Competing Bandits in Non-Stationary Matching Markets

Avishek Ghosh

IIT Bombay

Joint work with

Abishek Sankararaman (Amazon)

Kannan Ramchandran (UC Berkeley)

Tara Javidi (UC San Diego)

Arya Mazumdar (UC San Diego)

RL Workshop, IISc Bangalore

Feb, 2024

Outline

- Matching Markets—a Multi Agent Multi-Armed Bandit formulation

Outline

- Matching Markets—a Multi Agent Multi-Armed Bandit formulation
- Competition—Collision—Resolution of conflicts

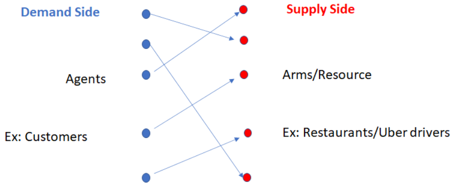
Outline

- Matching Markets—a Multi Agent Multi-Armed Bandit formulation
- Competition—Collision—Resolution of conflicts
- Dynamic (Non-Stationary) Markets
  - Algorithm for Non-Stationary Matching Markets
  - Insights for 2 agents
  - Analysis: Theoretical and Empirical

Matching Markets

Matching Markets

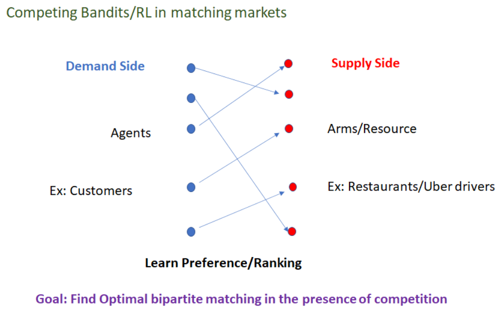
Competing Bandits/RL in matching markets



Learn Preference/Ranking

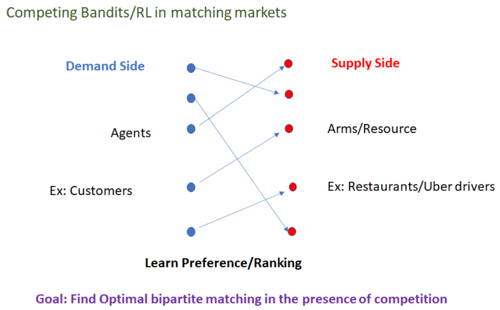
Goal: Find Optimal bipartite matching in the presence of competition

Matching Markets



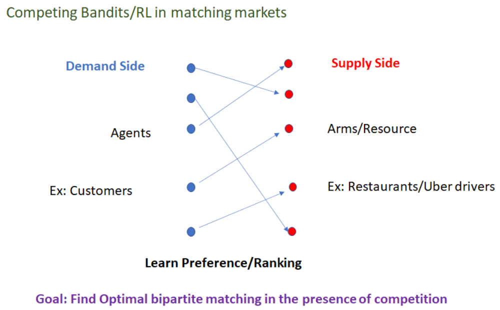
♣ Received a lot of interest in recent years (Liu et.al, 20, Johari et.al, 21, Sankararaman et. al, 21, Basu et.al, 21)

Matching Markets

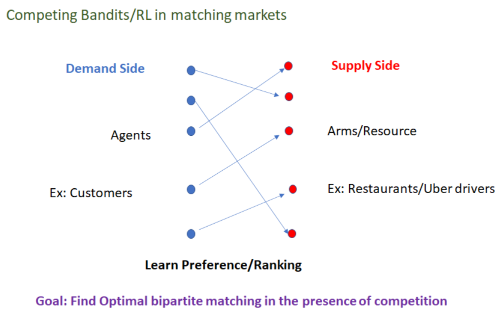


- Received a lot of interest in recent years (Liu et.al, 20, Johari et.al, 21, Sankararaman et. al, 21, Basu et.al, 21)
- Motivated by applications in Labor Market (ex. TaskRabbit, Upwork)
- College Admissions (classical motivation, Gale and Shapley, 1962)
- Matching medical interns/residents Dai and Jordan, 2021
- Objective: Match Demand to the Supply side

Matching Markets contd.

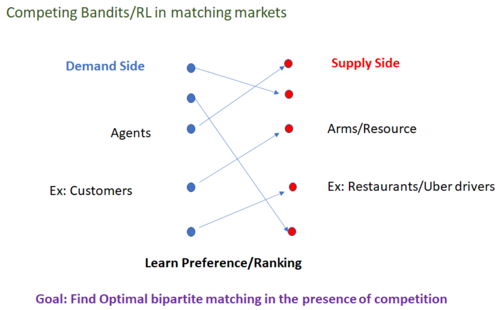


Matching Markets contd.



- Each Agent has a preference over Arms
- Each Arm has preference over Agents

Matching Markets contd.



- ♣ Each Agent has a preference over Arms
- ♣ Each Arm has preference over Agents
- ♣ If these preferences were known, a simple solution is Gale Shapley stable matching algorithm.
- ♣ However, we don't know these preferences—learn them via successive interactions between Agents and Arms

Model via Bandits framework

Model via Bandits framework

- Preference modeling via Bandits framework
- We consider N agents and K arms
  - When Agent i pulls arm j–receives a random reward with mean $\mu_{i,j}$
  - Agent i's preference—ordering of the arm means $\{\mu_{i,1}, \mu_{i,2}, \ldots, \mu_{i,k}\}$
  - Agent i is solving a Multi-Armed Bandit (MAB) problem

Model via Bandits framework

♣ Preference modeling via Bandits framework

♣ We consider N agents and K arms

- When Agent i pulls arm j–receives a random reward with mean $\mu_{i,j}$
- Agent i's preference—ordering of the arm means $\{\mu_{i,1}, \mu_{i,2}, \ldots, \mu_{i,k}\}$
- Agent i is solving a Multi-Armed Bandit (MAB) problem

♣ Solving multi-agent multi armed Bandit problem is equivalent to learning preferences

♣ Use tools from bandit literature—Liu et.al, 20, Jagadeesan et. al, 21, Johari et.al, 21, Sankararaman et. al, 21, Basu et.al, 21

Model via Bandits framework

♣ Preference modeling via Bandits framework

♣ We consider N agents and K arms

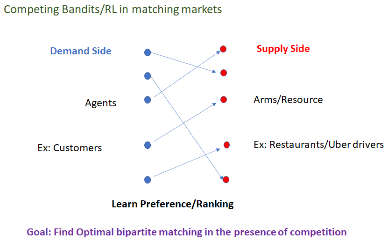
- When Agent i pulls arm j–receives a random reward with mean $\mu_{i,j}$
- Agent i's preference—ordering of the arm means $\{\mu_{i,1}, \mu_{i,2}, \ldots, \mu_{i,k}\}$
- Agent i is solving a Multi-Armed Bandit (MAB) problem

♣ Solving multi-agent multi armed Bandit problem is equivalent to learning preferences

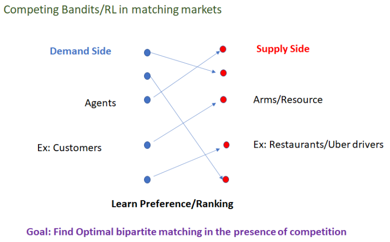
♣ Use tools from bandit literature—Liu et.al, 20, Jagadeesan et. al, 21, Johari et.al, 21, Sankararaman et. al, 21, Basu et.al, 21

♣ Caveat: There is competition in the system—2 or more agents may go for one arm—need to resolve collision/conflicts

One sided Learning and Conflict Resolution



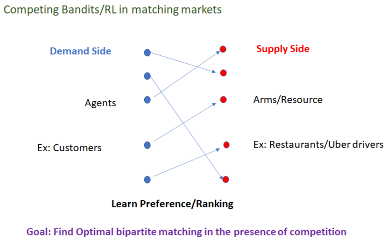
One sided Learning and Conflict Resolution



♣ Typical Assumption (Liu et.al, 20, Jagadeesan et. al, 21, Dai et. al '21, Sankararaman et. al, 21, Basu et.al, 21):

◀ Preferences of Agents (left) are known to the arms (right)–through some common knowledge—one sided learning

One sided Learning and Conflict Resolution

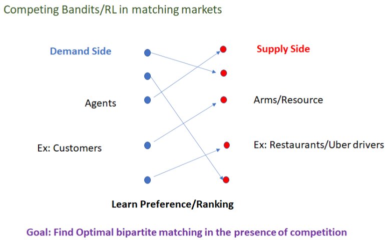


♣ Typical Assumption (Liu et.al, 20, Jagadeesan et. al, 21, Dai et. al '21, Sankararaman et. al, 21, Basu et.al, 21):

◀ Preferences of Agents (left) are known to the arms (right)–through some common knowledge—one sided learning

◀ Example: College Admissions; Agents are colleges—ranking of colleges are available

One sided Learning and Conflict Resolution



♣ Typical Assumption (Liu et.al, 20, Jagadeesan et. al, 21, Dai et. al '21, Sankararaman et. al, 21, Basu et.al, 21):

◀ Preferences of Agents (left) are known to the arms (right)–through some common knowledge—one sided learning

◀ Example: College Admissions; Agents are colleges—ranking of colleges are available

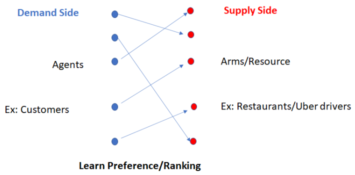
♣ Collision/Competition More than one agents pull same arm–Collision

♣ Reward is given to the Agent with highest rank among the competitors

♣ All other Agents receive 0 reward

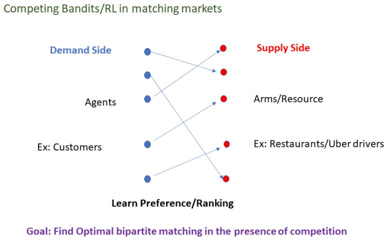
Even Simpler model–Serial Dictatorship

Competing Bandits/RL in matching markets



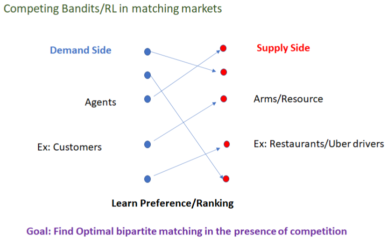
Goal: Find Optimal bipartite matching in the presence of competition

Even Simpler model–Serial Dictatorship



- ♣ Further Simplification (Sankararaman et. al, 21, Basu et.al, 21):
- ◀ Preferences of Agents (left) are same to all the Arms
- ◀ Example: College global ranking–same for all applicants

Even Simpler model–Serial Dictatorship



♣ Further Simplification (Sankararaman et. al, 21, Basu et.al, 21):

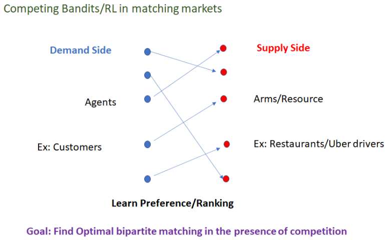
◀ Preferences of Agents (left) are same to all the Arms

◀ Example: College global ranking–same for all applicants

♣ Serial Dictatorship model is a popular in economics (Abdulkadirouglu and Somez, 1998)

♣ Relaxations to Serial Dictatorship to weaker models is also possible (Basu et. al, 21)

Even Simpler model–Serial Dictatorship



♣ Further Simplification (Sankararaman et. al, 21, Basu et.al, 21):

◀ Preferences of Agents (left) are same to all the Arms

◀ Example: College global ranking–same for all applicants

♣ Serial Dictatorship model is a popular in economics (Abdulkadirougl u and Somez, 1998)

♣ Relaxations to Serial Dictatorship to weaker models is also possible (Basu et. al, 21)

♣ Under Serial Dictatorship, the matching between Agents and Arms is unique—provides a reference point to characterize Regret

Regret

Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j

Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - Arm preferred by Agent 1:

$$\ell_*^{(1)} = \text{argmax}_{\ell \in [K]} \mu_{1,\ell}$$

Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - Arm preferred by Agent 1:

$$\ell_*^{(1)} = \mathrm{argmax}_{\ell \in [K]} \mu_{1,\ell}$$

  - Given collision structure—Arm preferred by Agent j is

$$\ell_*^{(j)} = \mathrm{argmax}_{\ell \in [K] \setminus \{\ell_*^{(1)},..,\ell_*^{(j-1)}\}} \mu_{j,\ell}$$

Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - Arm preferred by Agent 1:

$$\ell_*^{(1)} = \text{argmax}_{\ell \in [K]} \mu_{1,\ell}$$

  - Given collision structure—Arm preferred by Agent j is

$$\ell_*^{(j)} = \text{argmax}_{\ell \in [K] \setminus \{\ell_*^{(1)}, .., \ell_*^{(j-1)}\}} \mu_{j,\ell}$$

- $(1, \ell_*^{(1)}), (2, \ell_*^{(2)}), \ldots, (N, \ell_*^{(N)})$ forms a stable match–unique
- Objective: Minimize regret with respect to this unique stable match

Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - Arm preferred by Agent 1:

$$\ell_*^{(1)} = \mathrm{argmax}_{\ell \in [K]} \mu_{1,\ell}$$

  - Given collision structure—Arm preferred by Agent j is

$$\ell_*^{(j)} = \mathrm{argmax}_{\ell \in [K] \setminus \{\ell_*^{(1)}, .., \ell_*^{(j-1)}\}} \mu_{j,\ell}$$

- $(1, \ell_*^{(1)}), (2, \ell_*^{(2)}), \ldots, (N, \ell_*^{(N)})$ forms a stable match–unique
- Objective: Minimize regret with respect to this unique stable match
- Let $L^{(j)}$ be the arm played by an algorithm $\mathbb{A}$.
- Regret of agent j playing $\mathbb{A}$ is

$$R_j = \sum_{t=1}^{T} \mathbb{E}\left[\mu_{j,\ell_*^{(j)}} - \mu_{j,L^{(j)}} (\mathbf{1}_{L^{(j)} \text{matched player } j})\right]$$

UCB based algorithms

UCB based algorithms

♣ Assume one sided learning and serial dictatorship

♠ UCB-D3 by Sankararaman et. al 21, AISTATS (few extensions)

♠ Idea: Play UCB in a restrictive set of arms

♠ Avoid the arm preferred/most played by agents raked higher

UCB based algorithms

- ♣ Assume one sided learning and serial dictatorship
- ♠ UCB-D3 by Sankararaman et. al 21, AISTATS (few extensions)
- ♠ Idea: Play UCB in a restrictive set of arms
- ♠ Avoid the arm preferred/most played by agents raked higher
- ♠ Propose an epoch based method:
  - Agents estimate their rank first though collision
  - In each epoch, estimate the arms played/preferred by agents ranked higher through collisions
  - Remove them and play UCB on the restrictive set

UCB based algorithms

- ♣ Assume one sided learning and serial dictatorship
- ♠ UCB-D3 by Sankararaman et. al 21, AISTATS (few extensions)
- ♠ Idea: Play UCB in a restrictive set of arms
- ♠ Avoid the arm preferred/most played by agents raked higher
- ♠ Propose an epoch based method:
  - Agents estimate their rank first though collision
  - In each epoch, estimate the arms played/preferred by agents ranked higher through collisions
  - Remove them and play UCB on the restrictive set
- ♣ With this, regret of $\mathcal{O}(jK \log T)$ is obtained for j-th ranked agent

Explore-Then-Commit type algorithms

Explore-Then-Commit type algorithms

- ♣ Assume one sided learning
- ♠ ETGS (Explore-then-Gale-Shapley) by Kong et. al, SODA 23
- ♠ Idea: Explore and collect samples in a round robin fashion
- ♠ Construct the LCB and UCB indices

Explore-Then-Commit type algorithms

- Assume one sided learning
- ETGS (Explore-then-Gale-Shapley) by Kong et. al, SODA 23
- Idea: Explore and collect samples in a round robin fashion
- Construct the LCB and UCB indices
- Agent prefers arm ℓ_1 over ℓ_2 if LCB of arm $\ell_1 >$ UCB of arm ℓ_2

Explore-Then-Commit type algorithms

- Assume one sided learning
- ETGS (Explore-then-Gale-Shapley) by Kong et. al, SODA 23
- Idea: Explore and collect samples in a round robin fashion
- Construct the LCB and UCB indices
- Agent prefers arm ℓ_1 over ℓ_2 if LCB of arm $\ell_1 >$ UCB of arm ℓ_2
- Propose an epoch based method:
  - Agents estimate their preference though LCB and UCB indices
  - Commits to the most prefered arm
  - Exponentially growing epoch length so that after some finite epochs, gets the correct preference

Explore-Then-Commit type algorithms

- Assume one sided learning
- ETGS (Explore-then-Gale-Shapley) by Kong et. al, SODA 23
- Idea: Explore and collect samples in a round robin fashion
- Construct the LCB and UCB indices
- Agent prefers arm ℓ_1 over ℓ_2 if LCB of arm $\ell_1 >$ UCB of arm ℓ_2
- Propose an epoch based method:
  - Agents estimate their preference though LCB and UCB indices
  - Commits to the most prefered arm
  - Exponentially growing epoch length so that after some finite epochs, gets the correct preference
- With this, regret of $\mathcal{O}(K \log T)$ is obtained

Stationary vs. Dynamic Markets

Stationary vs. Dynamic Markets

- Previous works consider static market, and learn preferences only once

Stationary vs. Dynamic Markets

- Previous works consider static market, and learn preferences only once
- Markets are seldom stationary–preferences continuously evolving
  - Demand of sanitizers, masks increased during pandemic
  - Restaurants are busier over weekends

Stationary vs. Dynamic Markets

♣ Previous works consider static market, and learn preferences only once

♣ Markets are seldom stationary–preferences continuously evolving

- Demand of sanitizers, masks increased during pandemic
- Restaurants are busier over weekends

♣ Active area of research in Management sciences and Operations Research revolve around understanding the equilibrium properties in such evolving market (Lam et. al 05, Akbarpour et. al 20, Kurino 20, Johari et. al 21)

♣ In these works—The participants have exact knowledge over their preferences, and only need to optimize over other agents' competitive behavior with future changes

Stationary vs. Dynamic Markets

♣ Previous works consider static market, and learn preferences only once

♣ Markets are seldom stationary–preferences continuously evolving

- Demand of sanitizers, masks increased during pandemic
- Restaurants are busier over weekends

♣ Active area of research in Management sciences and Operations Research revolve around understanding the equilibrium properties in such evolving market (Lam et. al 05, Akbarpour et. al 20, Kurino 20, Johari et. al 21)

♣ In these works—The participants have exact knowledge over their preferences, and only need to optimize over other agents' competitive behavior with future changes

♣ We take a step towards bridging the two aforementioned lines of work

Non-Stationary Multi-Armed Bandit (MAB)

Non-Stationary Multi-Armed Bandit (MAB)

- Well studied in the MAB literature
- UCB does not work–Variations: discounted UCB, Window based UCB

Non-Stationary Multi-Armed Bandit (MAB)

- ♣ Well studied in the MAB literature
- ♣ UCB does not work–Variations: discounted UCB, Window based UCB Garivier 2008
- ♣ Environment vary slowly or abruptly (Wei et. al 2018)

Non-Stationary Multi-Armed Bandit (MAB)

- ♣ Well studied in the MAB literature
- ♣ UCB does not work–Variations: discounted UCB, Window based UCB Garivier 2008
- ♣ Environment vary slowly or abruptly (Wei et. al 2018)
- ♣ Total budget of change (a combination of slowly and abruptly) – Besbes et. al ,2014

Non-Stationary Multi-Armed Bandit (MAB)

♣ Well studied in the MAB literature

♣ UCB does not work–Variations: discounted UCB, Window based UCB Garivier 2008

♣ Environment vary slowly or abruptly (Wei et. al 2018)

♣ Total budget of change (a combination of slowly and abruptly) – Besbes et. al ,2014

♣ We use smooth varying framework of Wei and Srivastaba '18, Krishnamurthy and Gopalan '21)

- ♣ $|\mu_{.,.,t+1} - \mu_{.,.,t}| \leqslant \delta$ for all agent arm pairs

♣ Maximum drift is δ

♣ Useful in cloud computing, financial applications–decisions in seconds, distribution changes slowly

Smoothly Varying Markets – Regret

Smoothly Varying Markets – Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j

Smoothly Varying Markets – Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j

Smoothly Varying Markets – Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - At time t, Arm preferred by Agent 1:

$$\ell_*^{(1,t)} = \text{argmax}_{\ell \in [K]} \mu_{1,\ell,t}$$

Smoothly Varying Markets – Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - At time t, Arm preferred by Agent 1:

$$\ell_*^{(1,t)} = \mathrm{argmax}_{\ell \in [K]} \mu_{1,\ell,t}$$

  - Given collision structure—Arm preferred by Agent j is

$$\ell_*^{(j,t)} = \mathrm{argmax}_{\ell \in [K] \setminus \{\ell_*^{(1,t)},..,\ell_*^{(j-1,t)}\}} \mu_{j,\ell,t}$$

Smoothly Varying Markets – Regret

- Assume one sided learning and serial dictatorship
- Wlog, assume Agent j is ranked j
- Let's look at the unique stable match:
  - At time t, Arm preferred by Agent 1:

$$\ell_*^{(1,t)} = \mathrm{argmax}_{\ell \in [K]} \mu_{1,\ell,t}$$

  - Given collision structure—Arm preferred by Agent j is

$$\ell_*^{(j,t)} = \mathrm{argmax}_{\ell \in [K] \setminus \{\ell_*^{(1,t)}, .., \ell_*^{(j-1,t)}\}} \mu_{j,\ell,t}$$

- $(1, \ell_*^{(1,t)}), (2, \ell_*^{(2,t)}), \ldots, (N, \ell_*^{(N,t)})$ forms a stable match–unique
- Objective: Minimize regret with respect to this unique stable match

Algorithm: Non-stationary Competing Bandits (NSCB)

♣ We use the SNOOZE-IT algorithm of Krishnamurthy et.al'21

♣ Uses standard sliding window based successive Explore and Commit (similar to Slivkins '19)

Algorithm: Non-stationary Competing Bandits (NSCB)

♣ We use the SNOOZE-IT algorithm of Krishnamurthy et.al'21

♣ Uses standard sliding window based successive Explore and Commit (similar to Slivkins '19)

♣ Caveat: We also have competition accross multiple agents in the markets setup–seamless extension does not work

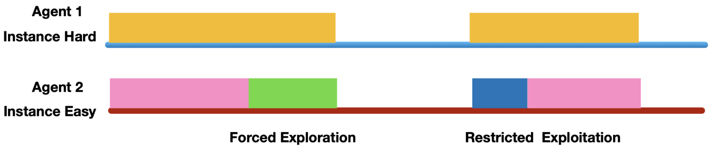
Algorithm: Non-stationary Competing Bandits (NSCB)

♣ We use the SNOOZE-IT algorithm of Krishnamurthy et.al'21

♣ Uses standard sliding window based successive Explore and Commit (similar to Slivkins '19)

♣ Caveat: We also have competition accross multiple agents in the markets setup–seamless extension does not work

♣ Forced Exploration and Restrictive Exploitation



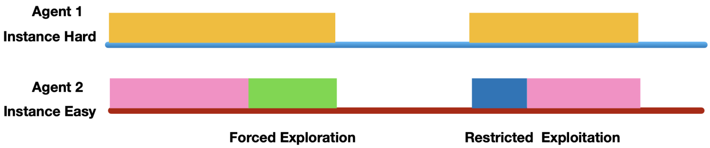
Algorithm: Non-stationary Competing Bandits (NSCB)

♣ We use the SNOOZE-IT algorithm of Krishnamurthy et.al'21

♣ Uses standard sliding window based successive Explore and Commit (similar to Slivkins '19)

♣ Caveat: We also have competition accross multiple agents in the markets setup–seamless extension does not work

♣ Forced Exploration and Restrictive Exploitation



◀ Agent 2 will face additional regret. We characterize this regret.

Non-stationary Competing Bandits (NSCB)

- Phase I: `Rank Estimation`
- Takes a total of $N-1$ time steps

Non-stationary Competing Bandits (NSCB)

- Phase I: `Rank Estimation`
- Takes a total of $N-1$ time steps
- At $t=1$, all agents pull arm 1

Non-stationary Competing Bandits (NSCB)

♣ Phase I: `Rank Estimation`

♣ Takes a total of $N - 1$ time steps

♣ At $t = 1$, all agents pull arm 1

♣ For $t \in [2, N - 1]$, agents who were matched, continues to play the matched arm

♣ Rest of the agents play arm index t

Non-stationary Competing Bandits (NSCB)

♣ Phase I: `Rank Estimation`

♣ Takes a total of $N - 1$ time steps

♣ At $t = 1$, all agents pull arm 1

♣ For $t \in [2, N - 1]$, agents who were matched, continues to play the matched arm

♣ Rest of the agents play arm index t Thanks to the collision structure,

at $N - 1$ instant, all agents know their own rank

Non-stationary Competing Bandits (NSCB)

♣ Phase I: `Rank Estimation`

♣ Takes a total of $N - 1$ time steps

♣ At $t = 1$, all agents pull arm 1

♣ For $t \in [2, N - 1]$, agents who were matched, continues to play the matched arm

♣ Rest of the agents play arm index t Thanks to the collision structure,

at $N - 1$ instant, all agents know their own rank

♣ Say $N = 2$, both agents pull arm 1–Agent 1 gets the reward, Agent 2 does not

♣ Based on this they know their rank

Non-stationary Competing Bandits (NSCB)

♣ Phase I: `Rank Estimation`

♣ Takes a total of $N-1$ time steps

♣ At $t=1$, all agents pull arm 1

♣ For $t \in [2, N-1]$, agents who were matched, continues to play the matched arm

♣ Rest of the agents play arm index t Thanks to the collision structure,

at $N-1$ instant, all agents know their own rank

♣ Say $N=2$, both agents pull arm 1–Agent 1 gets the reward, Agent 2 does not

♣ Based on this they know their rank

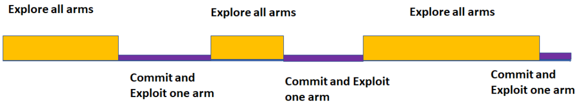
♠ Let us now look at the simplest non-trivial problem with $N=2$ agents

Non-stationary Competing Bandits (NSCB) for Agent 1

- ♣ Agent 1–highest ranked, faces no collision
- ♣ Uses standard sliding window based SNOOZE-IT algorithm (Krishnamurthy et.al '21)

Non-stationary Competing Bandits (NSCB) for Agent 1

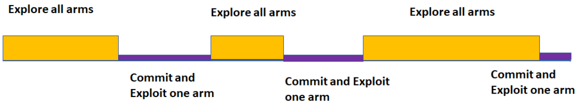
- Agent 1–highest ranked, faces no collision
- Uses standard sliding window based SNOOZE-IT algorithm (Krishnamurthy et.al '21)



Non-stationary Competing Bandits (NSCB) for Agent 1

♣ Agent 1–highest ranked, faces no collision

♣ Uses standard sliding window based SNOOZE-IT algorithm (Krishnamurthy et.al '21)



♣ Since arm means are changing, best arm is changing, so Agent 1 Explore and Commits successively over time.

◀ No market aspect, no competition

NSCB for Agent 1 contd.

- Epoch based–$s_1, s_2, ..$ denote starting of epoch
- In 1 epoch: Round robbin to find an *optimal* arm followed by exploiting that arm
- More formally Agent 1 performs the following test:

NSCB for Agent 1 contd.

♣ Epoch based–$s_1, s_2, ..$ denote starting of epoch

♣ In 1 epoch: Round robbin to find an *optimal* arm followed by exploiting that arm

♣ More formally Agent 1 performs the following test:

♣ Let $\hat{\mu}_{a,t}(\tilde{w})$ denote the empirical reward mean of arm a at time t, based on its last $\tilde{w}$ pulls.

NSCB for Agent 1 contd.

♣ Epoch based–$s_1, s_2, ..$ denote starting of epoch

♣ In 1 epoch: Round robbin to find an *optimal* arm followed by exploiting that arm

♣ More formally Agent 1 performs the following test:

♣ Let $\hat{\mu}_{a,t}(\tilde{w})$ denote the empirical reward mean of arm a at time t, based on its last $\tilde{w}$ pulls.

Definition

(Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$) At time t, an arm a is said to be Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$ with respect to set $\mathcal{A}$, if there exists $\tilde{\lambda} \in (0, 1)$ such that $\hat{\mu}_{a,t}(\tilde{w}) > \hat{\mu}_{b,t}(\tilde{w}) + 4r(\tilde{w}) - (k-1)\delta$, for all $b \in \mathcal{A} \setminus \{a\}$, where $\tilde{w} = \frac{c_1 \log T}{\tilde{\lambda}^2}$, and $r(\tilde{w}) = \sqrt{\frac{2 \log T}{\tilde{w}}}$.

NSCB for Agent 1 contd.

- Epoch based–$s_1, s_2, ..$ denote starting of epoch
- In 1 epoch: Round robbin to find an *optimal* arm followed by exploiting that arm
- More formally Agent 1 performs the following test:
- Let $\hat{\mu}_{a,t}(\tilde{w})$ denote the empirical reward mean of arm a at time t, based on its last $\tilde{w}$ pulls.

Definition

(Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$) At time t, an arm a is said to be Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$ with respect to set $\mathcal{A}$, if there exists $\tilde{\lambda} \in (0, 1)$ such that $\hat{\mu}_{a,t}(\tilde{w}) > \hat{\mu}_{b,t}(\tilde{w}) + 4r(\tilde{w}) - (k-1)\delta$, for all $b \in \mathcal{A} \setminus \{a\}$, where $\tilde{w} = \frac{c_1 \log T}{\tilde{\lambda}^2}$, and $r(\tilde{w}) = \sqrt{\frac{2 \log T}{\tilde{w}}}$.

- Once an arm a is found
  - Exploit arm a for $\frac{8}{\delta}\sqrt{\frac{k \log T}{t - s_i}} - 2(k-1)$ for the i-th epoch

NSCB for Agent 1 contd.

♣ Epoch based–$s_1, s_2, ..$ denote starting of epoch

♣ In 1 epoch: Round robbin to find an *optimal* arm followed by exploiting that arm

♣ More formally Agent 1 performs the following test:

♣ Let $\hat{\mu}_{a,t}(\tilde{w})$ denote the empirical reward mean of arm a at time t, based on its last $\tilde{w}$ pulls.

Definition

(Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$) At time t, an arm a is said to be Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$ with respect to set $\mathcal{A}$, if there exists $\tilde{\lambda} \in (0, 1)$ such that $\hat{\mu}_{a,t}(\tilde{w}) > \hat{\mu}_{b,t}(\tilde{w}) + 4r(\tilde{w}) - (k-1)\delta$, for all $b \in \mathcal{A} \setminus \{a\}$, where $\tilde{w} = \frac{c_1 \log T}{\tilde{\lambda}^2}$, and $r(\tilde{w}) = \sqrt{\frac{2 \log T}{\tilde{w}}}$.

♣ Once an arm a is found

- ♠ Exploit arm a for $\frac{8}{\delta}\sqrt{\frac{k \log T}{t - s_i}} - 2(k-1)$ for the i-th epoch

◀ Exploit period such that the arm remains optimal

Blackboard Communication Model

♣ We assume that agents can write *limited information* on a black-board, which other agents can see

Blackboard Communication Model

♣ We assume that agents can write *limited information* on a black-board, which other agents can see

♣ This is quite common is Distributed Optimization (Wainwright et. al)

♣ Mode of communication across various agents

Blackboard Communication Model

- ♣ We assume that agents can write *limited information* on a black-board, which other agents can see
- ♣ This is quite common is Distributed Optimization (Wainwright et. al)
- ♣ Mode of communication across various agents
- ♣ In Sankararaman et.al '21, this is achieved through structured collision

Blackboard Communication Model

♣ We assume that agents can write *limited information* on a black-board, which other agents can see

♣ This is quite common is Distributed Optimization (Wainwright et. al)

♣ Mode of communication across various agents

♣ In Sankararaman et.al '21, this is achieved through structured collision

♣ In Kong et. al '23, equivalent model is assumed through broadcasting

Blackboard Communication Model

- ♣ We assume that agents can write *limited information* on a black-board, which other agents can see
- ♣ This is quite common is Distributed Optimization (Wainwright et. al)
- ♣ Mode of communication across various agents
- ♣ In Sankararaman et.al '21, this is achieved through structured collision
- ♣ In Kong et. al '23, equivalent model is assumed through broadcasting
- ♣ Since we do not allow collision, we let the agents use the blackboard to communicate
- ♣ Note that this is still a decentralized system, at each time, we allow one agent to update

Blackboard Communication Model

- ♣ We assume that agents can write *limited information* on a black-board, which other agents can see
- ♣ This is quite common is Distributed Optimization (Wainwright et. al)
- ♣ Mode of communication across various agents
- ♣ In Sankararaman et.al '21, this is achieved through structured collision
- ♣ In Kong et. al '23, equivalent model is assumed through broadcasting
- ♣ Since we do not allow collision, we let the agents use the blackboard to communicate
- ♣ Note that this is still a decentralized system, at each time, we allow one agent to update
- ♣ After getting the optimal arm, Agent 1 updates the black-board
- ♣ Agent 1 writes the optimal arm index and the exploit time

Blackboard Communication Model

♣ We assume that agents can write *limited information* on a black-board, which other agents can see

♣ This is quite common is Distributed Optimization (Wainwright et. al)

♣ Mode of communication across various agents

♣ In Sankararaman et.al '21, this is achieved through structured collision

♣ In Kong et. al '23, equivalent model is assumed through broadcasting

♣ Since we do not allow collision, we let the agents use the blackboard to communicate

♣ Note that this is still a decentralized system, at each time, we allow one agent to update

♣ After getting the optimal arm, Agent 1 updates the black-board

♣ Agent 1 writes the optimal arm index and the exploit time

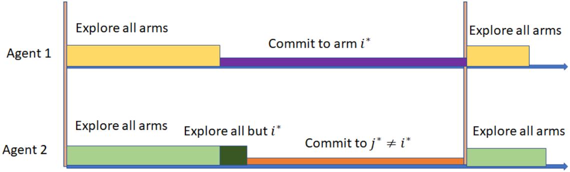
♠ We later remove the blackboard

NSCB for Agent 2

- Borns out the competitive nature of the market
- Agent 2–avoids collision; otherwise get 0 reward
- Let us look at different scenarios of Agent 2

NSCB for Agent 2

- Borns out the competitive nature of the market
- Agent 2–avoids collision; otherwise get 0 reward
- Let us look at different scenarios of Agent 2

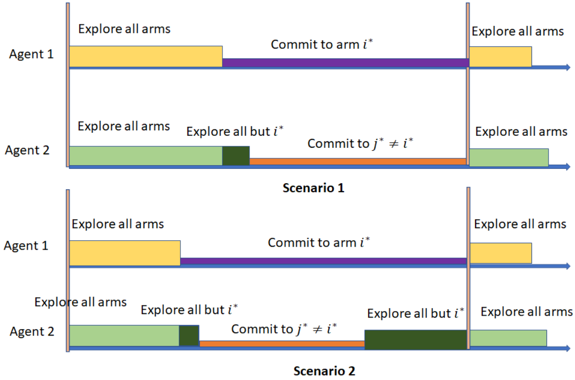


Scenario 1

Scenario 2

NSCB for Agent 2

- Borns out the competitive nature of the market
- Agent 2–avoids collision; otherwise get 0 reward
- Let us look at different scenarios of Agent 2



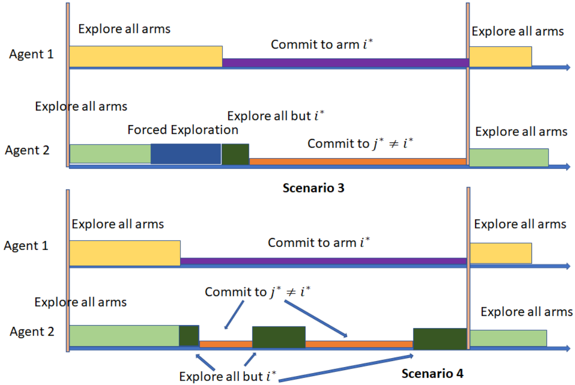
- (Domination 1) Restricted Exploration
- (Domination 2) Commitment to a dominated set of arms

NSCB for Agent 2 contd.

- Let us look at different scenarios of Agent 2

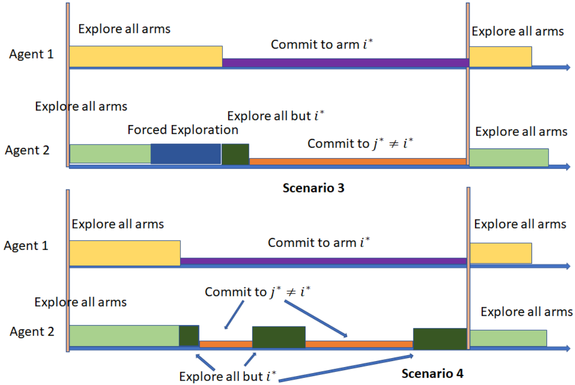
NSCB for Agent 2 contd.

- Let us look at different scenarios of Agent 2



NSCB for Agent 2 contd.

- Let us look at different scenarios of Agent 2



- (Domination 3) Forced Exploration
- (Domination 4) Restricted Exploitation

NSCB for Agent 2 contd.

- Let us summarize the actions of Agent 2

NSCB for Agent 2 contd.

- Let us summarize the actions of Agent 2

Algorithm 2 NSCB with $N = 2$; for Agent 2

Initialize set of tuples $S_2^{(j)} = \phi, \forall\, j \in [k]$, episode index $i_2 \leftarrow 1$
for $t = 1, 2, \ldots, T$ **do**
  Pull-Arm by Agent 2:
  if Agent 1 is not committed **then**
    Play round robbin on $[k]$ (pull arm $t + 1 \,\%\, k$), set $z_t(2) \leftarrow$Explore ALL
  else if Agent 1 is committed to arm j and $S_2^{(j)} = \phi$ **then**
    Play round robbin on $[k] \setminus \{j\}$ (i.e., pull arm index $t\%(k-1)$-th smallest arm id in $[k] \setminus \{j\}$)
    $z_t(2) \leftarrow$Explore-j
  else if Agent 1 is committed to arm j, and $\exists (x, s) \in S_2^{(j)}$ s.t. $s > t$ **then**
    Play arm x; set $z_t(2) \leftarrow$Exploit
  end if
  if $\{z_t(1) \neq z_{t-1}(1)\}$ OR $\{z_t(2) \neq z_{t-1}(2)\}$ **then**
    $i_2 \leftarrow i_2 + 1$, $t_{i_2} \leftarrow t$
  end if
  Test by Agent 2:
  for $j \in [k]$ s.t. $z_t(2) \in$ {Explore-j, Explore ALL} **do**
    if Arm a = Lambda-Opt $(\tilde{\lambda}, [k] \setminus \{j\})$ **then**
      $\tau_{i_2}^{(j)} \leftarrow t - t_{i_2}$, $\text{buffer}_2 = \max\left(\frac{8}{\delta}\sqrt{(k-1)\log T/\tau_{i_2}^{(j)}} - 2(k-2), 0\right)$
      if $\text{buffer}_2 > \tau_{i_2}^{(j)}$ **then**
        $S_2^{(j)} \leftarrow S_2^{(j)} \cup \{(a, \min\{t_{i_2} + \text{buffer}_2, s_{i_1+1})\}$
      end if
    end if
end for

Problem Complexity–Dynamic Gap

- Dynamic gap for agent r determines how complex the problem is
- Average gap between the pairwise arm-means over a window

Problem Complexity–Dynamic Gap

- Dynamic gap for agent r determines how complex the problem is
- Average gap between the pairwise arm-means over a window

Definition

For $\mathcal{C} \subseteq [k]$, the dynamic gap of Agent r on a dominated set $\mathcal{C}$ as,

$$\lambda_t^{\mathcal{C}}[r] = \max_{\lambda \in [0,1]} \left\{ \min_{\substack{a,b \in [k] \setminus \mathcal{C} \\ a \neq b}} \frac{1}{w(\lambda)} \left| \sum_{t'=s}^{t} \mu_{r,a,t'} - \mu_{r,b,t'} \right| \geqslant \lambda \right\},$$

and if such a λ does not exist, we set $\lambda_t^{\mathcal{C}}[r] = c_1 \sqrt{\frac{\log T}{t}}$. Here, $s = t - w(\lambda) + 1$, and $w(\lambda) = \frac{c_0 (k - |\mathcal{C}|) \log T}{\lambda^2}$.

Problem Complexity–Dynamic Gap

- Dynamic gap for agent r determines how complex the problem is
- Average gap between the pairwise arm-means over a window

Definition

For $\mathcal{C} \subseteq [k]$, the dynamic gap of Agent r on a dominated set $\mathcal{C}$ as,

$$\lambda_t^{\mathcal{C}}[r] = \max_{\lambda \in [0,1]} \left\{ \min_{\substack{a,b \in [k] \setminus \mathcal{C} \\ a \neq b}} \frac{1}{w(\lambda)} \left| \sum_{t'=s}^{t} \mu_{r,a,t'} - \mu_{r,b,t'} \right| \geqslant \lambda \right\},$$

and if such a λ does not exist, we set $\lambda_t^{\mathcal{C}}[r] = c_1 \sqrt{\frac{\log T}{t}}$. Here, $s = t - w(\lambda) + 1$, and $w(\lambda) = \frac{c_0(k - |\mathcal{C}|) \log T}{\lambda^2}$.

- No superscript when $\mathcal{C} = \phi$
- When $\delta = 0$, reduced to the usual gap in MAB
- Generalization of the standard gap

Regret Guarantee

- For Agent 1, regret of successive ETC
- For Agent 2, account for forced exploration and restrictive exploitation

Regret Guarantee

- For Agent 1, regret of successive ETC
- For Agent 2, account for forced exploration and restrictive exploitation

Theorem: Suppose we run `NSCB` with 2 Agents upto horizon T. Then the expected regret for Agent 1 is

$$R_1 \lesssim \sum_{l=1}^{m} \frac{1}{\lambda_{\min,l}[1]} k \log T$$

and for Agent 2 is

$$R_2 \lesssim \sum_{l=1}^{m} \left[\left(\frac{1}{\lambda_{\min,l}[2]} + \frac{1}{(\lambda_{\min,l}[1])^2} + \frac{1}{\min_a \lambda_{\min,l}^{\{a\}}[2]} \right) k \log T \right]$$

Regret Guarantee

- For Agent 1, regret of successive ETC
- For Agent 2, account for forced exploration and restrictive exploitation

Theorem: Suppose we run NSCB with 2 Agents upto horizon T. Then the expected regret for Agent 1 is

$$R_1 \lesssim \sum_{l=1}^{m} \frac{1}{\lambda_{\min,l}[1]} k \log T$$

and for Agent 2 is

$$R_2 \lesssim \sum_{l=1}^{m} \left[\left(\frac{1}{\lambda_{\min,l}[2]} + \frac{1}{(\lambda_{\min,l}[1])^2} + \frac{1}{\min_a \lambda^{\{a\}}_{\min,l}[2]} \right) k \log T \right]$$

- T is divided into m blocks with length $\delta^{-2/3} k^{1/3} \log^{1/3} T$
- $\lambda_{\min,l}[r] = \min_{t \in l\text{-th block}} \lambda_t[r]$, $\lambda^{\{a\}}_{\min,l}[r] = \min_{t \in l\text{-th block}} \lambda^{\{a\}}_t[r]$

Discussion

- For Agent 1, regret matches `SNOOZE-IT`
- Depends on the dynamic gap

Discussion

- For Agent 1, regret matches `SNOOZE-IT`
- Depends on the dynamic gap
- For Agent 2, we have 3 terms:

Discussion

- ♣ For Agent 1, regret matches `SNOOZE-IT`
- ♣ Depends on the dynamic gap
- ♣ For Agent 2, we have 3 terms:
  - First term: Comes from Exploration-ALL phase

Discussion

- For Agent 1, regret matches `SNOOZE-IT`
- Depends on the dynamic gap
- For Agent 2, we have 3 terms:
  - First term: Comes from Exploration-ALL phase
  - Second Term: Forced Exploration and Restrictive Exploitation–depends on the dynamic gap of Agent 1

Discussion

- For Agent 1, regret matches `SNOOZE-IT`
- Depends on the dynamic gap
- For Agent 2, we have 3 terms:
  - First term: Comes from Exploration-ALL phase
  - Second Term: Forced Exploration and Restrictive Exploitation–depends on the dynamic gap of Agent 1
  - Third Term: Comes from Restrictive exploration (Explore-j) phase, where Agent 1 is committed to arm j

Discussion

- For Agent 1, regret matches `SNOOZE-IT`
- Depends on the dynamic gap
- For Agent 2, we have 3 terms:
  - First term: Comes from Exploration-ALL phase
  - Second Term: Forced Exploration and Restrictive Exploitation–depends on the dynamic gap of Agent 1
  - Third Term: Comes from Restrictive exploration (Explore-j) phase, where Agent 1 is committed to arm j
- Regret matches that of UCB-D3 in stationary setup by putting $\delta = 0$. We get a regret of $\mathcal{O}(\frac{1}{\mathsf{Gap}^2} k \log T)$

NSCB for N Agents

NSCB for N Agents

♣ At time t, Agent r looks at the blackboard and constructs dominated set $\mathcal{C}_t(r)$: the set of committed arms by agents ranked $1, 2, \ldots, r-1$

NSCB for N Agents

♣ At time t, Agent r looks at the blackboard and constructs dominated set $\mathcal{C}_t(r)$: the set of committed arms by agents ranked $1, 2, \ldots, r-1$

♣ Based on $\mathcal{C}_t(r)$, Agent will either Explore-$\mathcal{C}_t(r)$ or Exploit

NSCB for N Agents

♣ At time t, Agent r looks at the blackboard and constructs dominated set $\mathcal{C}_t(r)$: the set of committed arms by agents ranked $1, 2, \dots, r-1$

♣ Based on $\mathcal{C}_t(r)$, Agent will either Explore-$\mathcal{C}_t(r)$ or Exploit

♣ Agent r gets to commit on an arm $[k]$ $\mathcal{C}_t(r)$ only if all the higher ranked agents have committed, i.e., $|\mathcal{C}_t(r)| = r - 1$

NSCB for N Agents

♣ At time t, Agent r looks at the blackboard and constructs dominated set $\mathcal{C}_t(r)$: the set of committed arms by agents ranked $1, 2, \ldots, r-1$

♣ Based on $\mathcal{C}_t(r)$, Agent will either Explore-$\mathcal{C}_t(r)$ or Exploit

♣ Agent r gets to commit on an arm $[k]\ \mathcal{C}_t(r)$ only if all the higher ranked agents have committed, i.e., $|\mathcal{C}_t(r)| = r - 1$

♣ Also, Agent r ends faces restricted exploitation if anyone of agents $\{1, \ldots, r-1\}$ ends their exploitation.

NSCB for N Agents

♣ At time t, Agent r looks at the blackboard and constructs dominated set $\mathcal{C}_t(r)$: the set of committed arms by agents ranked $1, 2, \ldots, r-1$

♣ Based on $\mathcal{C}_t(r)$, Agent will either Explore-$\mathcal{C}_t(r)$ or Exploit

♣ Agent r gets to commit on an arm $[k]$ $\mathcal{C}_t(r)$ only if all the higher ranked agents have committed, i.e., $|\mathcal{C}_t(r)| = r - 1$

♣ Also, Agent r ends faces restricted exploitation if anyone of agents $\{1, \ldots, r-1\}$ ends their exploitation.

♣ When Agent r commits to an arm, it updates the blackboard by the arm index and the committed period

NSCB for N Agents

- ♣ Let us summarize the actions of Agent r

NSCB for N Agents

♣ Let us summarize the actions of Agent r

Algorithm 3 NSCB for r-th Agent

Input: Horizon T, drift limit δ

Initialize $S_r^{(\Omega)} = \phi$ for all $\Omega \subseteq [k]$, and $|\Omega| \leq r-1$, Initialize $i_r \leftarrow 1$, $\mathcal{C}_1(r) = \phi$

RANK ESTIMATION()

for $t = 1, 2, \ldots, T$ **do**

Update State $z_t(r)$**:**

if $|\mathcal{C}_t(r)| < r-1$ **then**

$z_t(r) \leftarrow \texttt{Explore} - \mathcal{C}_t(r)$

else if $\exists (x, s) \in S_r^{(\mathcal{C}_t(r))}$ s.t. $s > t$, **then**

$z_t(r) \leftarrow \texttt{Exploit}(x)$

else

$z_t(r) \leftarrow \texttt{Explore} - \mathcal{C}_t(r)$

end if

if $z_t(r) \neq z_{t-1}(r)$ **then**

$i_r \leftarrow i_r + 1, \;\; t_{i_r} \leftarrow t$

end if

Pull-Arm by Agent r:

if $z_t = \texttt{Explore} - \mathcal{C}_t(r)$ **then**

Play round robbin with $[k] \setminus \mathcal{C}_t(r)$ (i.e., pull $t + (r - |\mathcal{C}_t(r)| - 1)\%(k - |\mathcal{C}_t(r)|)$ smallest arm in $[k] \setminus \mathcal{C}_t(r)$)

else if $z_t = \texttt{Exploit}(x)$ **then**

Play arm x

end if

Test by Agent r**:**

for $\Omega \subseteq [k]$ s.t. $|\Omega| = r-1$ and $z_t(r) = \texttt{Explore-}\mathcal{C}_t(r)$ **do**

if Arm a = Lambda-Opt $(\tilde{\lambda}, [k] \setminus \Omega)$ **then**

$\tau_{i_r}^{(\Omega)} \leftarrow t - t_{i_r}$, $\text{buffer}_r = \max\left(\frac{8}{\delta}\sqrt{(k - |\mathcal{C}_t(r)|)\frac{\log T}{\tau_{i_r}^{(\Omega)}}} - 2(k-r), 0\right)$,

define $\bar{t}_{i_r} = \min\{t_{i_r} + \text{buffer}_{i_r}, t_{i_{r-1}+1}\}$

if $\text{buffer}_{i_r} > \tau_{i_r}$, **then** $S_r^{\Omega} \leftarrow S_r^{\Omega} \cup \{(a, \bar{t}_{i_r})\}$,

else $i_{i_r} \leftarrow i_r + 1, t_{i_r} \leftarrow t$

end if

end if

end for

Updates Black Board:

if $\exists (x, s)$ s.t. $s \geq t+1$, **then** write $(x, \bar{t}_{i_r}, r)$ on the black board **end if**

Updates Dominated set $\mathcal{C}_{t+1}(r)$**:**

Updates $\mathcal{C}_{t+1}(r) = \{x \in [k] : \exists s > t+1$, and $\exists j \leq r-1$ *s.t.* (x, s, j) exists on board$\}$

end if

end for

NSCB for N Agents

- ♣ For Agent r–look at the behavior of $r-1$-th Agent
- ♣ Uses inductive structure induced by the serial dictatorship framework

NSCB for N Agents

- For Agent r–look at the behavior of $r-1$-th Agent
- Uses inductive structure induced by the serial dictatorship framework

Theorem: Suppose we run NSCB with N Agents upto horizon T. Then the expected regret for Agent r is

$$R_r \lesssim \sum_{\text{phases}} \left[\left(\frac{1}{\mathtt{Gap}[r]} + \frac{1}{\mathtt{Gap}^2[r-1]} \right) k \log T \right]$$

NSCB for N Agents

♣ For Agent r–look at the behavior of $r-1$-th Agent

♣ Uses inductive structure induced by the serial dictatorship framework

■ Theorem: Suppose we run NSCB with N Agents upto horizon T. Then the expected regret for Agent r is

$$R_r \lesssim \sum_{\text{phases}} \left[\left(\frac{1}{\mathtt{Gap}[r]} + \frac{1}{\mathtt{Gap}^2[r-1]} \right) k \log T \right]$$

♣ Agent r is dominated by Agents $1, \ldots, r-1$–captured in gap def.

♣ Regret of Agent r depends on all the agents $1, \ldots, r-1$

♣ Similar to the 2 Agent case, the first term comes from Exploring all arms, and the second term comes from forced exploration and restricted exploitation.

Learning without Blackboard; $N = 2$

♣ With blackboard, Agent 2 knows whether Agent 1 is exploring or committed to a particular arm

♣ Agent 2 maintains a latent variable $Q_t \in \{\text{Explore}, \text{Exploit}\}$: indicates action of Agent 1

Learning without Blackboard; $N = 2$

♣ With blackboard, Agent 2 knows whether Agent 1 is exploring or committed to a particular arm

♣ Agent 2 maintains a latent variable $Q_t \in \{\text{Explore}, \text{Exploit}\}$: indicates action of Agent 1

♣ If Agent 2 faces collision on arm j then either

Learning without Blackboard; $N = 2$

♣ With blackboard, Agent 2 knows whether Agent 1 is exploring or committed to a particular arm

♣ Agent 2 maintains a latent variable $Q_t \in \{\text{Explore}, \text{Exploit}\}$: indicates action of Agent 1

♣ If Agent 2 faces collision on arm j then either

- (a) Agent 1 has ended exploration and committed on arm j
- (b) Agent 1 has ended exploitation and is exploring

Learning without Blackboard; $N = 2$

♣ With blackboard, Agent 2 knows whether Agent 1 is exploring or committed to a particular arm

♣ Agent 2 maintains a latent variable $Q_t \in \{\text{Explore}, \text{Exploit}\}$: indicates action of Agent 1

♣ If Agent 2 faces collision on arm j then either

- (a) Agent 1 has ended exploration and committed on arm j
- (b) Agent 1 has ended exploitation and is exploring

♣ After a collision Agent 2 looks at Q_{t-1}

- If $Q_{t-1} =$ Explore, (a) has occurred
- If $Q_{t-1} =$ Exploit, (b) has occurred

Learning without Blackboard; $N = 2$

♣ With blackboard, Agent 2 knows whether Agent 1 is exploring or committed to a particular arm

♣ Agent 2 maintains a latent variable $Q_t \in \{\text{Explore}, \text{Exploit}\}$: indicates action of Agent 1

♣ If Agent 2 faces collision on arm j then either

- (a) Agent 1 has ended exploration and committed on arm j
- (b) Agent 1 has ended exploitation and is exploring

♣ After a collision Agent 2 looks at Q_{t-1}

- If $Q_{t-1} =$ Explore, (a) has occurred
- If $Q_{t-1} =$ Exploit, (b) has occurred

♣ Toggling Q_t is sufficient to get the information broadcasted by the blackboard

Learning without Blackboard; $N = 2$

♣ With blackboard, Agent 2 knows whether Agent 1 is exploring or committed to a particular arm

♣ Agent 2 maintains a latent variable $Q_t \in \{\text{Explore}, \text{Exploit}\}$: indicates action of Agent 1

♣ If Agent 2 faces collision on arm j then either

- (a) Agent 1 has ended exploration and committed on arm j
- (b) Agent 1 has ended exploitation and is exploring

♣ After a collision Agent 2 looks at Q_{t-1}

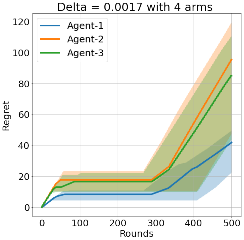
- If $Q_{t-1} =$ Explore, (a) has occurred
- If $Q_{t-1} =$ Exploit, (b) has occurred

♣ Toggling Q_t is sufficient to get the information broadcasted by the blackboard

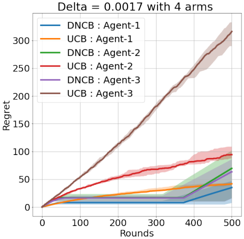
♣ Same idea can be extended to the N Agent setup

Simulations

♣ Consider 3 Agents $N = 3$, 4 Arms $K = 4$



♣ Compare with Dominated UCB of Sankararaman et. al'21



♣ Dominated UCB is for static market–hence large regret

Open Problems

- Markets and Bandits framework has several open problems
- We only consider smooth changes with known δ, what about abrupt change or total budgeted change?
- Beyond Serial Dictatorship?

Open Problems

♣ Markets and Bandits framework has several open problems

♣ We only consider smooth changes with known δ, what about abrupt change or total budgeted change?

♣ Beyond Serial Dictatorship?

♣ All works in this framework consider one sided learning to resolve conflict—the two sided learning problem in decentralized framework is still open

♣ Recent Work: Two-Sided Bandit Learning in Fully-Decentralized Matching Markets; Tejas Pagare, Avishek Ghosh, 2023 (short version ICML 2023 Workshop on Preference Learning, Full version under review)

Open Problems

♣ Markets and Bandits framework has several open problems

♣ We only consider smooth changes with known δ, what about abrupt change or total budgeted change?

♣ Beyond Serial Dictatorship?

♣ All works in this framework consider one sided learning to resolve conflict—the two sided learning problem in decentralized framework is still open

♣ Recent Work: Two-Sided Bandit Learning in Fully-Decentralized Matching Markets; Tejas Pagare, Avishek Ghosh, 2023 (short version ICML 2023 Workshop on Preference Learning, Full version under review)

♣ Spurious Markets: Handling adversaries amidst competition?

♣ Structured Markets: Linear/Contextual Bandits?

Reference

♣ Two-Sided Bandit Learning in Fully-Decentralized Matching Markets; Tejas Pagare and Avishek Ghosh – International Conference on Machine Learning (ICML) Workshop on Many Facets of Preference-Based Learning, Hawaii, 2023

♣ Decentralized Competing Bandits In Non-Stationary Matching Markets – Avishek Ghosh, Abishek Sankararaman, Kannan Ramchandran, Tara Javidi and Arya Mazumdar– IEEE Transactions on Information Theory, 2023

Reference contd.

♣ Two-Sided Bandit Learning in Fully-Decentralized Matching Markets; Tejas Pagare and Avishek Ghosh – available at https://tejassp2002.github.io/assets/pdf/MFPL_Workshop_ICML_Tejas_Avishek.pdf

♣ Decentralized Competing Bandits In Non-Stationary Matching Markets – Avishek Ghosh, Abishek Sankararaman, Kannan Ramchandran, Tara Javidi and Arya Mazumdar– available at https://arxiv.org/pdf/2206.00120.pdf

Thank You