^{\mathbf{MOSS}} = \hat{r}_t(p) \ + \ \sqrt{\frac{\max\{\log\left(\frac{T}{Kn_t(p)}\right), 0\}}{n_t(p)}}$$

Observe the consumption vector c^t and reward r^t

Simulation

Simulation setup:

- T = 100000 buyers
- M = 3 types of VM
- N = 3 edge nodes (ENs)
- K = 20 pricing options (possible price vector)

M1 General Purpose Large	m1.large	7.5 GiB	2 vCPUs	840 GB (2 * 420 GB HDD)
M1 General Purpose Medium	m1.medium	3.75 GiB	1 vCPUs	410 GB HDD
M1 General Purpose Small	m1.small	1.7 GiB	1 vCPUs	160 GB HDD
M1 General Purpose Extra Large	m1.xlarge	15.0 GiB	4 vCPUs	1680 GB (4 * 420 GB HDD)
M2 High Memory Double Extra Large	m2.2xlarge	34.2 GiB	4 vCPUs	850 GB HDD
M2 High Memory Quadruple Extra Large	m2.4xlarge	68.4 GiB	8 vCPUs	1680 GB (2 * 840 GB HDD)
M2 High Memory Extra Large	m2.xlarge	17.1 GiB	2 vCPUs	420 GB HDD
M3 General Purpose Double Extra Large	m3.2xlarge	30.0 GiB	8 vCPUs	160 GB (2 * 80 GB SSD)
M3 General Purpose Large	m3.large	7.5 GiB	2 vCPUs	32 GB SSD

Simulation

Simulation setup:

- T = 100000 buyers
- M = 3 types of VM
- N = 3 edge nodes (ENs)
- K = 20 pricing options (possible price vector)
- **Three different scenarios to simulate buyer's valuation (v_t) :**
  - ✓ Uniform distribution: U[0,1]
  - ✓ Gaussian: with mean $\mu = 0.2$ and var $\sigma = 0.2$
  - ✓ Exponential: with mean $\mu = \frac{1}{\lambda} = 0.2$

Simulation

THE UNIVERSITY OF BRITISH COLUMBIA

Simulation setup:

- T = 100000 buyers
- M = 3 types of VM
- N = 3 edge nodes (ENs)
- K = 20 pricing options (possible price vector)
- **Three different scenarios to simulate buyer's valuation (v_t) :**
  - ✓ Uniform distribution: U[0,1]
  - ✓ Gaussian: with mean $\mu = 0.2$ and var $\sigma = 0.2$
  - ✓ Exponential: with mean $\mu = \frac{1}{\lambda} = 0.2$
- **Three baseline benchmarks**
  - ➢ Upper-Confidence-Bound (UCB) [Lai & Robbins '85, Auer et al. '02]
  - ➢ Epsilon greedy [Lai & Robbins '85, Sutton & Barto '98]
  - ➢ Thompson sampling [Thompson '33, Agrawal & Goyal '12]

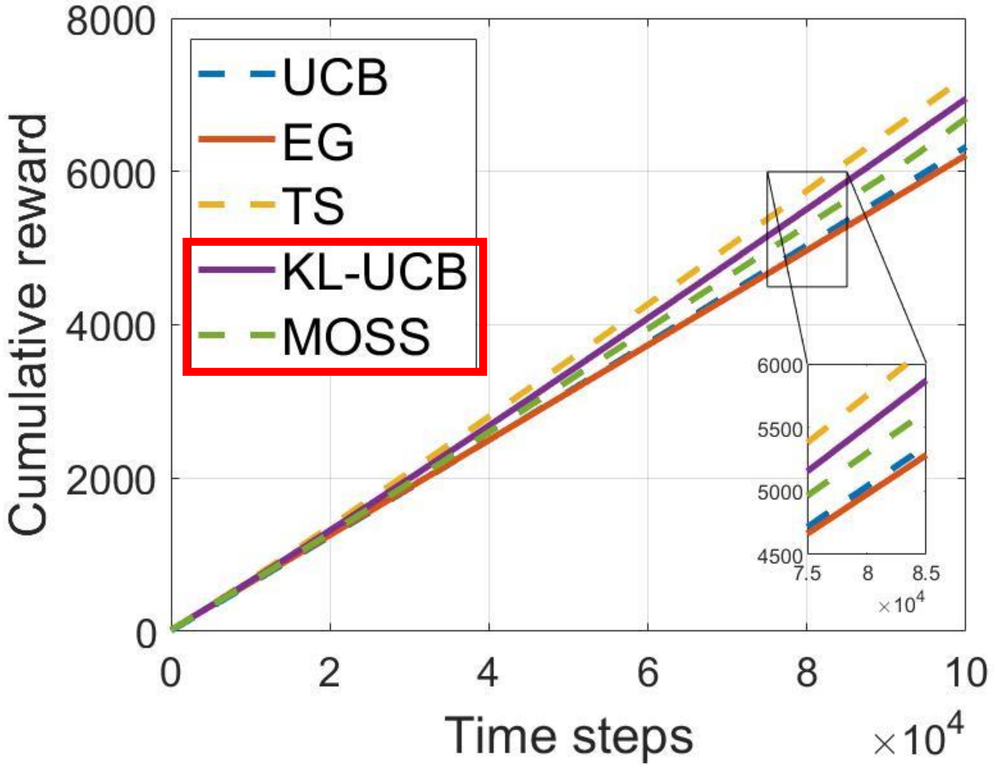
Simulation

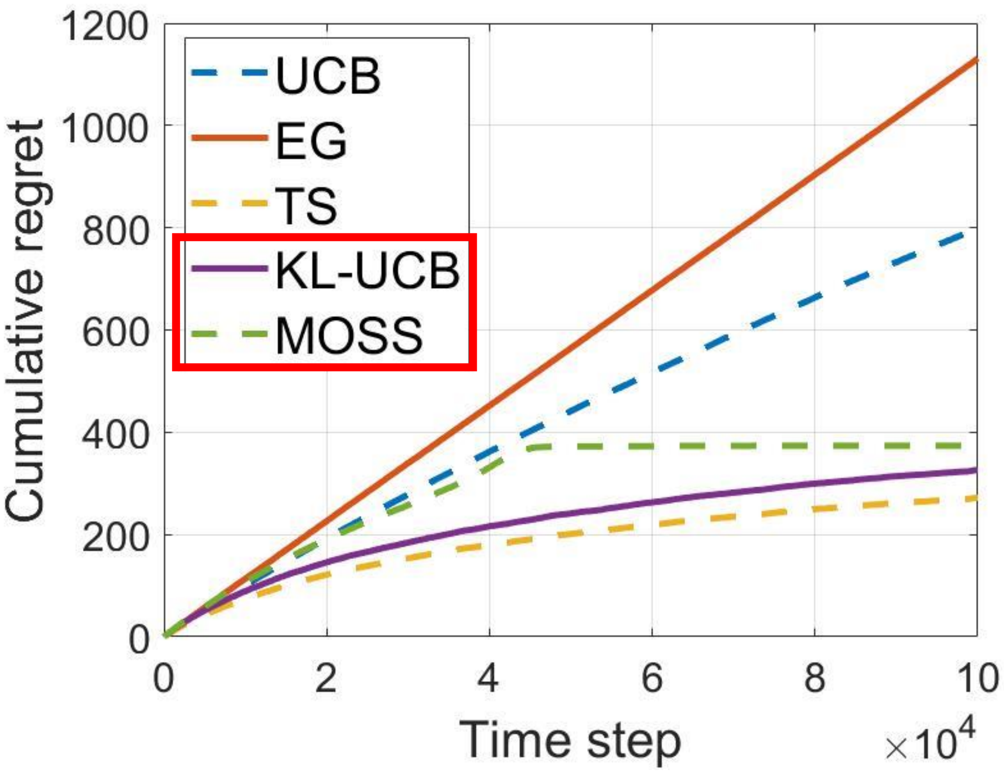
THE UNIVERSITY OF BRITISH COLUMBIA

Simulation setup:

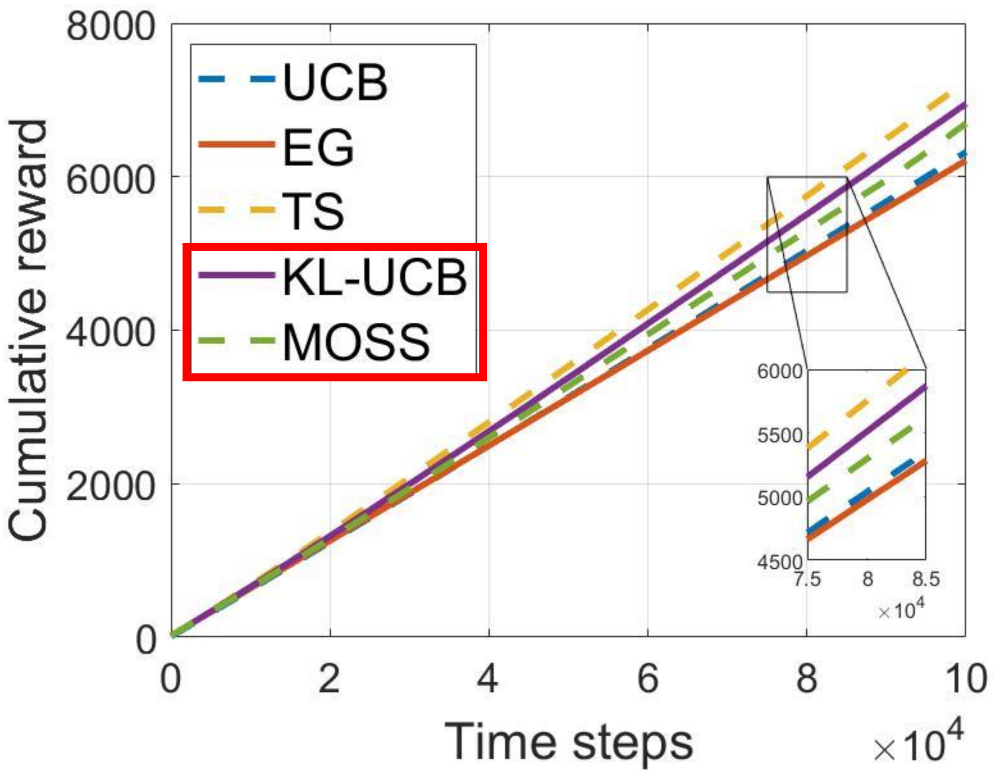
- T = 100000 buyers
- M = 3 types of VM
- N = 3 edge nodes (ENs)
- K = 20 pricing options (possible price vector)
- **Three different scenarios to simulate buyer's valuation (v_t) :**
  - ✓ Uniform distribution: U[0,1]
  - ✓ Gaussian: with mean $\mu = 0.2$ and var $\sigma = 0.2$
  - ✓ Exponential: with mean $\mu = \frac{1}{\lambda} = 0.2$
- **Three baseline benchmarks**
  - ➢ Upper-Confidence-Bound (UCB) [Lai & Robbins '85, Auer et al. '02]
  - ➢ Epsilon greedy [Lai & Robbins '85, Sutton & Barto '98]
  - ➢ Thompson sampling [Thompson '33, Agrawal & Goyal '12]
- **Performance metrices**
  - ➢ *Cumulative reward (higher - better)*
    - ➢ Cumulative sum of obtained reward from each round
  - ➢ *Cumulative regret (lower - better)*
    - ➢ The reward difference between optimal arm and selected arm.

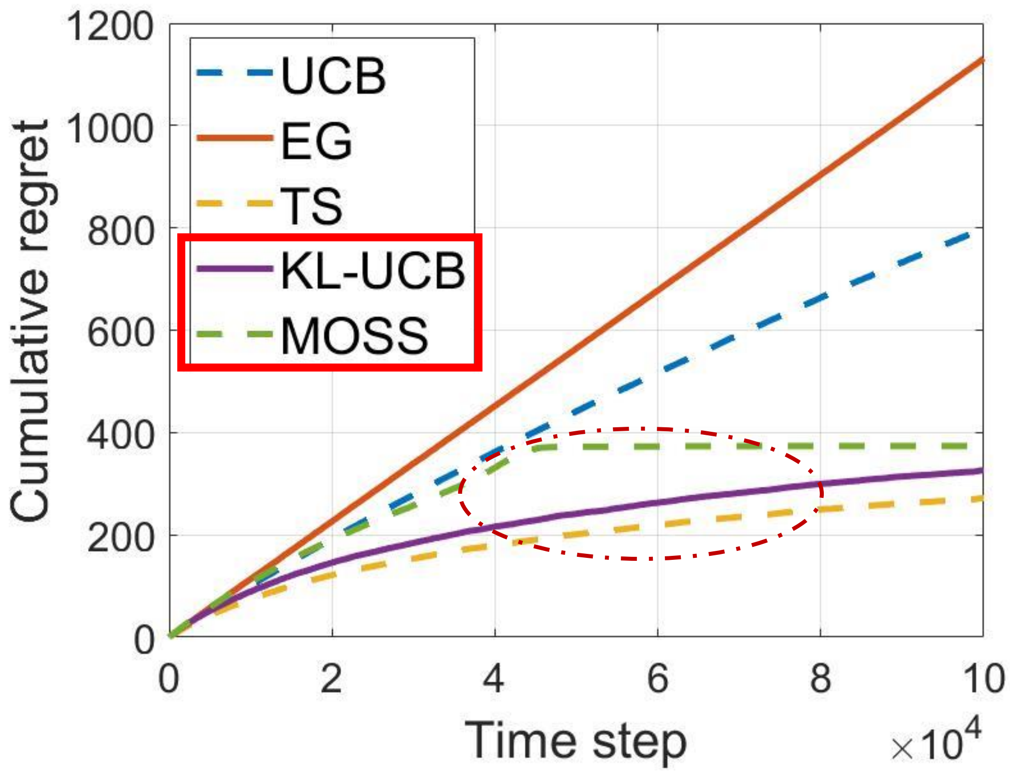
Gaussian: with mean μ = 0.2 and var σ = 0.2





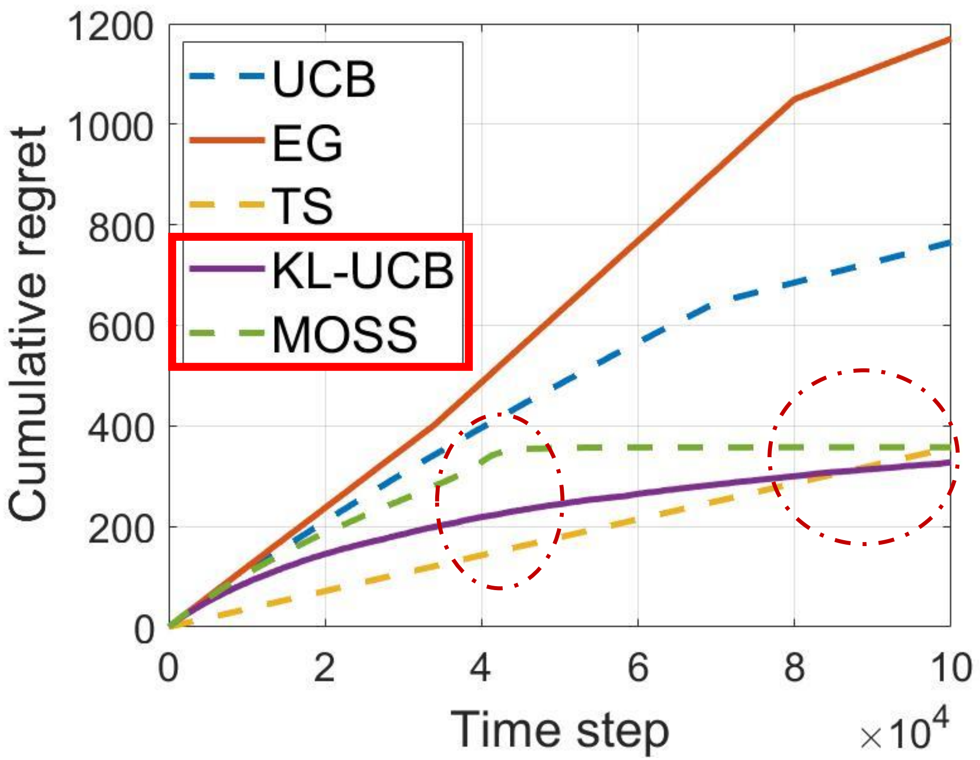
Gaussian: with mean μ = 0.2 and var σ = 0.2

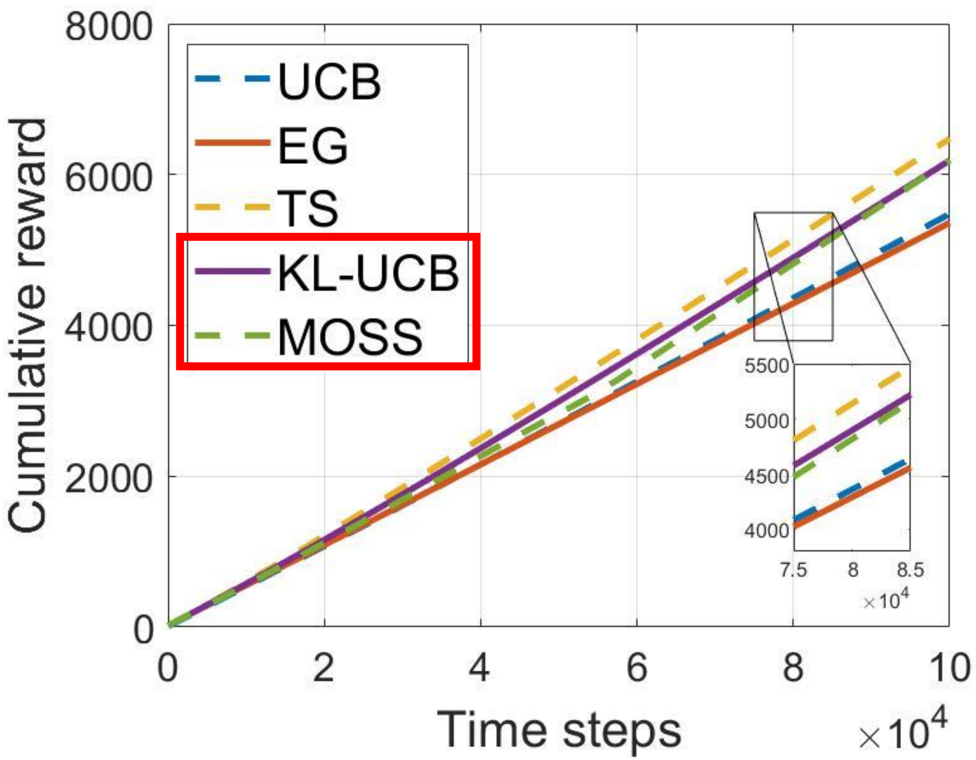




- **TS** achieve highest cumulative reward and lowest cumulative regret
- **KL-UCB** and **MOSS** also perform well compared to TS
- **KL-UCB** and **TS** enjoy flatter and lower cumulative regret compared to other schemes

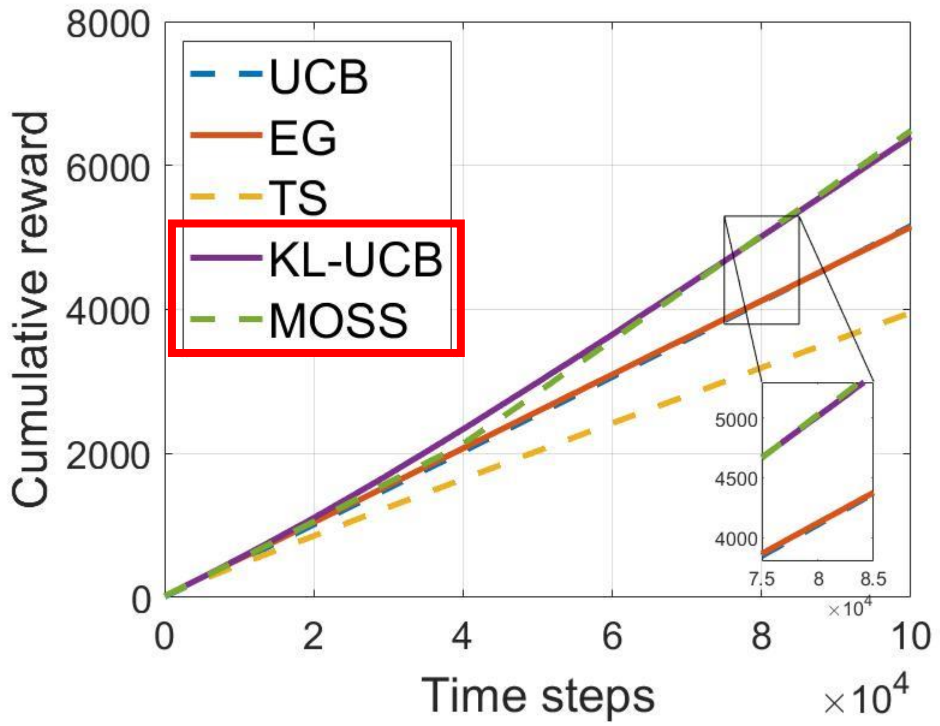
Uniform distribution: U[0,1]

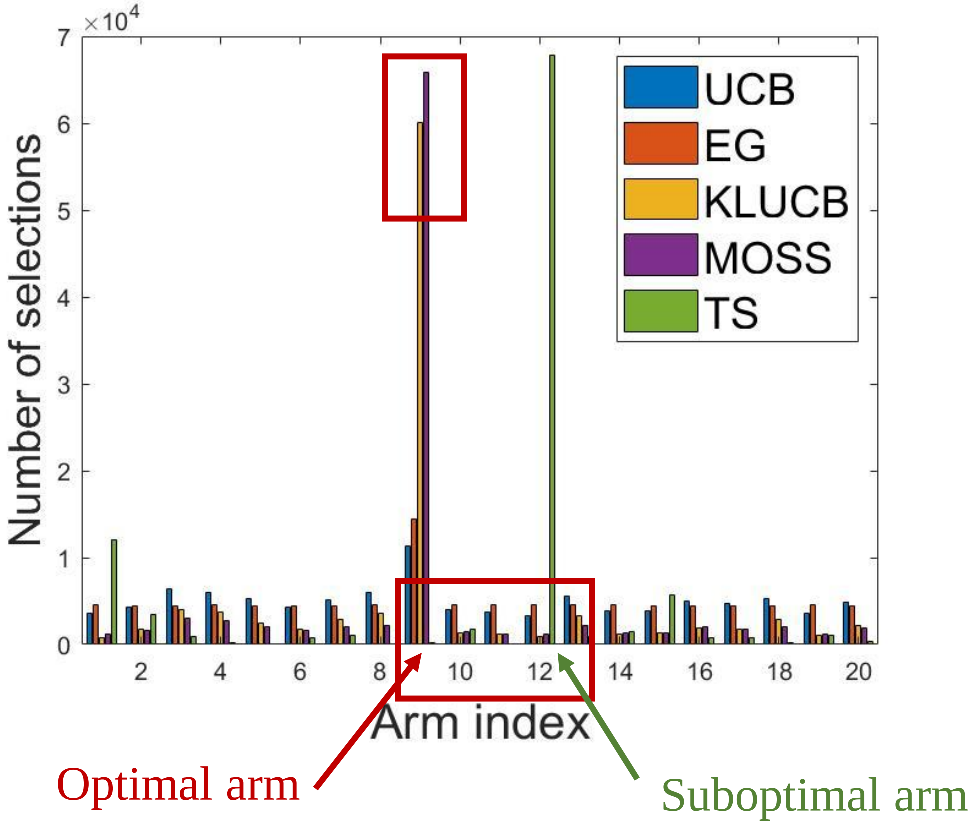




- **TS** achieve highest cumulative reward and lowest cumulative regret
- **KL-UCB** enjoy faster convergence rate of the cumulative regret

Exponential: with mean $\mu = 1/\lambda = 0.2$





- KLUCB and MOSS achieves best performance compared to other algorithms with the exponential scenario

- Higher doesn't always mean better
- **TS** suggests a suboptimal arm which lead to a larger cumulative regret
- **MOSS** picked the optimal price vector more often than **KLUCB**

Summary

- The proposed **KL-UCB** and **MOSS** based pricing scheme perform well over all scenarios in terms of cumulative regret and cumulative reward.
- **TS** performs optimally when dealing with ***uniform*** and ***Gaussian*** distribution but perform poorly on exponential distributions

Run-time results/setup

Number of arms (K)	100	200	300	400	500
UCB (sec)	2.76	3.75	5.04	7.62	12.76
EG (sec)	2.51	4.35	6.36	9.77	19.12
TS (sec)	25.93	49.87	74.21	97.28	120.38
KL-UCB (sec)	20.11	40.31	59.89	82.49	101.57
MOSS (sec)	3.11	4.33	6.94	7.84	12.72

- Run-time = Total execution time for total 100,000 buyers
- Take average run-time over 500 instances
- K = [100,500]

1: All the experiments are implemented in MATLAB on a desktop with an Intel Core i7-11700KF CPU and 16BG RAM

Run-time analysis

THE UNIVERSITY OF BRITISH COLUMBIA

Number of arms (K)	100	200	300	400	500
UCB (sec)	2.76	3.75	5.04	7.62	12.76
EG (sec)	2.51	4.35	6.36	9.77	19.12
TS (sec)	25.93	49.87	74.21	97.28	120.38
KL-UCB (sec)	20.11	40.31	59.89	82.49	101.57
MOSS (sec)	3.11	4.33	6.94	7.84	12.72

More time

- Larger price option space consumes more time

1: All the experiments are implemented in MATLAB on a desktop with an Intel Core i7-11700KF CPU and 16BG RAM

Run-time comparison

Number of arms (K)	100	200	300	400	500
UCB (sec)	2.76	3.75	5.04	7.62	12.76
EG (sec)	2.51	4.35	6.36	9.77	19.12
TS (sec)	25.93	49.87	74.21	97.28	120.38
KL-UCB (sec)	20.11	40.31	59.89	82.49	101.57
MOSS (sec)	3.11	4.33	6.94	7.84	12.72

1. **MOSS proves to be more computationally efficient**
   - MOSS improves the performance of original UCB but still keep the original index-based format.

Run-time comparisons

Number of arms (K)	100	200	300	400	500
UCB (sec)	2.76	3.75	5.04	7.62	12.76
EG (sec)	2.51	4.35	6.36	9.77	19.12
TS (sec)	25.93	49.87	74.21	97.28	120.38
KL-UCB (sec)	20.11	40.31	59.89	82.49	101.57
MOSS (sec)	3.11	4.33	6.94	7.84	12.72

1. **MOSS proves to be more computationally efficient**
   - MOSS improves the performance of original UCB but still keep the original index-based format.

2. **KL-UCB** incurs **a higher computational cost**
   - This process involves solving an optimization problem and computing the KL divergence between the estimated and prior distributions

Summary

MOSS

- Achieve the best distribution-free regret of $\sqrt{TK}$ for stochastic bandits
- Provide a unique UCB estimate based on the empirical mean reward
- Computationally efficient

- Higher cumulative regret/ slower convergence rate of regret

KL-UCB

- Distribution-free
- Enjoy a lower and faster convergence rate of the cumulative regret compared to MOSS

- Incurs a higher computational cost when number of arms become larger

Conclusion

- ✓ Cast the dynamic pricing for VM problem into an MAB problem
- ✓ Presented two novel online posted pricing mechanisms for allocating heterogeneous edge resources
  - ❑ Without prior knowledge of demand distribution
- ✓ Simulations
  - ❑ Good performance in both cumulative reward and cumulative regret
  - ❑ Time complexity analysis when size of arms is large

Conclusion

Thank you!

Q& A

UCB algorithm

THE UNIVERSITY OF BRITISH COLUMBIA

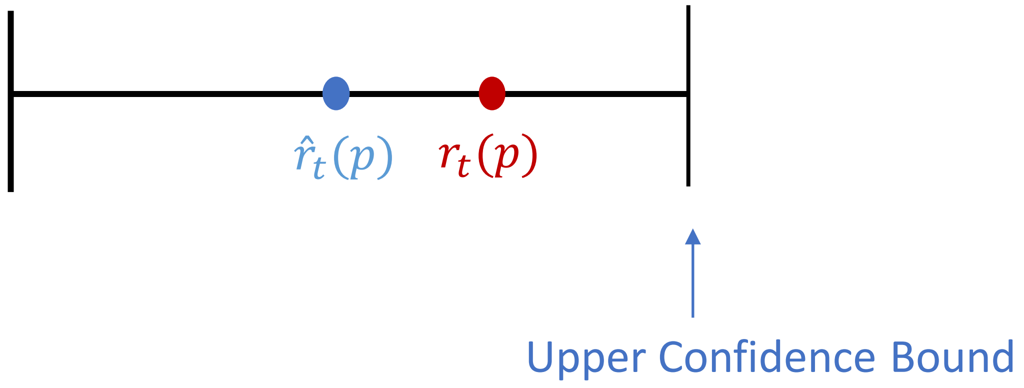
- $n_t(p)$: number of times price vector p is selected up to round t
- $\hat{r}^t(p)$: empirical (mean) reward of price vector p at round t

$$\hat{r}^t(p) = \frac{R_t}{n_t(p)} = \frac{\sum_{\tau=1}^{t} r^\tau(p)}{n_t(p)}$$

- R_t: denotes the accumulative sum of reward up to time t

UCB algorithm

- $n_t(p)$: number of times price vector p is selected up to round t
- $\hat{r}_t(p)$: empirical mean of reward for arm p at time t (vs. $r_t(p)$ true mean)



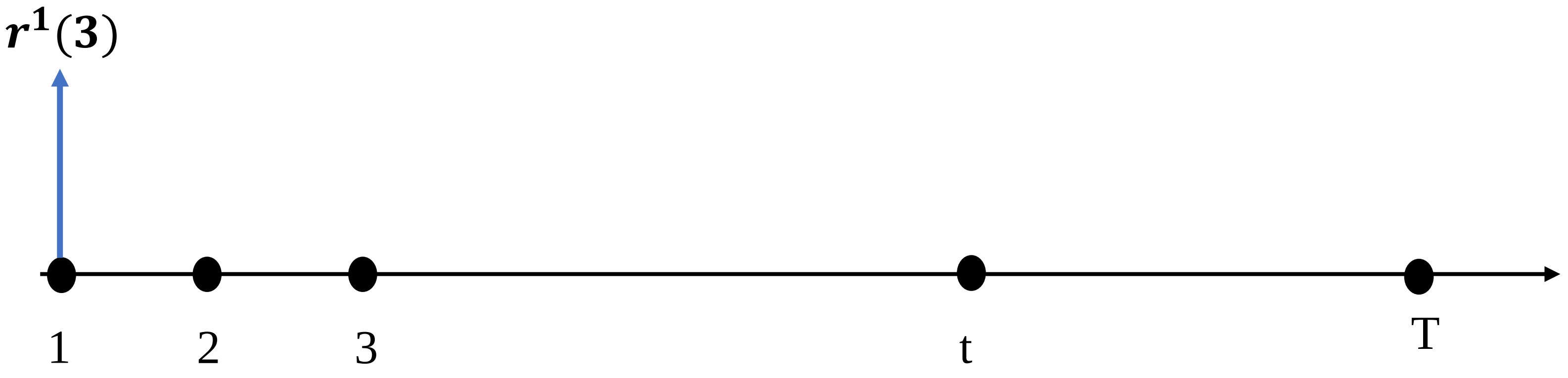
Problem formulation

- $\boldsymbol{p_{i,j}^{t}} \in \{p_{i,j}^{t,1}, p_{i,j}^{t,2}, p_{i,j}^{t,3}, \dots \dots, p_{i,j}^{t,V}\}$: finite action space for the price, where v represents the different price options/levels

$$p_{i,j}^{t,1} < p_{i,j}^{t,2} < p_{i,j}^{t,3} < \ \dots < p_{i,j}^{t,V}$$

- $\boldsymbol{r^{t}} \in [0,1]$: in round t/when buyer t arrives.

$$\boldsymbol{r^{t}} = \sum_{i}^{M} \sum_{j}^{N} p_{i,j}^{t}\, c_{i,j}^{t}$$



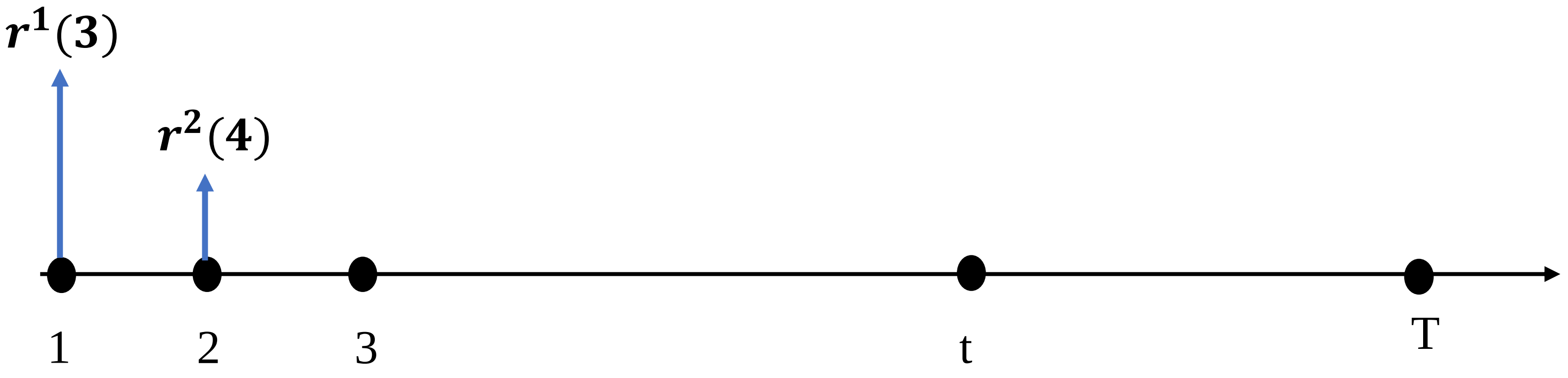
Problem formulation

- $\boldsymbol{p_{i,j}^t} \in \{p_{i,j}^{t,1}, p_{i,j}^{t,2}, p_{i,j}^{t,3}, \ldots \ldots, p_{i,j}^{t,V}\}$: finite action space for the price, where v represents the different price options/levels

$$p_{i,j}^{t,1} < p_{i,j}^{t,2} < p_{i,j}^{t,3} < \ \ldots < p_{i,j}^{t,V}$$

- $\boldsymbol{r^t} \in [0,1]$: in round t/when buyer t arrives.

$$\boldsymbol{r^t} = \sum_{i}^{M} \sum_{j}^{N} p_{i,j}^{t}\, c_{i,j}^{t}$$



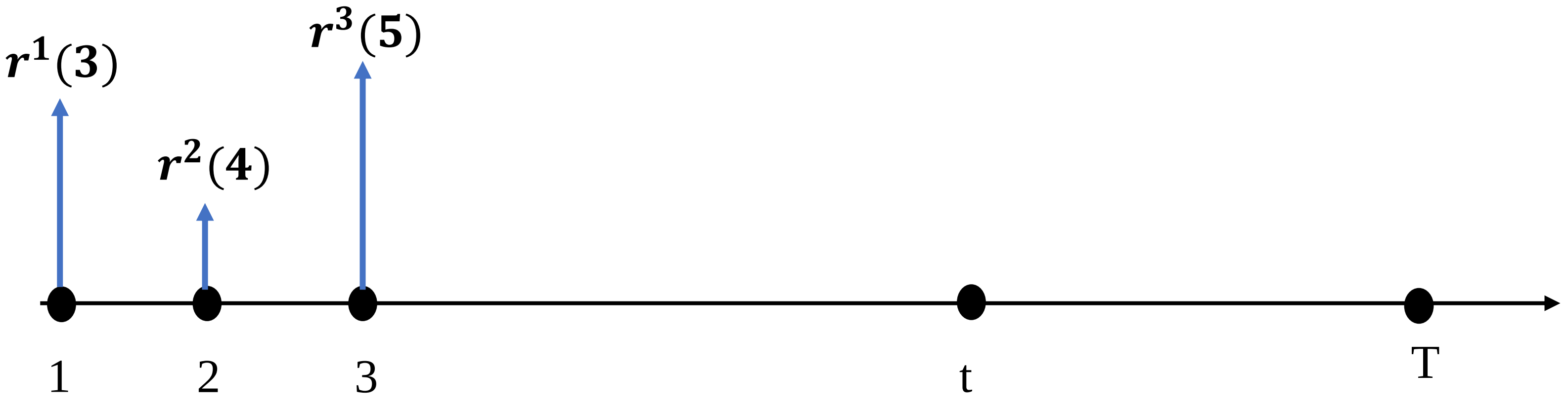
Problem formulation

- $\boldsymbol{p_{i,j}^t} \in \{p_{i,j}^{t,1}, p_{i,j}^{t,2}, p_{i,j}^{t,3}, \ldots \ldots, p_{i,j}^{t,V}\}$: finite action space for the price, where v represents the different price options/levels

$$p_{i,j}^{t,1} < p_{i,j}^{t,2} < p_{i,j}^{t,3} < \ldots < p_{i,j}^{t,V}$$

- $\boldsymbol{r^t} \in [0,1]$: in round t/when buyer t arrives.

$$\boldsymbol{r^t} = \sum_{i}^{M} \sum_{j}^{N} p_{i,j}^t \, c_{i,j}^t$$



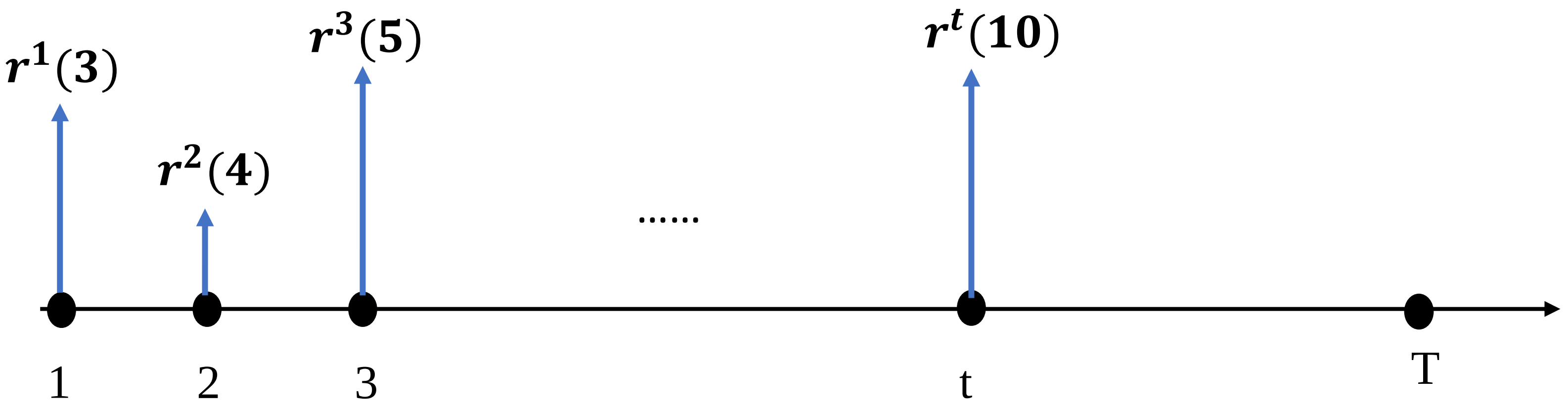
Problem formulation

- $\boldsymbol{p_{i,j}^{t}} \in \{p_{i,j}^{t,1}, p_{i,j}^{t,2}, p_{i,j}^{t,3}, \dots\dots, p_{i,j}^{t,V}\}$: finite action space for the price, where v represents the different price options/levels

$$p_{i,j}^{t,1} < p_{i,j}^{t,2} < p_{i,j}^{t,3} < \dots < p_{i,j}^{t,V}$$

- $\boldsymbol{r^{t}} \in [0,1]$: in round t/when buyer t arrives.

$$\boldsymbol{r^{t}} = \sum_{i}^{M}\sum_{j}^{N} p_{i,j}^{t}\, c_{i,j}^{t}$$



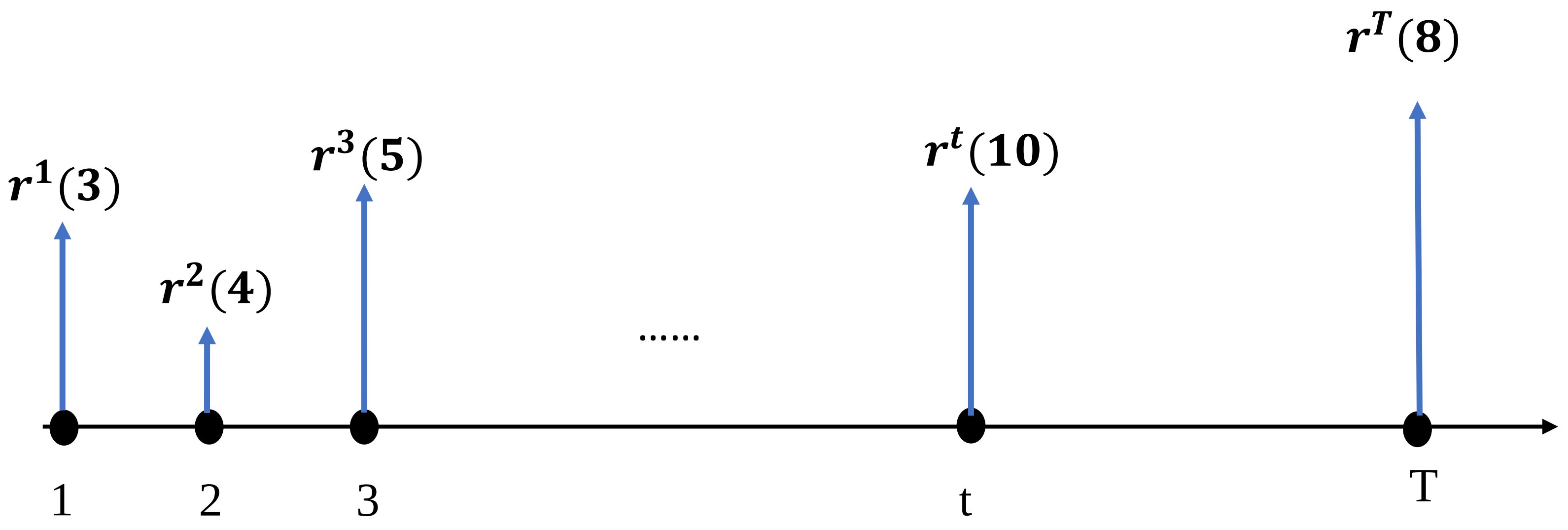
Problem formulation

- $\boldsymbol{p_{i,j}^{t}} \in \{p_{i,j}^{t,1}, p_{i,j}^{t,2}, p_{i,j}^{t,3}, \dots\dots, p_{i,j}^{t,V}\}$: finite action space for the price, where v represents the different price options/levels

$$p_{i,j}^{t,1} < p_{i,j}^{t,2} < p_{i,j}^{t,3} < \dots < p_{i,j}^{t,V}$$

- $\boldsymbol{r^{t}} \in [0,1]$: in round t/when buyer t arrives.

$$\boldsymbol{r^{t}} = \sum_{i}^{M} \sum_{j}^{N} p_{i,j}^{t}\, c_{i,j}^{t}$$



Dynamic pricing scheme

THE UNIVERSITY OF BRITISH COLUMBIA

$\boldsymbol{\lambda_p}$: fractional relaxation for the number of buyers by which a given price p is accepted.

We can write this dynamic pricing problem in the form of the linear programming problem, denoted by LP

Dynamic pricing scheme

THE UNIVERSITY OF BRITISH COLUMBIA

$\boldsymbol{\lambda_p}$: fractional relaxation for the number of buyers by which a given price p is accepted.
We can write this dynamic pricing problem in the form of the linear programming problem, denoted by LP

$$\max_{\lambda} \lambda_p \, r(p,\mu)$$

$$s.t. \sum_{p \in P} \lambda_p \, c_{i,j}(p,\mu) \leq C_{i,j} \, , \forall \, i,j$$

$$\sum_{p \in P} c_{i,j}(p,\mu) \leq T$$

$$\lambda_p \geq 0, \forall \text{ p}$$

Dynamic pricing scheme

THE UNIVERSITY OF BRITISH COLUMBIA

$\boldsymbol{\lambda_p}$: fractional relaxation for the number of buyers by which a given price p is accepted.
We can write this dynamic pricing problem in the form of the linear programming problem, denoted by LP

$$\max_{\lambda} \lambda_p \, r(p,\mu)$$

$$s.t. \sum_{p \in P} \lambda_p \, c_{i,j}(p,\mu) \leq C_{i,j}, \forall \, i,j$$

Resource capacity constraint for each type VM

$$\sum_{p \in P} c_{i,j}(p,\mu) \leq T$$

$$\lambda_p \geq 0, \forall \text{ p}$$

Dynamic pricing scheme

$\boldsymbol{\lambda_p}$: fractional relaxation for the number of buyers by which a given price p is accepted.
We can write this dynamic pricing problem in the form of the linear programming problem, denoted by LP

$$\max_{\lambda} \lambda_p \, r(p, \mu)$$

$$s.t. \sum_{p \in P} \lambda_p \, c_{i,j}(p, \mu) \leq C_{i,j}, \forall \, i, j$$

$$\sum_{p \in P} c_{i,j}(p, \mu) \leq T$$

⟹ Time runs out/ no more buyers

$$\lambda_p \geq 0, \forall \text{ p}$$

Dynamic pricing scheme

Resource capacity constraint ($C_{i,j}$) is limited in practice.
For simplicity, we consider unlimited case where $T \ll C_{i,j}$

Unlimited case ($T \ll C_{i,j}$) => reduce the problem into the following LP

$$\max_{\lambda} \lambda_p \, r(p, \mu)$$

$$s.t. \sum_{p \in P} c_{i,j}(p, \mu) \leq T, \forall\, i, j$$

$$\lambda_p \geq 0, \forall \text{ p}$$