Learning + Queueing: Operating Dynamic Online Service Platforms under Uncertain Payoffs

Jiaming Xu

The Fuqua School of Business
Duke University

Joint work with
Wei-Kang Hsu, Xiaojun Lin, and Mark R. Bell (Purdue ECE)

Jun 24, 2019, EC Tutorial

Motivation

Uncertain Dynamics

Uncertain Payoffs/Features

- Performance optimization, NUM/Lyapunov-based control
- Strength: dealing with uncertainty in agent dynamics
- Weakness: key agent features (e.g. click-through rate of an online ad) are assumed known

- Offline and online learning: Multi-armed bandit
- Strength: resolving uncertainty in agent features
- Weakness: often assume a fixed instance with no agent dynamics

Online Service Platforms: Uncertainty in both Agent Dynamics and Agent Features

- Online service platforms bring consumers and providers together
- Key problem: assigning clients to servers to maximize total payoffs
- Typically face uncertainty in both agent dynamics and agent features

Examples with Both Types of Uncertainty

	Online shopping: Stitch Fix	**Online Ads: Google AdWords**	**Crowd sourcing: Amazon Mechanical Turk**
Clients	Customers buying clothing	Advertisers	Posted jobs
Servers	Stylists recommend clothing	Slots on webpages for display ads	Workers
Dynamics	Customers arrive and depart	Advertisers come and go	Jobs arrive and depart
Features	Customer satisfaction	Click-through-rate	Completion efficiency

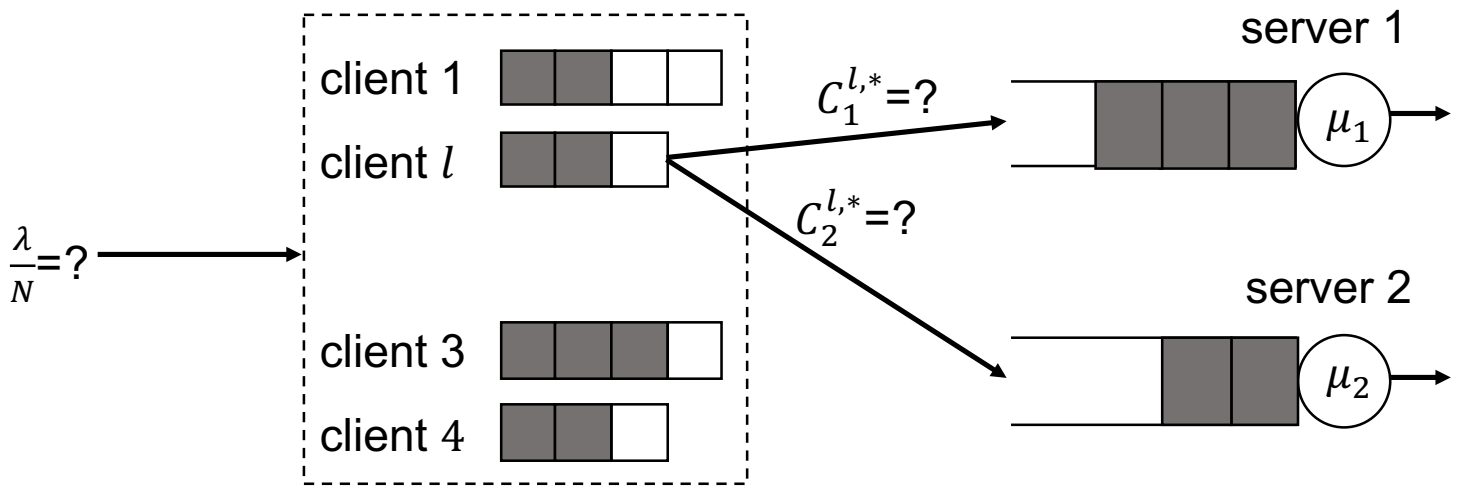
Key Take-Away Message

- Just picking a good algorithm from each field does not work well
- Our contribution: Developed an utility-guided online learning and task assignment policy with provably-low regret

Outline of the Talk

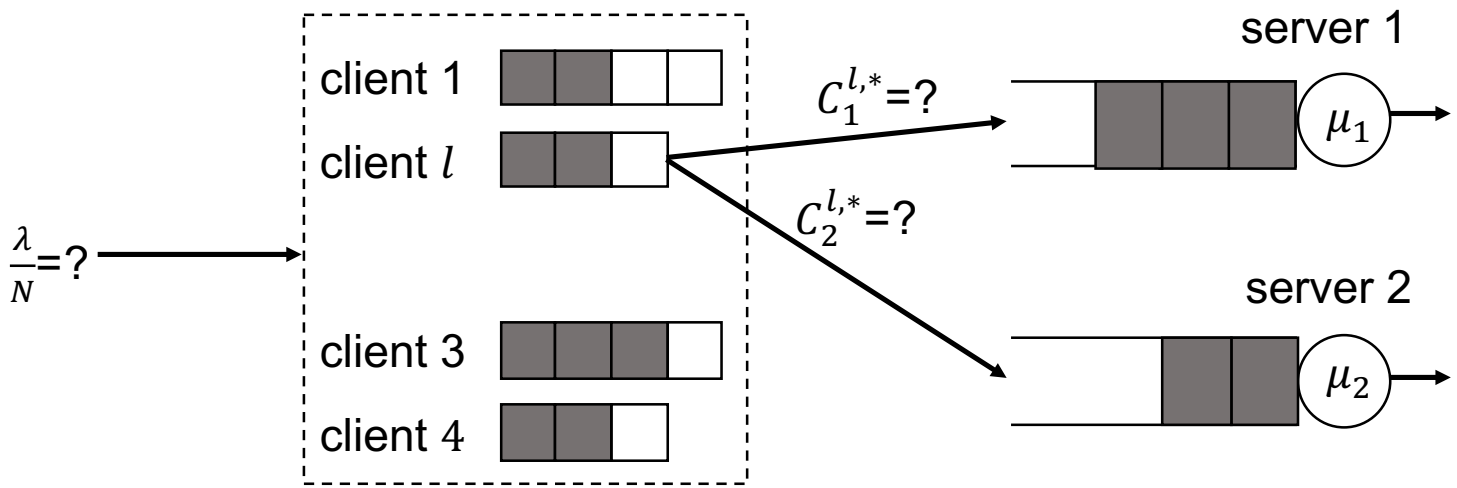
- System Model
- Two Failed Attempts
- Utility-guided Learning and Task Assignment Policy
- Intuitions behind the Main Results
- Conclusions and Remarks

System Model Inspired by [Johari-Kamble-Kanoria '17]



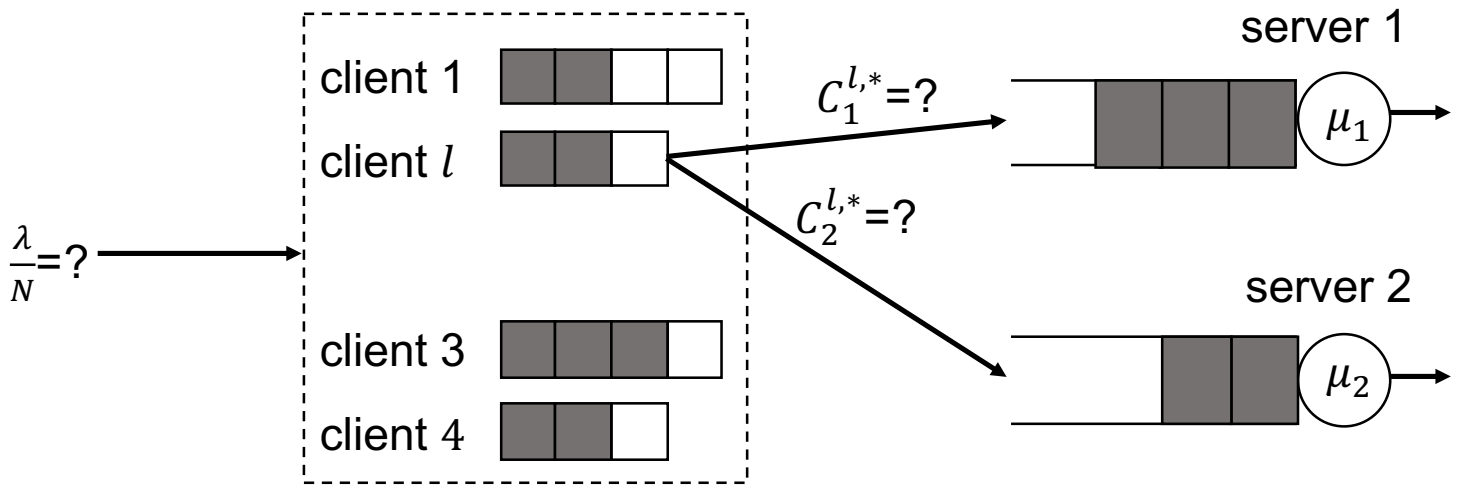
- Client arrives with rate $\frac{\lambda}{N}$
- $\mathrm{Geom}(N)$ number of tasks per client (Tasks arrive with rate λ)
- J servers: server j serves exactly μ_j tasks per time-slot

System Model Inspired by [Johari-Kamble-Kanoria '17]



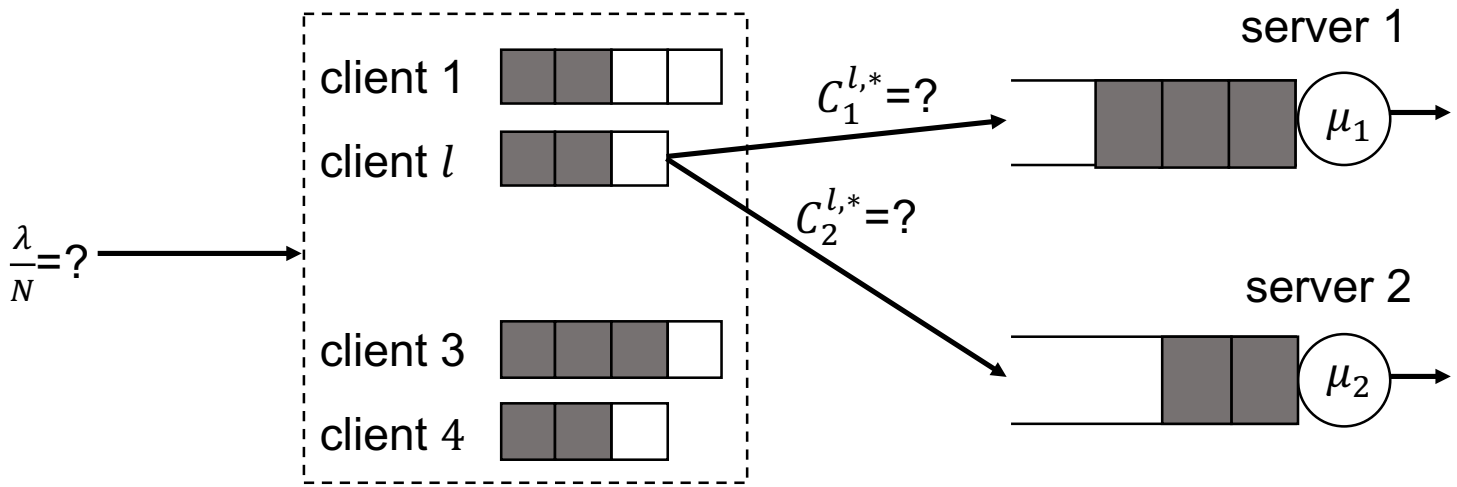
- Client arrives with rate $\frac{\lambda}{N}$
- $\mathrm{Geom}(N)$ number of tasks per client (Tasks arrive with rate λ)
- J servers: server j serves exactly μ_j tasks per time-slot
- Bernoulli random *payoff feedback* with unknown mean $C_j^{l,*}$ if a task from client l is completed by server j
- Goal: assign tasks to servers so that the system payoff is maximized

System Model: Uncertainty Dynamics and Payoffs



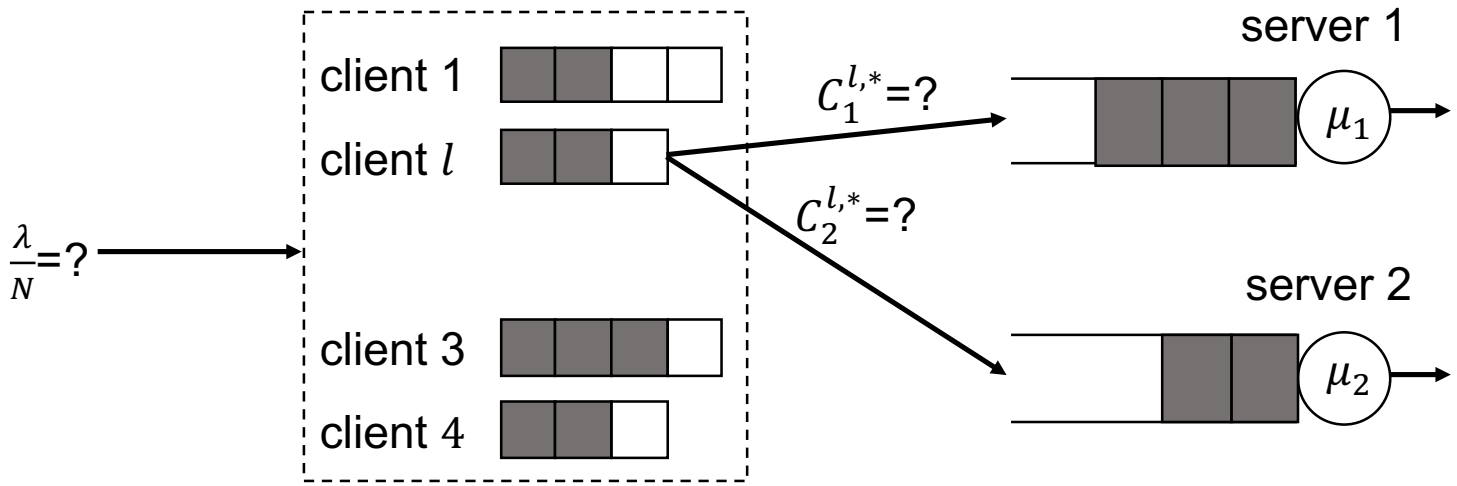
- If payoff parameters were known
  - typical queue control problem
  - Focus would have been on dealing with uncertain dynamics: λ, N

System Model: Uncertainty Dynamics and Payoffs

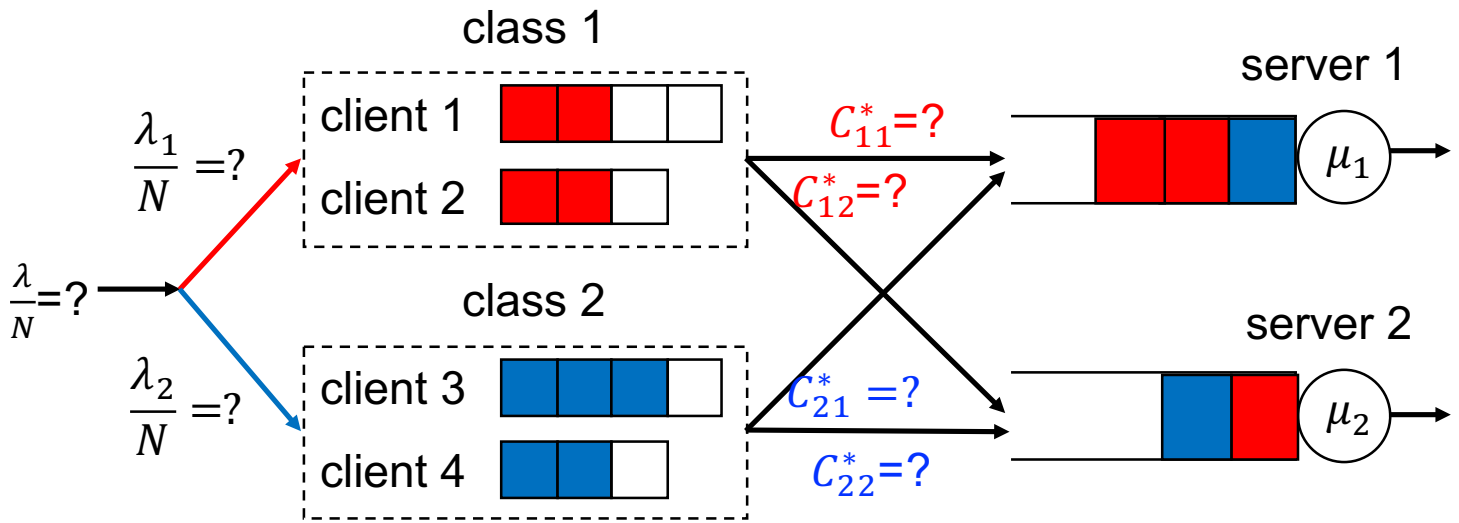


- If payoff parameters were known
  - typical queue control problem
  - Focus would have been on dealing with uncertain dynamics: λ, N
- Uncertain features: Payoff vector $\left[C_j^{l,*}\right]$
- Must combine queuing with learning (of payoff parameters from feedback)

System Model: Add a Distribution of Payoff-Vectors



System Model: Add a Distribution of Payoff-Vectors



- For each client, her unknown payoff-vector $\left[C_j^{l,*}\right]$ is chosen as:

$$\mathbb{P}\left\{\left[C_j^{l,*}\right] = \left[C_{ij}^*\right]\right\} = \rho_i, \quad \forall i = 1, \ldots, I$$

  in which case we say the client is of class i unknown

- $\lambda_i = \lambda\rho_i$: unknown task arrival-rate of class i

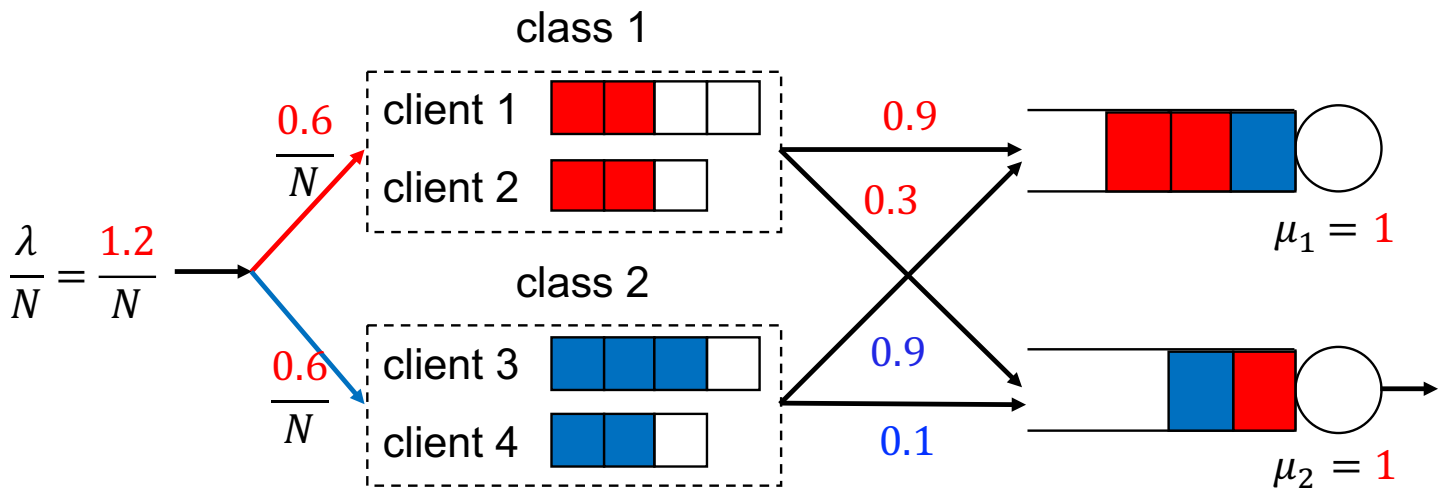
Oracle Upper bound R^*

Performance upper bound: with perfect information $\left(\lambda_i, C_{ij}^*\right)$

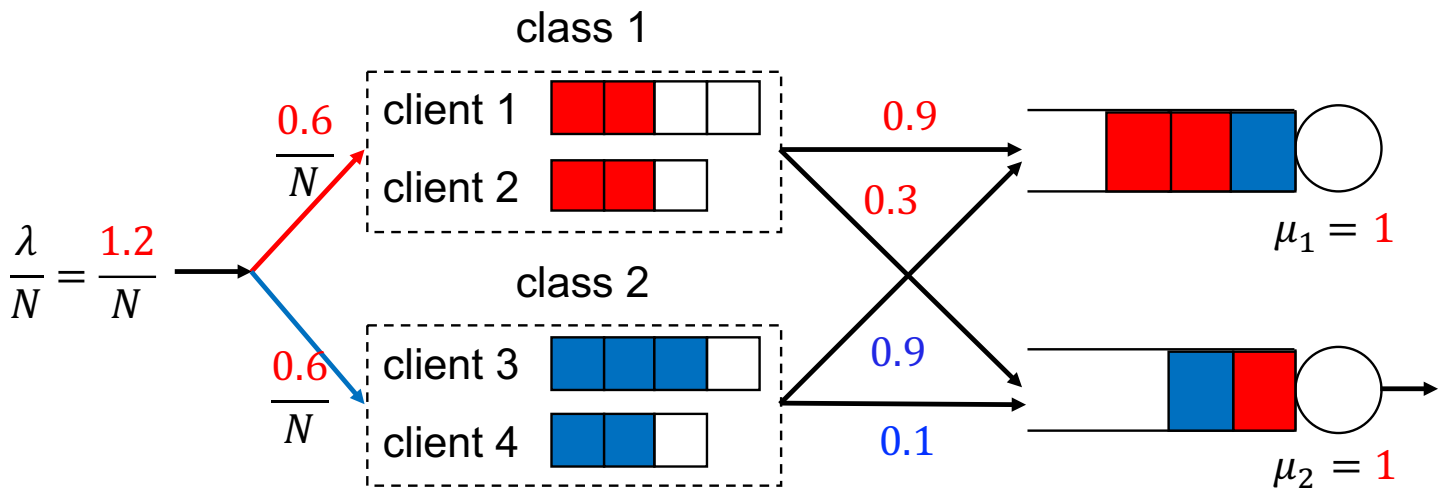
$$
\begin{aligned}
R^* = \max_{[p_{ij}] \geq 0} \quad & \sum_{i=1}^{I} \lambda_i \sum_{j=1}^{J} p_{ij} C_{ij}^* \\
\text{s.t.} \quad & \sum_{i=1}^{I} \lambda_i p_{ij} \leq \mu_j \text{ for all servers } j = 1, \ldots, J \\
& \sum_{j=1}^{J} p_{ij} = 1 \text{ for all classes } i = 1, \ldots, I
\end{aligned}
$$

p_{ij}: average fraction of tasks from class i that are assigned to server j

Oracle Upper bound R^*: A Running Example

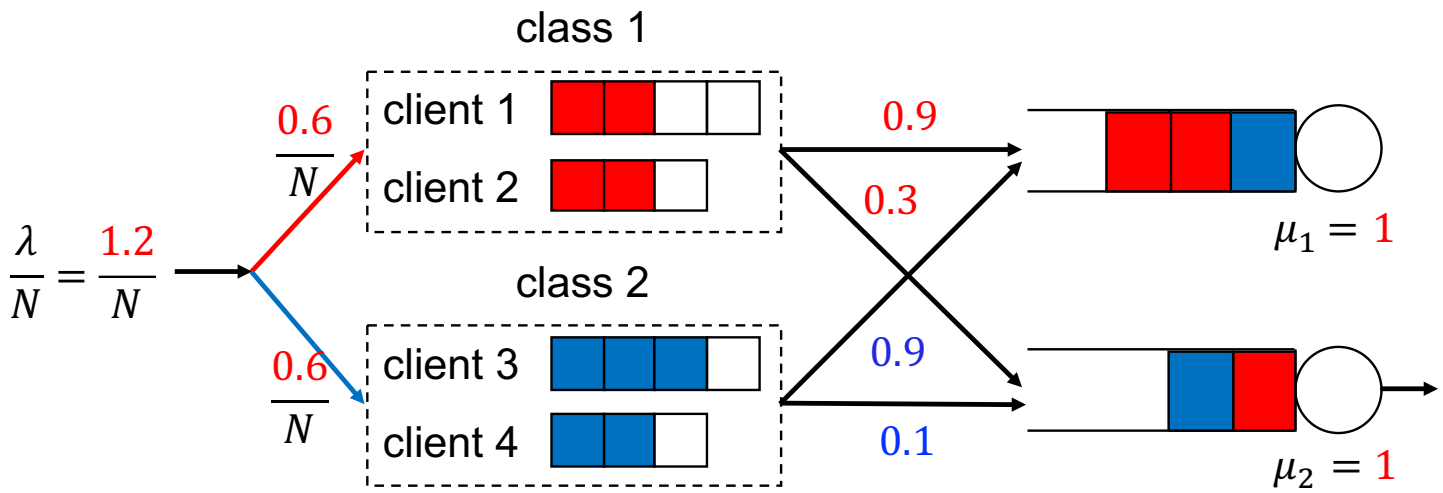


Oracle Upper bound R^*: A Running Example



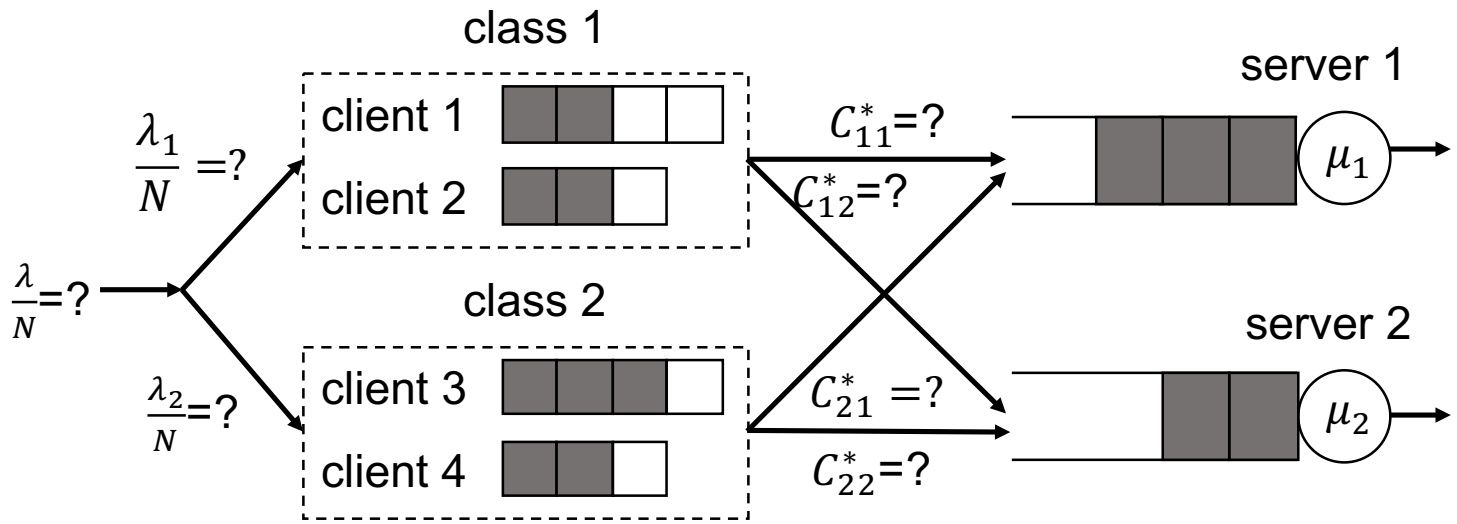
- Optimal solution: $p_{11}^* = \frac{2}{3},\ p_{12}^* = \frac{1}{3},\ p_{21}^* = 1,\ p_{22}^* = 0$
  - Both classes prefer server 1
  - However, server 1's capacity is insufficient to support both classes
  - Better to kick class 1 out since it is second-preference is better

Oracle Upper bound R^*: A Running Example



- Optimal solution: $p_{11}^* = \frac{2}{3},\ p_{12}^* = \frac{1}{3},\ p_{21}^* = 1,\ p_{22}^* = 0$
  - Both classes prefer server 1
  - However, server 1's capacity is insufficient to support both classes
  - Better to kick class 1 out since it is second-preference is better
- Lessons:
  - Optimal decisions depend not only on each client's own payoffs
  - Low-payoff servers are often avoided even if they are available

Performance Metric: Regret

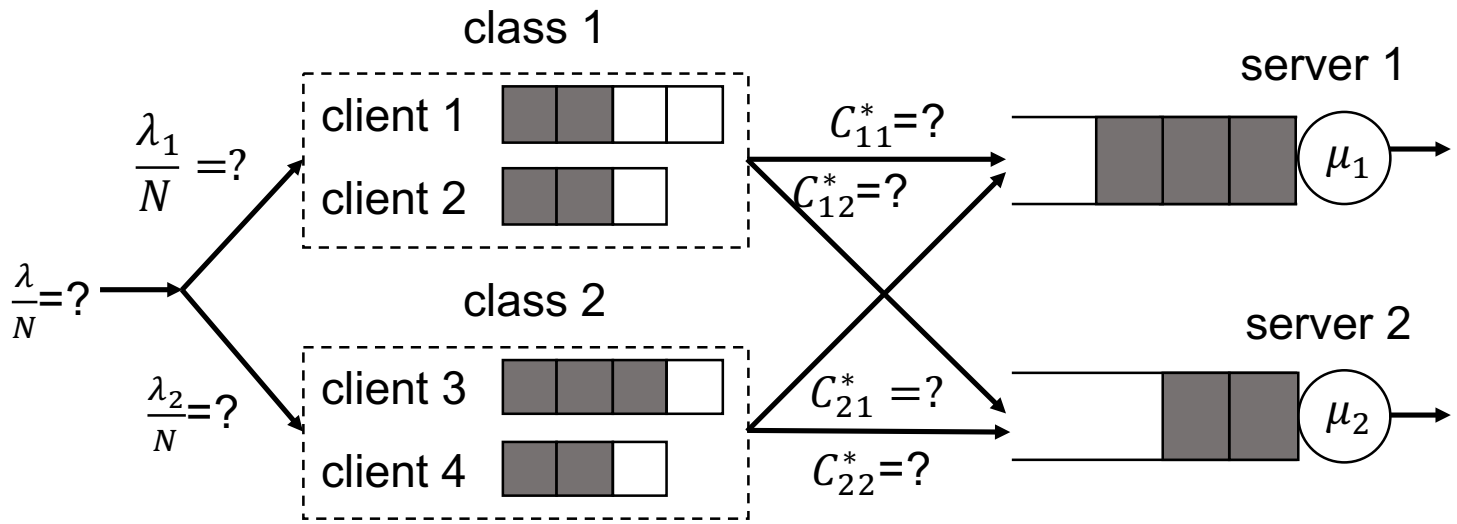


- Expected payoff per unit time of a policy Π:

$$R_T(\Pi) = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{J}\mathbb{E}\left[\sum_{l=1}^{n(t)} p_j^l(t) C_{i(l),j}^*\right]$$

  - $n(t)$: number of clients in the system at t
  - $p_j^l(t)$: for client l, the mean # of tasks assigned to server j at time t

Performance Metric: Regret

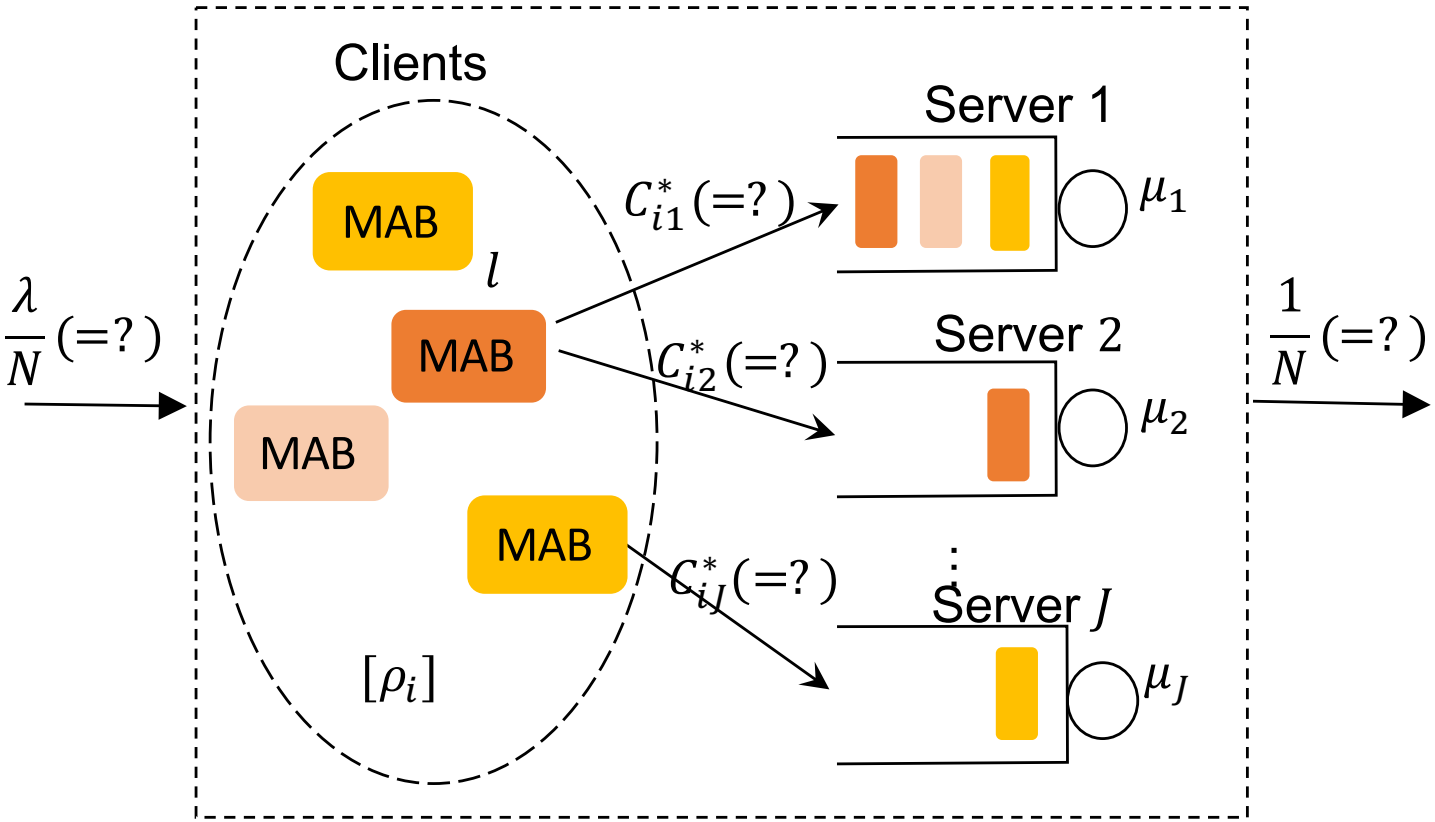


- Expected payoff per unit time of a policy Π:

$$R_T(\Pi) = \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{J} \mathbb{E} \left[\sum_{l=1}^{n(t)} p_j^l(t) C_{i(l),j}^* \right]$$

  - $n(t)$: number of clients in the system at t
  - $p_j^l(t)$: for client l, the mean # of tasks assigned to server j at time t
- Regret: $R^* - R_T(\Pi)$

As a Dynamic Multi-arm Bandit (MAB) Problem



- Each client is associated with a MAB problem
- Each server corresponds to an arm
- Must balance queueing and learning processes among clients

Outline of the Talk

- System Model
- Two Failed Attempts
  - Attempt #1: Myopic Matching + UCB
  - Attempt #2: Queue-Based Control + UCB
- Utility-guided Learning and Task Assignment Policy
- Intuitions behind the Main Results
- Conclusions and Remarks

Failed Attempt #1: Myopic Matching + UCB

- Use Upper confidence Bound (UCB) estimates for payoff:

$$C_j^l(t) = \min\left\{\overline{C}_j^l(t-1) + \sqrt{\frac{2\log h^l(t-1)}{h_j^l(t-1)}},\ 1\right\}$$

where for each client l and server j

- $\overline{C}_j^l(t-1)$: empirical average payoff of client l assigned to server j
- $h_j^l(t-1)$: # of tasks from client l assigned to server j
- $h^l(t-1) = \sum_j h_j^l(t-1)$: total # of tasks from client l assigned

Failed Attempt #1: Myopic Matching + UCB

- Use Upper confidence Bound (UCB) estimates for payoff:

$$C_j^l(t) = \min\left\{\overline{C}_j^l(t-1) + \sqrt{\frac{2\log h^l(t-1)}{h_j^l(t-1)}},\ 1\right\}$$

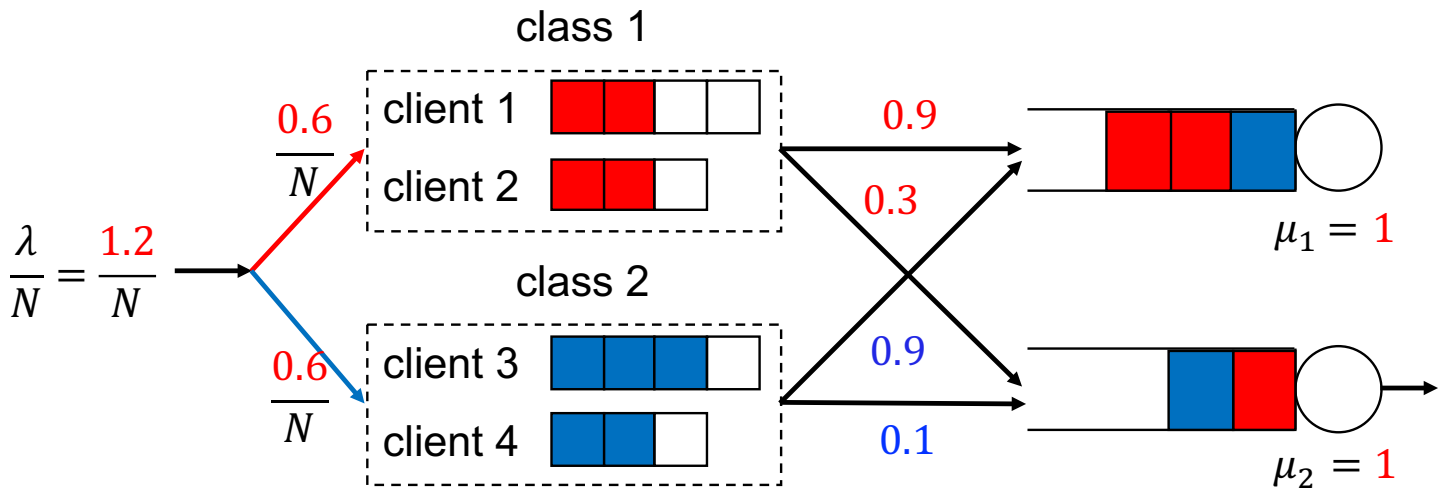
  where for each client l and server j
  - $\overline{C}_j^l(t-1)$: empirical average payoff of client l assigned to server j
  - $h_j^l(t-1)$: # of tasks from client l assigned to server j
  - $h^l(t-1) = \sum_j h_j^l(t-1)$: total # of tasks from client l assigned

- Myopic Matching: maximize instantaneous total payoff

$$\max_{[p_j^l]\geq 0} \quad \sum_{l=1}^{n(t)}\sum_{j=1}^{J} p_j^l\, C_j^l(t)$$

$$\text{s.t.} \quad \sum_{l=1}^{n(t)} p_j^l \leq \mu_j \quad \text{for all servers } j = 1,\ldots,J$$

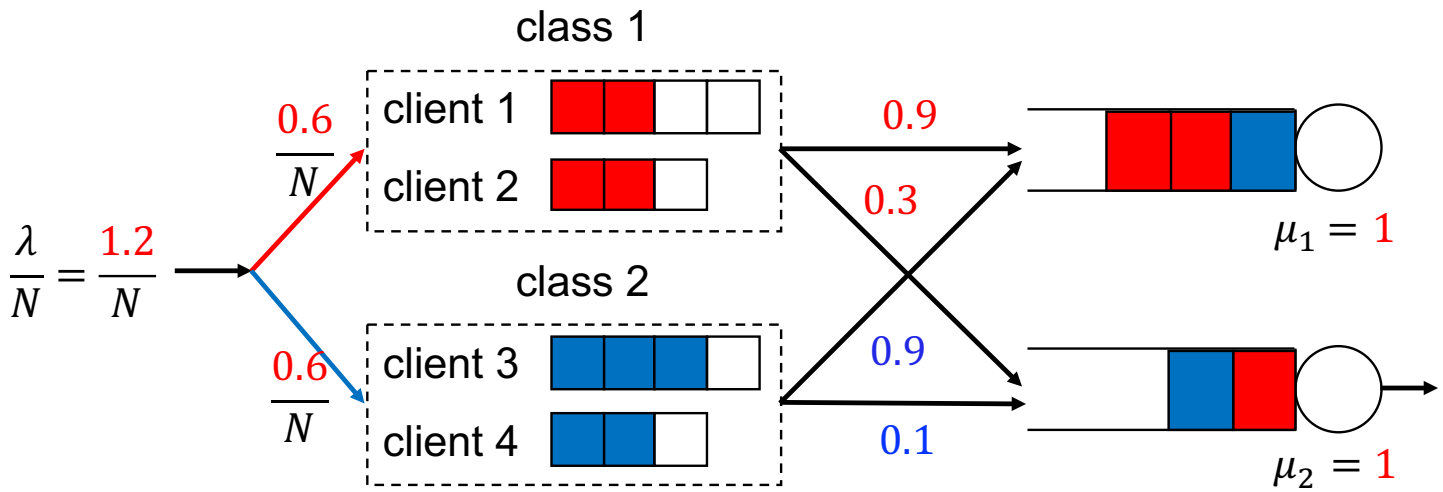
But... Myopic Matching is Suboptimal



- Recall the optimal solution: $p_{11}^* = \frac{2}{3},\ p_{12}^* = \frac{1}{3},\ p_{21}^* = 1,\ p_{22}^* = 0$

$$R^* = 0.4 \times 0.9 + 0.2 \times 0.3 + 0.6 \times 0.9 + 0 \times 0.1 = 0.96$$

But... Myopic Matching is Suboptimal



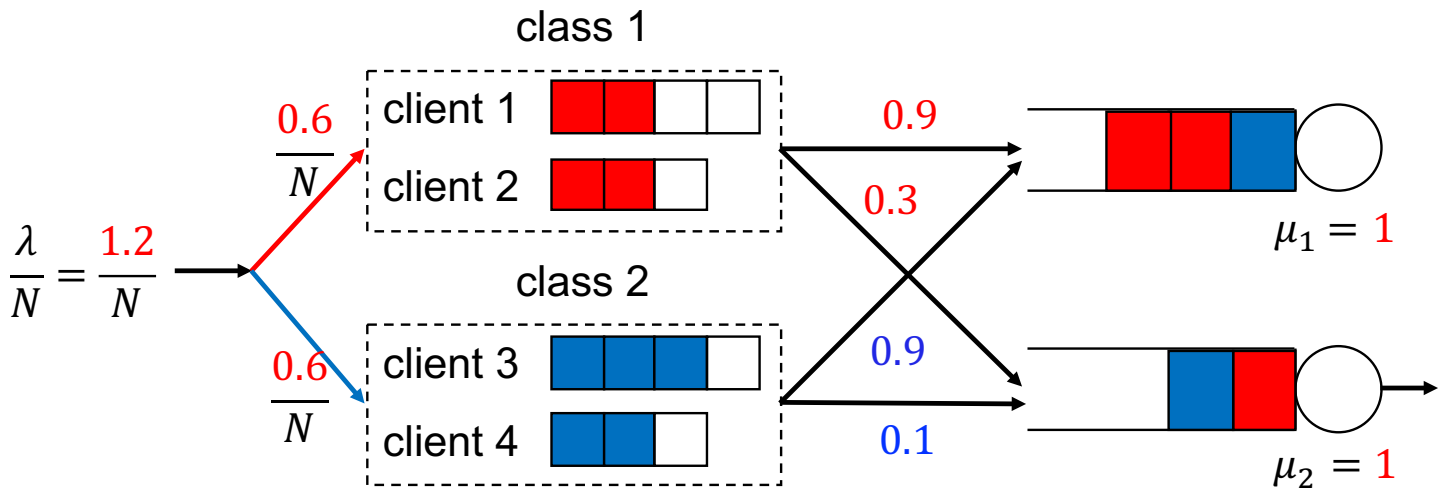
- Recall the optimal solution: $p^*_{11} = \frac{2}{3},\ p^*_{12} = \frac{1}{3},\ p^*_{21} = 1,\ p^*_{22} = 0$

$$R^* = 0.4 \times 0.9 + 0.2 \times 0.3 + 0.6 \times 0.9 + 0 \times 0.1 = 0.96$$

- Myopic matching: Whenever there are at least 2 tasks, one task will be assigned to server 2. The overall payoff is no more than

$$1.2 \times \frac{0.9 + 0.3}{2} = 0.72$$

But... Myopic Matching is Suboptimal



- Recall the optimal solution: $p^*_{11} = \frac{2}{3},\ p^*_{12} = \frac{1}{3},\ p^*_{21} = 1,\ p^*_{22} = 0$

$$R^* = 0.4 \times 0.9 + 0.2 \times 0.3 + 0.6 \times 0.9 + 0 \times 0.1 = 0.96$$

- Myopic matching: Whenever there are at least 2 tasks, one task will be assigned to server 2. The overall payoff is no more than

$$1.2 \times \frac{0.9 + 0.3}{2} = 0.72$$

- Lesson: must consider not only the current payoff, but also the long-term dynamics!

Failed Attempt #2: Queue-Based Control + UCB

Starting from the oracle upper bound:

$$R^* = \max_{[p_{ij}] \geq 0} \quad \sum_{i=1}^{I} \lambda_i \sum_{j=1}^{J} p_{ij} C^*_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \lambda_i p_{ij} \leq \mu_j \text{ for all servers } j = 1, \ldots, J$$

$$\sum_{j=1}^{J} p_{ij} = 1 \text{ for all classes } i = 1, \ldots, I$$

Failed Attempt #2: Queue-Based Control + UCB

Starting from the oracle upper bound:

$$R^* = \max_{[p_{ij}] \geq 0} \quad \sum_{i=1}^{I} \lambda_i \sum_{j=1}^{J} p_{ij} C^*_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \lambda_i p_{ij} \leq \mu_j \text{ for all servers } j = 1, \ldots, J \text{ shadow price } q_j$$

$$\sum_{j=1}^{J} p_{ij} = 1 \text{ for all classes } i = 1, \ldots, I$$

- At each time, assign 1 task from each class i to the server whose adjusted payoff $C^*_{ij} - q_j$ is the highest (among all servers)
- Update the dual variables

$$q_j(t+1) = \left[q_j(t) + \frac{1}{V} \left(\sum_l p^l_j(t) - \mu_j \right) \right]^+$$

Failed Attempt #2: Queue-Based Control + UCB

Starting from the oracle upper bound:

$$R^* = \max_{[p_{ij}] \geq 0} \quad \sum_{i=1}^{I} \lambda_i \sum_{j=1}^{J} p_{ij} C_{ij}^*$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \lambda_i p_{ij} \leq \mu_j \text{ for all servers } j = 1, \ldots, J \text{ shadow price } q_j$$

$$\sum_{j=1}^{J} p_{ij} = 1 \text{ for all classes } i = 1, \ldots, I$$

- At each time, assign 1 task from each client l to the server whose adjusted payoff $C_j^l(t) - q_j(t)$ is the highest (among all servers)
- Update the dual variables

$$q_j(t+1) = \left[q_j(t) + \frac{1}{V} \left(\sum_l p_j^l(t) - \mu_j \right) \right]^+$$

But... Queueing Delays Learning

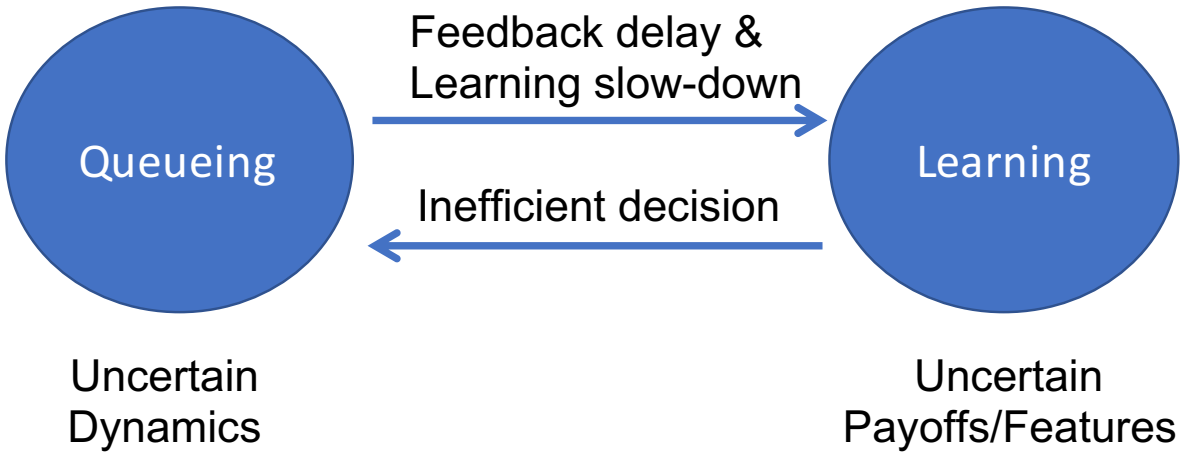
$$q_j(t+1) = \left[q_j(t) + \frac{1}{V} \left(\sum_l p_j^l(t) - \mu_j \right) \right]^+$$

- The dual variable $= 1/V \times$ the server queue length
- V increases $\Rightarrow$ server queue increases $\Rightarrow$ delays the feedback for learning!

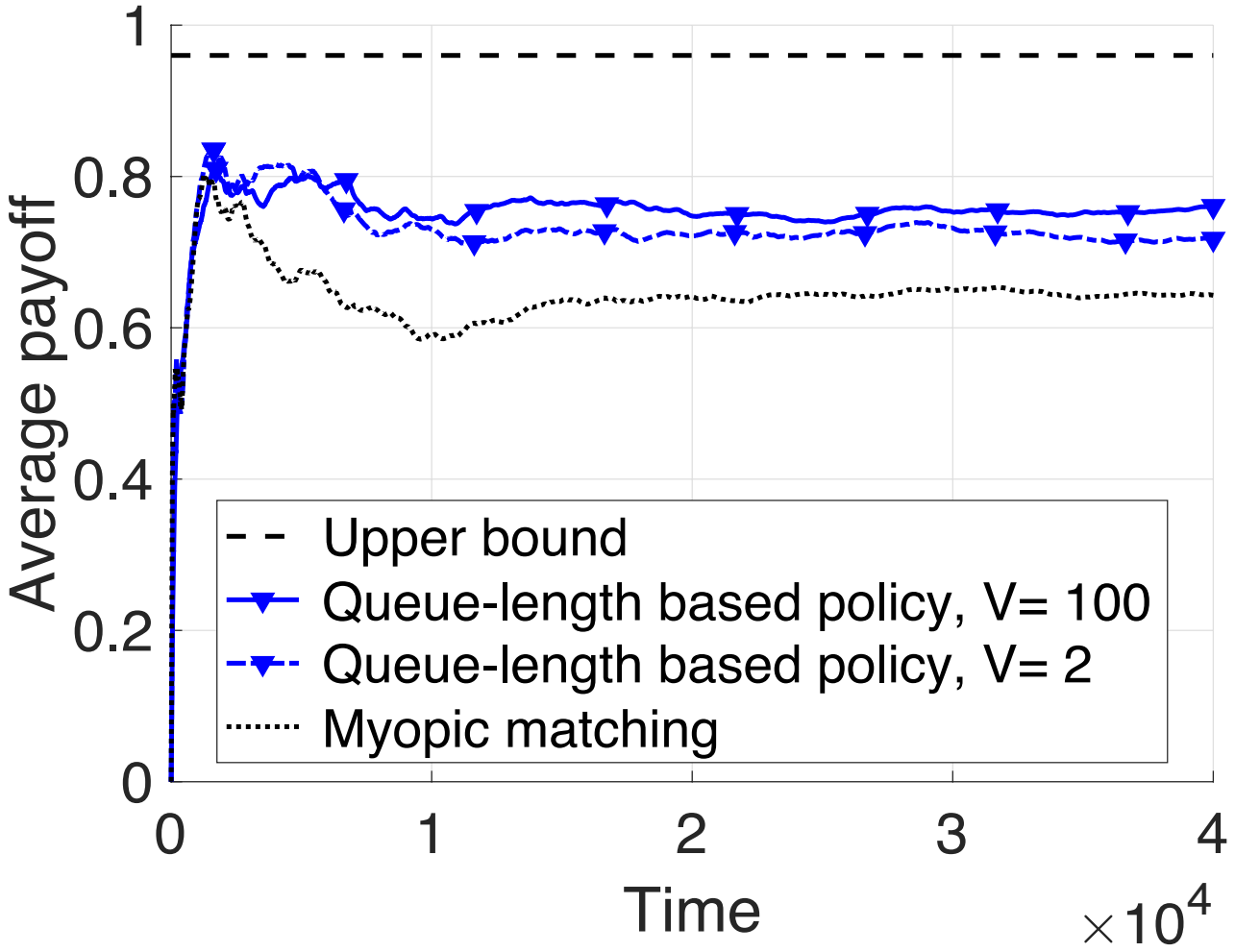
But... Queueing Delays Learning

$$q_j(t+1) = \left[q_j(t) + \frac{1}{V} \left(\sum_l p_j^l(t) - \mu_j \right) \right]^+$$

- The dual variable $= 1/V \times$ the server queue length
- V increases $\Rightarrow$ server queue increases $\Rightarrow$ delays the feedback for learning!
- A vicious cycle!



Numerical Results: Performance Comparison

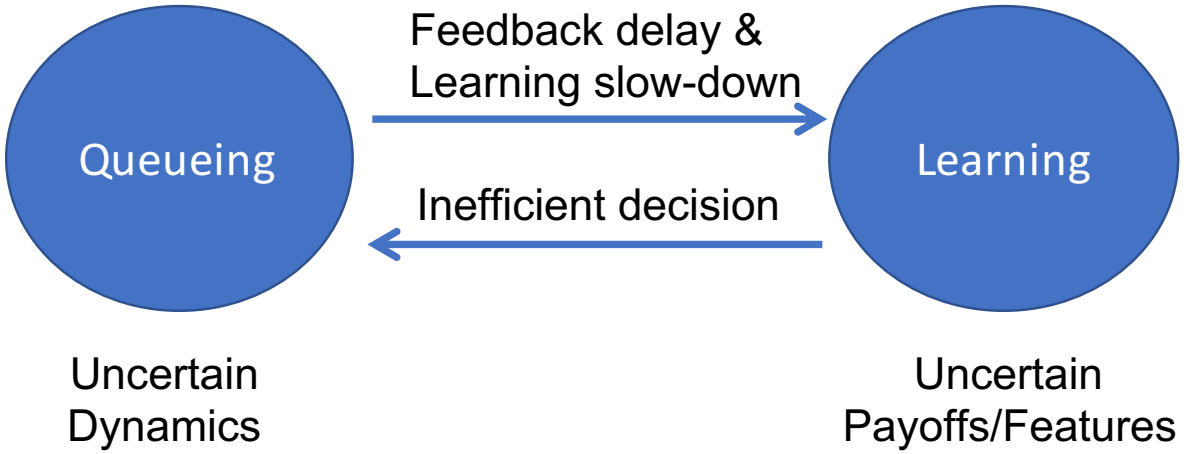


$N = 100$ and $\gamma = 1.1$ and $R^* = 0.96$

Outline of the Talk

- System Model
- Two Failed Attempts
- Utility-guided Learning and Task Assignment Policy
- Intuitions behind the Main Results
- Conclusions and Remarks

Towards the New Policy



- Myopic decisions do not work $\Rightarrow$ Must use queues to learn the long-term contention levels
- Server queues delay learning $\Rightarrow$ Cannot use server queues to infer contention levels
- A dilemma?

Utility-guided Online Learning Algorithm

- Use UCB-style estimate for payoff

$$C_j^l(t) = \min\left\{\overline{C}_j^l(t-1) + \sqrt{\frac{2\log h^l(t-1)}{h_j^l(t-1)}},\ 1\right\}$$

- Utility-guided task assignment:

$$\max_{[p_j^l]\geq 0} \quad \sum_{l=1}^{n(t)}\left\{\frac{1}{V}\log\left(\sum_{j=1}^{J} p_j^l\right) + \sum_{j=1}^{J} p_j^l (C_j^l(t)-\gamma)\right\}$$

$$\text{s.t.} \quad \sum_{l=1}^{n(t)} p_j^l \leq \mu_j \quad \text{for all servers } j = 1,\ldots,J\ .$$

  - Assign on average p_j^l tasks from client l to server j
  - Does not need to know λ_i, N, and payoff vectors (even the finite class assumption is not needed)!

Intuition Behind the Proposed Algorithm

$$\max_{[p_j^l] \geq 0} \quad \sum_{l=1}^{n(t)} \left\{ \frac{1}{V} \log \left(\sum_{j=1}^{J} p_j^l \right) + \sum_{j=1}^{J} p_j^l (C_j^l(t) - \gamma) \right\}$$

$$\text{s.t.} \quad \sum_{l=1}^{n(t)} p_j^l \leq \mu_j \quad \text{for all servers } j = 1, \ldots, J \ .$$

- Respect server constraint at each time $\Rightarrow$ no server queues or feedback delay!

Intuition Behind the Proposed Algorithm

$$\max_{[p_j^l]\geq 0} \quad \sum_{l=1}^{n(t)} \left\{ \frac{1}{V} \log\left(\sum_{j=1}^{J} p_j^l\right) + \sum_{j=1}^{J} p_j^l (C_j^l(t) - \gamma) \right\}$$

$$\text{s.t.} \quad \sum_{l=1}^{n(t)} p_j^l \leq \mu_j \quad \text{for all servers } j = 1, \ldots, J \ .$$

- Respect server constraint at each time $\Rightarrow$ no server queues or feedback delay!
- Log utility function promotes fairness
  - Fairness has an eye for the future, which is inspired by flow-level congestion control in communication networks [Bonald & Massoulie '01]
  - Fairness ensures that all clients can learn/explore

Intuition Behind the Proposed Algorithm

$$\max_{[p_j^l]\geq 0} \quad \sum_{l=1}^{n(t)} \left\{ \frac{1}{V} \log\left(\sum_{j=1}^{J} p_j^l\right) + \sum_{j=1}^{J} p_j^l \left(C_j^l(t) - \gamma\right) \right\}$$

$$\text{s.t.} \quad \sum_{l=1}^{n(t)} p_j^l \leq \mu_j \quad \text{for all servers } j = 1, \ldots, J \ .$$

- Truncate the UCB estimates by 1
- $\gamma > 1$: prevent assigning clients to low-payoff servers too aggressively

$$\frac{\partial f}{\partial p_j^l} = \frac{1}{V \sum_{j=1}^{J} p_j^l} + \left(C_j^l(t) - \gamma\right)$$

Intuition Behind the Proposed Algorithm

$$\max_{[p_j^l]\geq 0} \quad \sum_{l=1}^{n(t)} \left\{ \frac{1}{V} \log\left(\sum_{j=1}^{J} p_j^l\right) + \sum_{j=1}^{J} p_j^l \left(C_j^l(t) - \gamma\right) \right\}$$

$$\text{s.t.} \quad \sum_{l=1}^{n(t)} p_j^l \leq \mu_j \quad \text{for all servers } j = 1, \ldots, J \ .$$

- Truncate the UCB estimates by 1
- $\gamma > 1$: prevent assigning clients to low-payoff servers too aggressively

$$\frac{\partial f}{\partial p_j^l} = \frac{1}{V \sum_{j=1}^{J} p_j^l} + \left(C_j^l(t) - \gamma\right)$$

- $V > 0$: as V increases
  - more conservative in assigning clients to low-payoff servers
  - more congestion/backlog

Main Result #1: Bound on Number of Backlogged Clients

- $n(t)$: number of clients in the system at time t
- λ: total task arrival rate. $\mu = \sum_j \mu_j$: total service rate
- Assume $\lambda < \mu$

Theorem (mean number of backlogged clients)

$$\mathbb{E}\left[n(t)\right] \leq \frac{2\mu}{\mu - \lambda}\left(1 + \frac{\mu^2\gamma}{\gamma - 1}\right) + \mu\gamma V$$

Main Result #1: Bound on Number of Backlogged Clients

- $n(t)$: number of clients in the system at time t
- λ: total task arrival rate. $\mu = \sum_j \mu_j$: total service rate
- Assume $\lambda < \mu$

Theorem (mean number of backlogged clients)

$$\mathbb{E}\left[n(t)\right] \leq \frac{2\mu}{\mu - \lambda}\left(1 + \frac{\mu^2\gamma}{\gamma - 1}\right) + \mu\gamma V$$

- implies system is stable
- mean number of backlogged clients increase linearly in V
- Proof ideas:
  - when $n(t)$ is large, all servers will be fully utilized
  - couple to a Geom/Geom/μ queue with Bernoulli arrivals and Binomial departures

Main Result #2: Payoff Regret

Theorem

For any time duration T, the gap between the expected per-unit-time payoff and the oracle upper-bound is no greater than

$$R^* - R_T \leq \frac{\beta_1}{V} + \beta_2 \sqrt{\frac{\log N}{N}} + \beta_3 \frac{N(V+1)}{T},$$

where $\beta_1, \beta_2, \beta_3$ are functions of $I, J, \lambda_i, \mu_j, \gamma$.

Main Result #2: Payoff Regret

Theorem

For any time duration T, the gap between the expected per-unit-time payoff and the oracle upper-bound is no greater than

$$R^* - R_T \leq \underbrace{\frac{\beta_1}{V}}_{\text{uncertain dynamics}} + \beta_2 \sqrt{\frac{\log N}{N}} + \beta_3 \frac{N(V+1)}{T},$$

where $\beta_1, \beta_2, \beta_3$ are functions of $I, J, \lambda_i, \mu_j, \gamma$.

Main Result #2: Payoff Regret

Theorem

For any time duration T, the gap between the expected per-unit-time payoff and the oracle upper-bound is no greater than

$$R^* - R_T \leq \underbrace{\frac{\beta_1}{V}}_{\text{uncertain dynamics}} + \underbrace{\beta_2\sqrt{\frac{\log N}{N}}}_{\text{uncertain payoff}} + \beta_3\frac{N(V+1)}{T},$$

where $\beta_1, \beta_2, \beta_3$ are functions of $I, J, \lambda_i, \mu_j, \gamma$.

Main Result #2: Payoff Regret

Theorem

For any time duration T, the gap between the expected per-unit-time payoff and the oracle upper-bound is no greater than

$$R^* - R_T \leq \underbrace{\frac{\beta_1}{V}}_{\text{uncertain dynamics}} + \underbrace{\beta_2\sqrt{\frac{\log N}{N}}}_{\text{uncertain payoff}} + \underbrace{\beta_3\frac{N(V+1)}{T}}_{\text{backlogged tasks}},$$

where $\beta_1, \beta_2, \beta_3$ are functions of $I, J, \lambda_i, \mu_j, \gamma$.

Main Result #2: Payoff Regret

Theorem

For any time duration T, the gap between the expected per-unit-time payoff and the oracle upper-bound is no greater than

$$R^* - R_T \leq \underbrace{\frac{\beta_1}{V}}_{\text{uncertain dynamics}} + \underbrace{\beta_2\sqrt{\frac{\log N}{N}}}_{\text{uncertain payoff}} + \underbrace{\beta_3\frac{N(V+1)}{T}}_{\text{backlogged tasks}},$$

where $\beta_1, \beta_2, \beta_3$ are functions of $I, J, \lambda_i, \mu_j, \gamma$.

- The result holds for any finite time T!

Main Result #2: Payoff Regret

Theorem

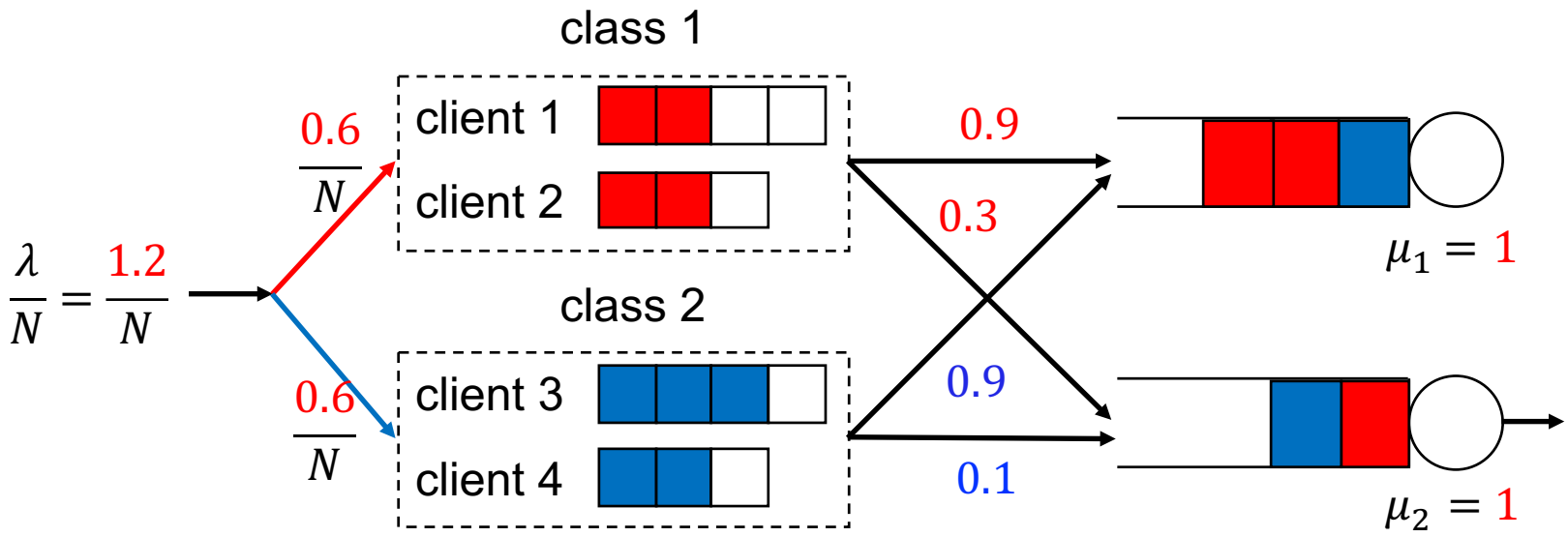
For any time duration T, the gap between the expected per-unit-time payoff and the oracle upper-bound is no greater than

$$R^* - R_T \leq \underbrace{\frac{\beta_1}{V}}_{\text{uncertain dynamics}} + \underbrace{\beta_2\sqrt{\frac{\log N}{N}}}_{\text{uncertain payoff}} + \underbrace{\beta_3\frac{N(V+1)}{T}}_{\text{backlogged tasks}},$$

where $\beta_1, \beta_2, \beta_3$ are functions of $I, J, \lambda_i, \mu_j, \gamma$.

- The result holds for any finite time T!
- $\sqrt{\frac{\log N}{N}}$ regret holds for any payoff vector distribution with finite support $\Rightarrow$ close to the optimal instance-independent regret $\frac{1}{\sqrt{N}}$ [Auer-Cesa-Bianchi-Freund-Schapire '98]

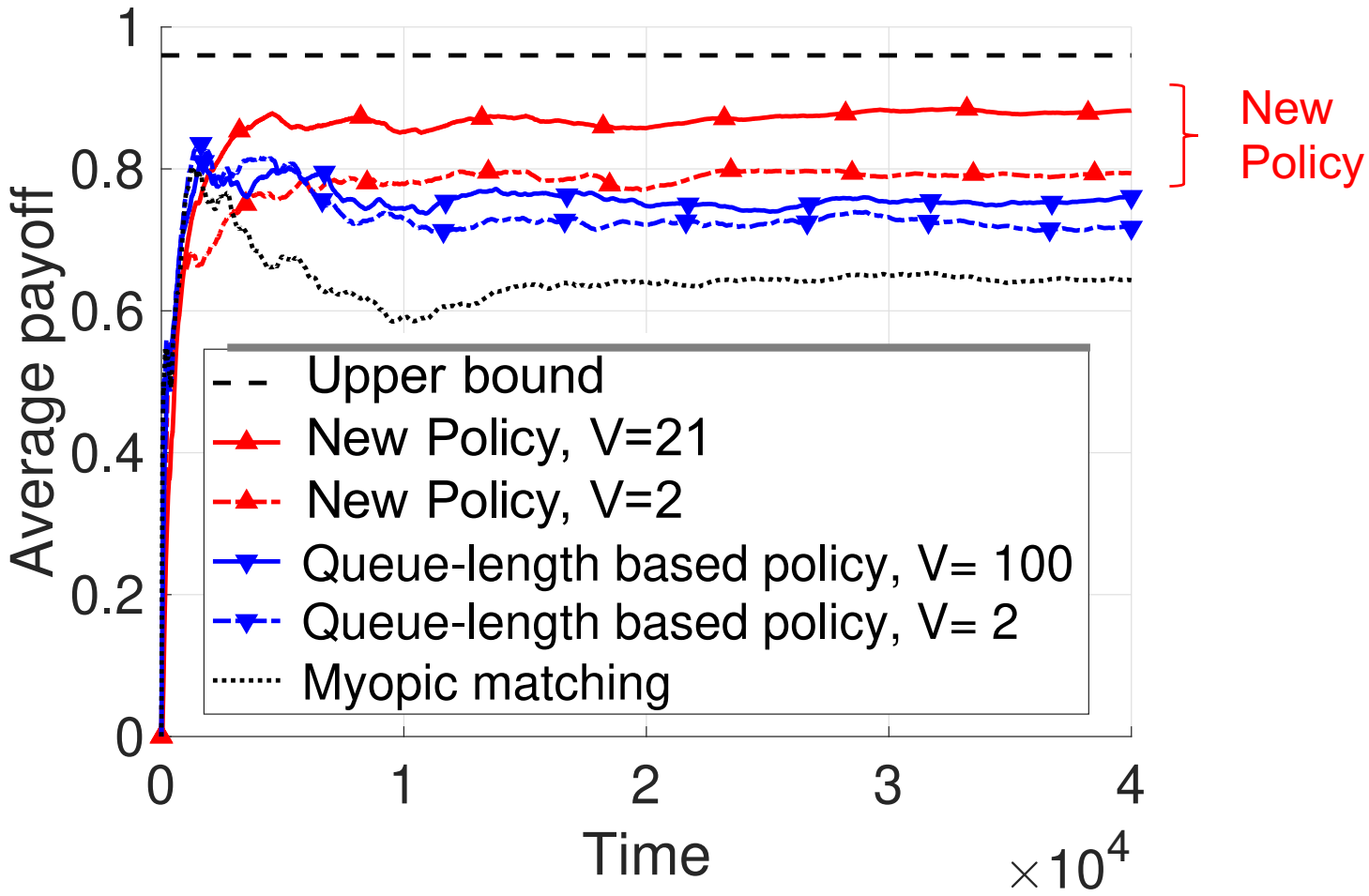
Performance Comparison



Optimal solution:

$$p_{11}^* = \frac{2}{3}, \quad p_{12}^* = \frac{1}{3}, \quad p_{21}^* = 1, \quad p_{22}^* = 0, \quad R^* = 0.96$$

Numerical Results: Performance Comparison

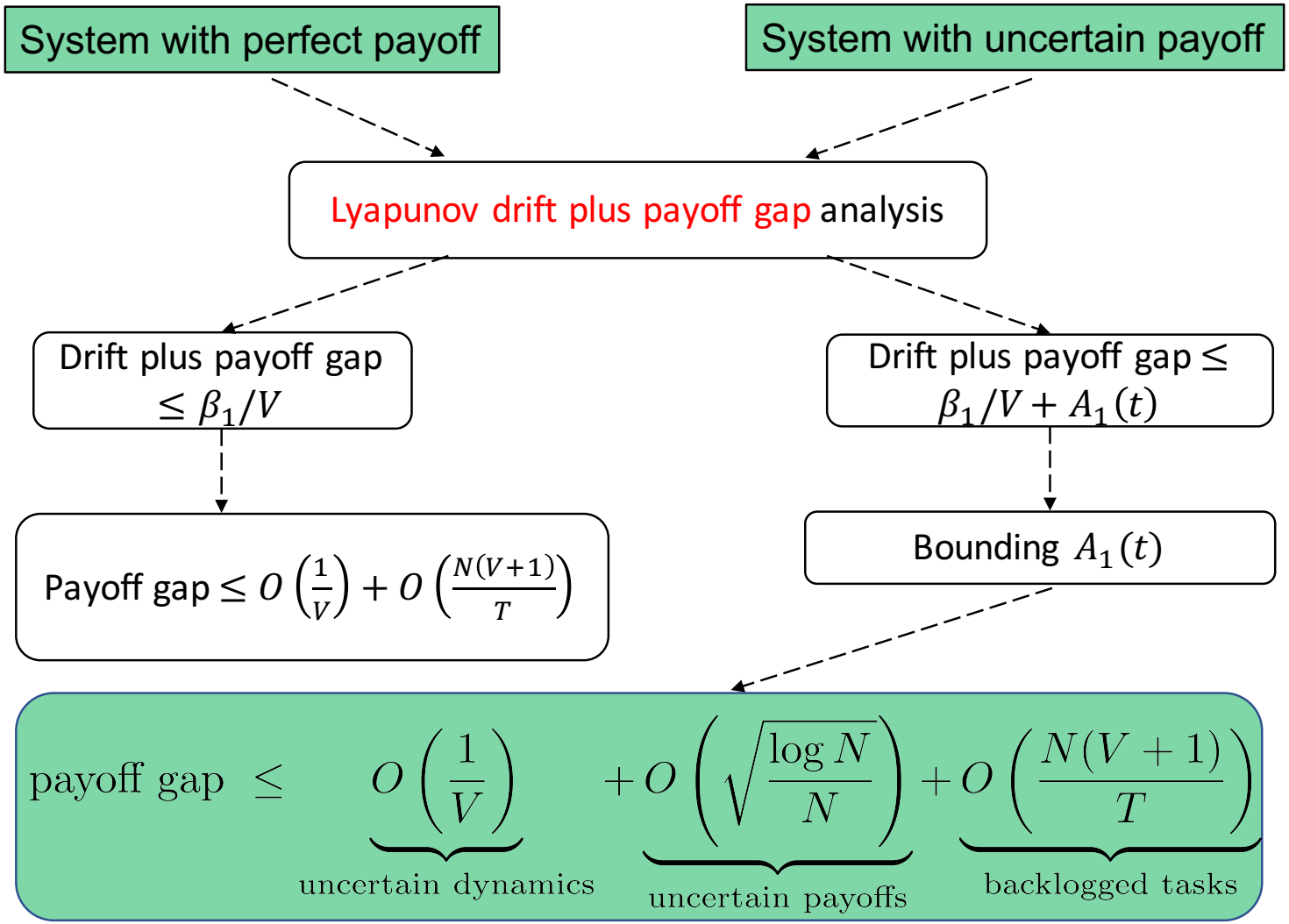


$N = 100$ and $\gamma = 1.1$ and $R^* = 0.96$

Outline of the Talk

- System Model
- Two Failed Attempts
- Utility-guided Learning and Task Assignment Policy
- Intuitions behind the Main Results
- Conclusions and Remarks

Proof Overview of Main Result #2 (Payoff Regret)



Step 1: Lyapunov Drift + Payoff Gap Analysis

$$L\left(\boldsymbol{n}(t)\right) = \frac{N}{2V}\sum_{i=1}^{I}\frac{n_i^2(t)}{\lambda_i}$$

$$\Delta(t) = \underbrace{\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i p_{ij}^*\left(C_{ij}^* - \gamma\right)}_{\text{adjusted oracle upper bound}} - \underbrace{\sum_{l=1}^{n(t)}\sum_{j=1}^{J}p_j^l(t)\left(C_{i(l),j}^* - \gamma\right)}_{\text{adjusted payoff of our policy}}$$

Step 1: Lyapunov Drift + Payoff Gap Analysis

$$L\left(\boldsymbol{n}(t)\right) = \frac{N}{2V}\sum_{i=1}^{I}\frac{n_i^2(t)}{\lambda_i}$$

$$\Delta(t) = \underbrace{\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i p_{ij}^*\left(C_{ij}^* - \gamma\right)}_{\text{adjusted oracle upper bound}} - \underbrace{\sum_{l=1}^{n(t)}\sum_{j=1}^{J}p_j^l(t)\left(C_{i(l),j}^* - \gamma\right)}_{\text{adjusted payoff of our policy}}$$

Lemma (Expected Lyapunov drift plus payoff gap is bounded)

$$\mathbb{E}\left[L\left(\boldsymbol{n}(t+1)\right) - L\left(\boldsymbol{n}(t)\right) + \Delta(t) \mid \boldsymbol{n}(t), \boldsymbol{p}(t)\right] \leq \frac{\beta_1}{V} + A_1(t)$$

Step 1: Lyapunov Drift + Payoff Gap Analysis

$$L\left(\boldsymbol{n}(t)\right) = \frac{N}{2V}\sum_{i=1}^{I}\frac{n_i^2(t)}{\lambda_i}$$

$$\Delta(t) = \underbrace{\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i p_{ij}^*\left(C_{ij}^* - \gamma\right)}_{\text{adjusted oracle upper bound}} - \underbrace{\sum_{l=1}^{n(t)}\sum_{j=1}^{J}p_j^l(t)\left(C_{i(l),j}^* - \gamma\right)}_{\text{adjusted payoff of our policy}}$$

Lemma (Expected Lyapunov drift plus payoff gap is bounded)

$$\mathbb{E}\left[L\left(\boldsymbol{n}(t+1)\right) - L\left(\boldsymbol{n}(t)\right) + \Delta(t) \mid \boldsymbol{n}(t), \boldsymbol{p}(t)\right] \leq \frac{\beta_1}{V} + A_1(t)$$

Taking telescope sum and using $L(\boldsymbol{n}(0)) = 0$ and $L(\boldsymbol{n}(T)) \geq 0 \Rightarrow$

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\Delta(t)\right] \leq \frac{\beta_1}{V} + \frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[A_1(t)\right]$$

Step 1: Lyapunov Drift + Payoff Gap Analysis

$$L\left(\boldsymbol{n}(t)\right) = \frac{N}{2V}\sum_{i=1}^{I}\frac{n_i^2(t)}{\lambda_i}$$

$$\Delta(t) = \underbrace{\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i p_{ij}^*\left(C_{ij}^* - \gamma\right)}_{\text{adjusted oracle upper bound}} - \underbrace{\sum_{l=1}^{n(t)}\sum_{j=1}^{J}p_j^l(t)\left(C_{i(l),j}^* - \gamma\right)}_{\text{adjusted payoff of our policy}}$$

Lemma (Expected Lyapunov drift plus payoff gap is bounded)

$$\mathbb{E}\left[L\left(\boldsymbol{n}(t+1)\right) - L\left(\boldsymbol{n}(t)\right) + \Delta(t) \mid \boldsymbol{n}(t), \boldsymbol{p}(t)\right] \leq \frac{\beta_1}{V} + A_1(t)$$

Taking telescope sum and using $L(\boldsymbol{n}(0)) = 0$ and $L(\boldsymbol{n}(T)) \geq 0 \Rightarrow$

$$R^* - R_T \leq \frac{\beta_1}{V} + \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[A_1(t)\right] + \frac{\gamma N}{T}\mathbb{E}\left[n(T)\right]$$

Step 2: Bounding Regret due to Inaccurate Payoff

$$A_1(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \underbrace{\left(C_j^l(t) - C_{i(l),j}^*\right)}_{\text{learning error}} \underbrace{\left(p_j^l(t) - \widehat{p}_j^l(t)\right)}_{\text{controlling error}}$$

- $C_j^l(t)$: UCB payoff estimate for client l at server j
- $p_j^l(t)$: Assignment of our policy with UCB estimates
- $\widehat{p}_j^l(t)$: Assignment of our policy if the perfect payoffs were known

Step 2: Bounding Regret due to Inaccurate Payoff

$$A_1(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \underbrace{\left(C_j^l(t) - C^*_{i(l),j}\right)}_{\text{learning error}} \underbrace{\left(p_j^l(t) - \widehat{p}_j^l(t)\right)}_{\text{controlling error}}$$

$$= \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C_j^l(t) - C^*_{i(l),j}\right) p_j^l(t) \quad \longrightarrow A_2(t)$$

$$+ \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C^*_{i(l),j} - C_j^l(t)\right) \widehat{p}_j^l(t) \quad \longrightarrow A_3(t)$$

- $C_j^l(t)$: UCB payoff estimate for client l at server j
- $p_j^l(t)$: Assignment of our policy with UCB estimates
- $\widehat{p}_j^l(t)$: Assignment of our policy if the perfect payoffs were known

Step 2(a): Bounding $A_2(t)$ due to Learning Error

$$A_2(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t)$$

- The UCB estimate $C_j^l(t)$ is likely to over-estimate $C^*_{i(l),j}$
- The over-estimate amount is likely to decrease as $p_j^l(t)$ accumulates

Step 2(a): Bounding $A_2(t)$ due to Learning Error

$$A_2(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t)$$

- The UCB estimate $C_j^l(t)$ is likely to over-estimate $C^*_{i(l),j}$
- The over-estimate amount is likely to decrease as $p_j^l(t)$ accumulates

$$\sum_t \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t) \overset{\text{UCB}}{\approx} \sum_t \frac{1}{\sqrt{h_j^l(t-1)}} p_j^l(t)$$

Step 2(a): Bounding $A_2(t)$ due to Learning Error

$$A_2(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t)$$

- The UCB estimate $C_j^l(t)$ is likely to over-estimate $C^*_{i(l),j}$
- The over-estimate amount is likely to decrease as $p_j^l(t)$ accumulates

$$\sum_t \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t) \overset{\text{UCB}}{\approx} \sum_t \frac{1}{\sqrt{h_j^l(t-1)}} p_j^l(t)$$

$$\overset{\text{Martingale}}{\approx} \sum_t \frac{1}{\sqrt{h_j^l(t-1)}} \left(h_j^l(t) - h_j^l(t-1) \right)$$

Step 2(a): Bounding $A_2(t)$ due to Learning Error

$$A_2(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t)$$

- The UCB estimate $C_j^l(t)$ is likely to over-estimate $C^*_{i(l),j}$
- The over-estimate amount is likely to decrease as $p_j^l(t)$ accumulates

$$\sum_t \left(C_j^l(t) - C^*_{i(l),j} \right) p_j^l(t) \overset{\text{UCB}}{\approx} \sum_t \frac{1}{\sqrt{h_j^l(t-1)}} p_j^l(t)$$

$$\overset{\text{Martingale}}{\approx} \sum_t \frac{1}{\sqrt{h_j^l(t-1)}} \left(h_j^l(t) - h_j^l(t-1) \right)$$

$$\approx \sum_{k=1}^{N} \frac{1}{\sqrt{k}} \approx \sqrt{N}$$

Step 2(b): Bounding $A_3(t)$ due to Controlling Error

$$A_3(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C^*_{i(l),j} - C^l_j(t) \right) \widehat{p}^l_j(t)$$

- Good news: $\left(C^*_{i(l),j} - C^l_j(t) \right)$ is likely to negative

Step 2(b): Bounding $A_3(t)$ due to Controlling Error

$$A_3(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C^*_{i(l),j} - C^l_j(t) \right) \widehat{p}^l_j(t)$$

- Good news: $\left(C^*_{i(l),j} - C^l_j(t) \right)$ is likely to negative
- The bad event happens w.p. at most $\left(h^l_j(t) \right)^{-4}$

Step 2(b): Bounding $A_3(t)$ due to Controlling Error

$$A_3(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C^*_{i(l),j} - C^l_j(t) \right) \widehat{p}^l_j(t)$$

- Good news: $\left(C^*_{i(l),j} - C^l_j(t) \right)$ is likely to negative
- The bad event happens w.p. at most $\left(h^l_j(t) \right)^{-4}$
- But, it is possible that $\widehat{p}^l_j(t)$ accumulates while $h^l_j(t)$ remains small

Step 2(b): Bounding $A_3(t)$ due to Controlling Error

$$A_3(t) = \sum_{l=1}^{n(t)} \sum_{j=1}^{J} \left(C^*_{i(l),j} - C^l_j(t) \right) \widehat{p}^l_j(t)$$

- Good news: $\left(C^*_{i(l),j} - C^l_j(t) \right)$ is likely to negative
- The bad event happens w.p. at most $\left(h^l_j(t) \right)^{-4}$
- But, it is possible that $\widehat{p}^l_j(t)$ accumulates while $h^l_j(t)$ remains small
- Fortunately, we can bound the difference between $p^l_j(t)$ and $\widehat{p}^l_j(t)$:

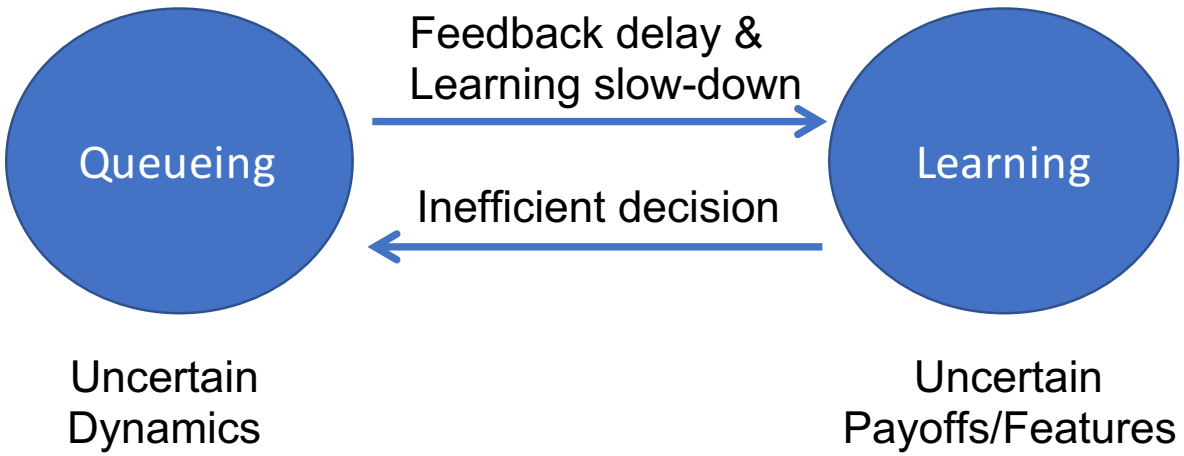
$$\frac{\sum_{j=1}^{J} \widehat{p}^l_j(t)}{\sum_{j=1}^{J} p^l_j(t)} \leq \left(\frac{\gamma}{\gamma - 1} \right)^2 \longleftarrow \text{Convex duality}$$

Thanks to fairness, no client's assignment probabilities are heavily skewed by UCB estimates

Outline of the Talk

- System Model
- Two Failed Attempts
- Utility-guided Learning and Task Assignment Policy
- Intuitions behind the Main Results
- Conclusions and Remarks

Summary



- Learning cannot ignore queueing dynamics: Myopic decisions may not produce good long-term performance
- Queueing cannot ignore learning either: Closed-loop interactions between them can degrade both learning and queueing performance
- A new utility-guided online learning and control algorithm achieves provably low finite-time regret

Extensions and Open Problems

Our policy based on convex program $\Rightarrow$ virtual-queue (dual) algorihtm

1. Server-side uncertainty: random service time with unknown mean
2. Decentralized decisions: each client makes her own decision

Extensions and Open Problems

Our policy based on convex program $\Rightarrow$ virtual-queue (dual) algorihtm

1. Server-side uncertainty: random service time with unknown mean
2. Decentralized decisions: each client makes her own decision

Open Problems

- The unknown payoff vector has a continuous distribution
- Regret lower-bound: $\sqrt{\frac{\log N}{N}}$ v.s. $\frac{\log N}{N}$
- A growing # of servers with additional structures (linear bandits)
- Other types of learning such as Thompson Sampling
- Combine offline and online learning

Extensions and Open Problems

Our policy based on convex program $\Rightarrow$ virtual-queue (dual) algorihtm

1. Server-side uncertainty: random service time with unknown mean
2. Decentralized decisions: each client makes her own decision

Open Problems

- The unknown payoff vector has a continuous distribution
- Regret lower-bound: $\sqrt{\frac{\log N}{N}}$ v.s. $\frac{\log N}{N}$
- A growing # of servers with additional structures (linear bandits)
- Other types of learning such as Thompson Sampling
- Combine offline and online learning

Reference

W.-K. Hsu, J. Xu, X. Lin, and M. R. Bell. *Integrate Learning and Control in Queueing Systems with Uncertain Paoyffs*, Information Theory and Applications Workshop, UC San Diego, CA, February 2018.

Further References on Learning + Queueing

1. Online matching while learning for payoff maximization: [Johari-Kamble-Kanoria '17]
   - Stationary setting with known arrival rates and payoff vectors
   - Assignment based on preliminary guessed labels and shadow prices
2. Learning unknown labels with capacity constraints: [Xu-Massoulié '16]
   - Focus on maximizing throughput with a target learning accuracy
   - Assignment based on preliminary guessed labels and current workloads at servers via "max-weight" type policy
3. Processing tasks of unknown types with capacity constraints: [Bimpikis-Markakis '15], [Shah-Gulikers-Massoulié-Vojnovic '17]
   - Focus on maximizing throughput
   - Either a small system with two types of clients and two servers, or needs an infinite # of queues in control (computationally expensive)