Online Learning for Combinatorial Network Optimization with Restless Markovian Rewards

Yi Gai*, Bhaskar Krishnamachari* and Mingyan Liu‡

* Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089

‡ Department of Electrical Engineering and Computer Science
University of Michigan, MI 48109

{*ygai, bkrishna*}*@usc.edu*; mingyan@eecs.umich.edu

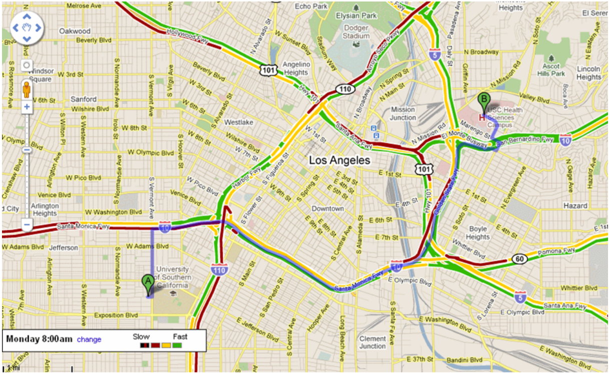
June 19, 2012

Outline

1. Introduction
   - Motivating Examples
   - General Formulation: MAB with Linear Rewards
   - Preliminaries
   - Problem Formulation
   - Applications
   - Challenges
2. Combinatorial Learning with Restless Markov Rewards (CLRMR)
   - Contribution
   - Proposed Algorithms
   - Analysis of Regret
   - An Extension
   - Simulations
3. Conclusion

Motivating Example 1

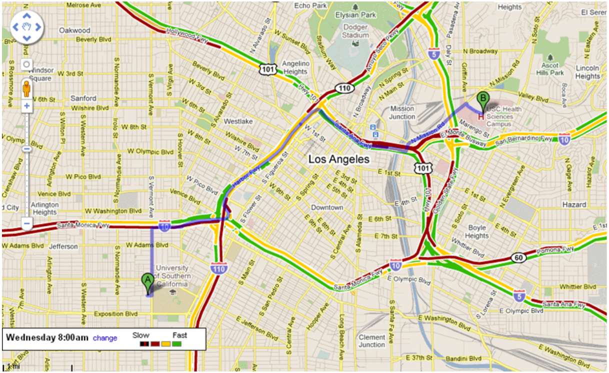
- Finding the lowest expected delay path through traffic using prior observations.



A sample path from Google Maps.

Motivating Example 1

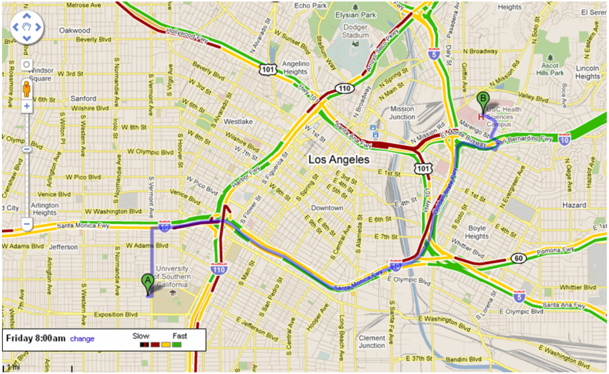
- Finding the lowest expected delay path through traffic using prior observations.



A sample path from Google Maps.

Motivating Example 1

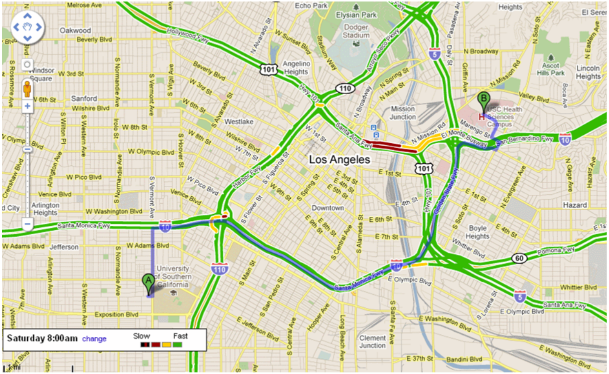
- Finding the lowest expected delay path through traffic using prior observations.



A sample path from Google Maps.

Motivating Example 1

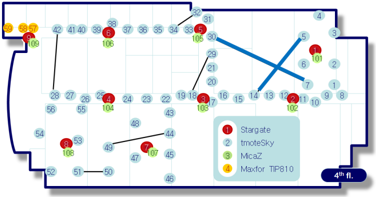
- Finding the lowest expected delay path through traffic using prior observations.



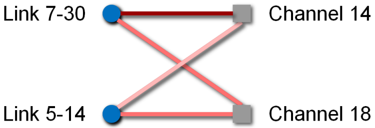
A sample path from Google Maps.

Motivating Example 2

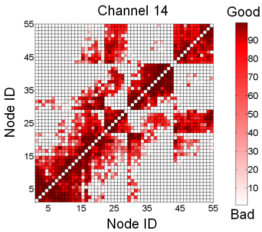
- Channel allocation for wireless links.



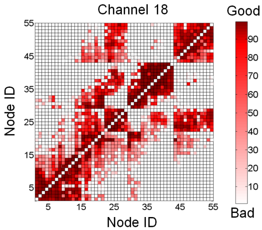
The TutorNet testbed at USC.



Bipartite link channel allocation graph.



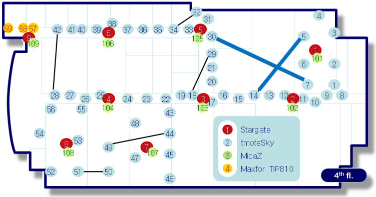
Link qualities on channel 14.



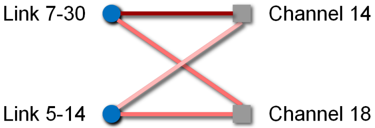
Link qualities on channel 18.

Motivating Example 2

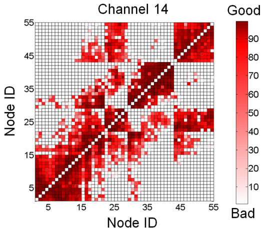
- Channel allocation for wireless links.



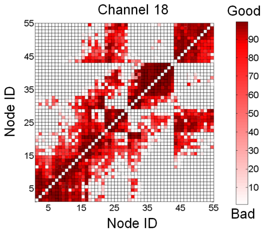
The TutorNet testbed at USC.



Bipartite link channel allocation graph.



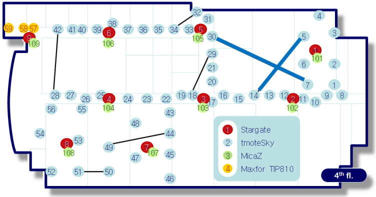
Link qualities on channel 14.



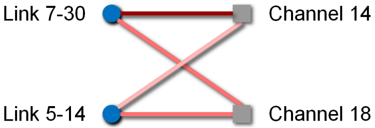
Link qualities on channel 18.

Motivating Example 2

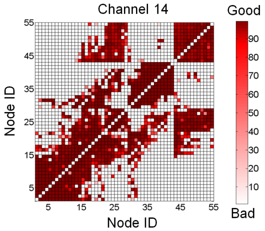
- Channel allocation for wireless links.



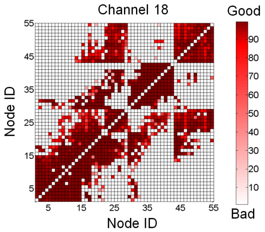
The TutorNet testbed at USC.



Bipartite link channel allocation graph.



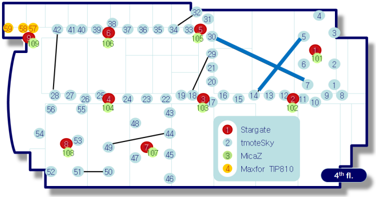
Link qualities on channel 14.



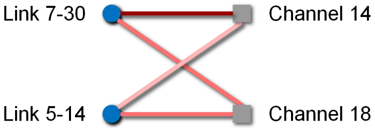
Link qualities on channel 18.

Motivating Example 2

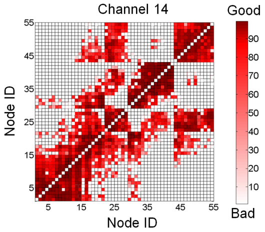
- Channel allocation for wireless links.



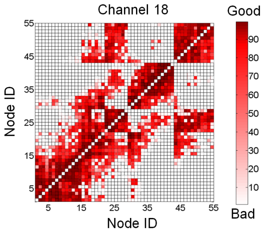
The TutorNet testbed at USC.



Bipartite link channel allocation graph.



Link qualities on channel 14.



Link qualities on channel 18.

Online Learning for Stochastic Network Optimization

- Common theme: find an optimal network structure (best path / matching), assuming the underlying edge weights are unknown random variables.

Online Learning for Stochastic Network Optimization

- Common theme: find an optimal network structure (best path / matching), assuming the underlying edge weights are unknown random variables.
- Problem formulation:

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned} \tag{1}$$

Online Learning for Stochastic Network Optimization

- Common theme: find an optimal network structure (best path / matching), assuming the underlying edge weights are unknown random variables.
- Problem formulation:

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau) X_i(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned} \tag{1}$$

where $X_i(\tau)$ are unknown random variables;

Online Learning for Stochastic Network Optimization

- Common theme: find an optimal network structure (best path / matching), assuming the underlying edge weights are unknown random variables.
- Problem formulation:

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned} \tag{1}$$

where $X_i(\tau)$ are unknown random variables; $\mathbf{a}(\tau)$ is action at time τ;

Online Learning for Stochastic Network Optimization

- Common theme: find an optimal network structure (best path / matching), assuming the underlying edge weights are unknown random variables.
- Problem formulation:

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned} \tag{1}$$

where $X_i(\tau)$ are unknown random variables; $\mathbf{a}(\tau)$ is action at time τ; $\mathcal{F}$ is a finite set.

Online Learning for Stochastic Network Optimization

- Common theme: find an optimal network structure (best path / matching), assuming the underlying edge weights are unknown random variables.
- Problem formulation:

$$\begin{aligned} \max \quad & \mathbb{E}\Big[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau) X_i(\tau)\Big] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned} \tag{1}$$

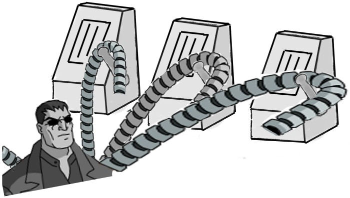
where $X_i(\tau)$ are unknown random variables; $\mathbf{a}(\tau)$ is action at time τ; $\mathcal{F}$ is a finite set.

General goal

- Develop online learning algorithms for combinatorial network optimization with restless Markovian rewards.

Multi-Armed Bandits (MAB)

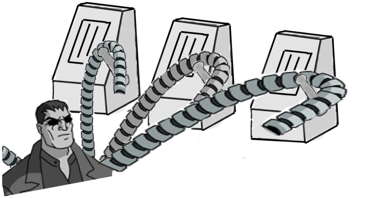
- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



Multi-Armed Bandit Problem

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.

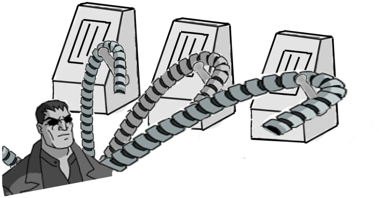


Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



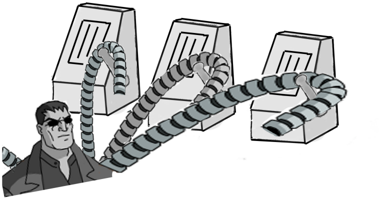
Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

- K slot machines (arms).

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



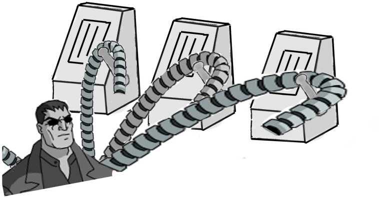
Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

- K slot machines (arms).
- Each arm generates random reward by i.i.d. sampling from an unknown distribution.

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



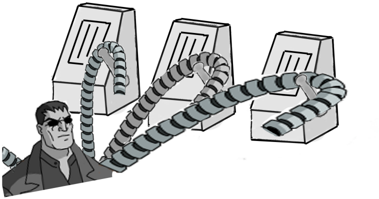
Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

- K slot machines (arms).
- Each arm generates random reward by i.i.d. sampling from an unknown distribution.
- The rewards are independent across arms.

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



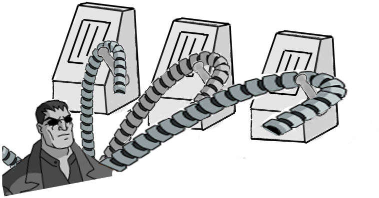
Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

- K slot machines (arms).
- Each arm generates random reward by i.i.d. sampling from an unknown distribution.
- The rewards are independent across arms.
- Pull one arm at each time slot.

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



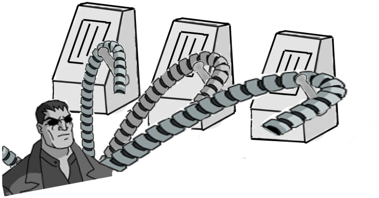
Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

- K slot machines (arms).
- Each arm generates random reward by i.i.d. sampling from an unknown distribution.
- The rewards are independent across arms.
- Pull one arm at each time slot.
- Objective: maximize long-term total reward.

Multi-Armed Bandits (MAB)

- Multi-armed bandit (MAB) problems provide a fundamental approach to learning under stochastic rewards.



Multi-Armed Bandit Problem

Classic Multi-Armed Bandit Problem

- K slot machines (arms).
- Each arm generates random reward by i.i.d. sampling from an unknown distribution.
- The rewards are independent across arms.
- Pull one arm at each time slot.
- Objective: maximize long-term total reward.

Trade-off

- Exploration vs Exploitation

Evaluation: Regret

Evaluation of learning algorithm performance:

Regret

Definition: the difference between the total expected reward, summed over times 1 to t, that could be obtained by a genie that can pick an optimal arm at each time, and that obtained by the given algorithm.

Evaluation: Regret

Evaluation of learning algorithm performance:

Regret

Definition: the difference between the total expected reward, summed over times 1 to t, that could be obtained by a genie that can pick an optimal arm at each time, and that obtained by the given algorithm.

Two varieties of upper bounds on regret:

- asymptotic: only achieved when $t \to \infty$
- uniform: achieved for every t

Problem Formulation

Problem Formulation: MAB with Markovian rewards

Problem Formulation

Problem Formulation: MAB with Markovian rewards

- **Markovian rewards**: the rewards are associated with finite-state Markov chains, with unknown transition matrices.

Problem Formulation

Problem Formulation: MAB with Markovian rewards

- **Markovian rewards**: the rewards are associated with finite-state Markov chains, with unknown transition matrices.
- **Restless Markovian rewards**: MCs evolve every time slot.

Problem Formulation

Problem Formulation: MAB with Markovian rewards

- **Markovian rewards**: the rewards are associated with finite-state Markov chains, with unknown transition matrices.
- **Restless Markovian rewards**: MCs evolve every time slot.
- **weak regret**: optimal policy plays a static arm
  (the problem is difficult and proved to be PSPACE-hard even when the transition matrices are known)

Problem Formulation

Problem Formulation: MAB with Markovian rewards

- **Markovian rewards**: the rewards are associated with finite-state Markov chains, with unknown transition matrices.
- **Restless Markovian rewards**: MCs evolve every time slot.
- **weak regret**: optimal policy plays a static arm
  (the problem is difficult and proved to be PSPACE-hard even when the transition matrices are known)
- Time is slotted, indexed by t.

Problem Formulation

Problem Formulation: MAB with Markovian rewards

- **Markovian rewards**: the rewards are associated with finite-state Markov chains, with unknown transition matrices.
- **Restless Markovian rewards**: MCs evolve every time slot.
- **weak regret**: optimal policy plays a static arm
(the problem is difficult and proved to be PSPACE-hard even when the transition matrices are known)
- Time is slotted, indexed by t.
- N edges.

Problem Formulation

Problem Formulation: MAB with Markovian rewards

- **Markovian rewards**: the rewards are associated with finite-state Markov chains, with unknown transition matrices.
- **Restless Markovian rewards**: MCs evolve every time slot.
- **weak regret**: optimal policy plays a static arm
  (the problem is difficult and proved to be PSPACE-hard even when the transition matrices are known)
- Time is slotted, indexed by t.
- N edges.

Other notations:

- i: index of edges (MCs)
- $\mathbf{a}$: index of an arm, an N-dimensional action vector

Prior Work

- Thompson '33 (first work on MAB).

Prior Work

- Thompson '33 (first work on MAB).
- i.i.d. rewards:
    - Lai & Robbins'85:
        - lower bound of regret: $K \ln t$
        - proposed a policy that achieves an asymptotical upper bound on regret $O(K \ln t)$
    - Anantharam *et al.*'87: extension from single play to multiple plays.
    - Auer *et al.*'02 (UCB1 algorithm): an optimal logarithmic regret is achievable uniformly over time

$$\text{regret}(t) \leq C_1 K \ln t + C_2 \qquad \text{for all } t$$

Prior Work

- Thompson '33 (first work on MAB).
- i.i.d. rewards:
    - Lai & Robbins'85:
        - lower bound of regret: $K \ln t$
        - proposed a policy that achieves an asymptotical upper bound on regret $O(K \ln t)$
    - Anantharam *et al.*'87: extension from single play to multiple plays.
    - Auer *et al.*'02 (UCB1 algorithm): an optimal logarithmic regret is achievable uniformly over time

$$\text{regret}(t) \leq C_1 K \ln t + C_2 \qquad \text{for all } t$$

- Restless Markovian rewards:
    - Tekin and Liu'10: RCA algorithm with logarithmic weak regret
    - Liu *et al.*'10: RUCB algorithm with logarithmic weak regret
    - Dai *et al.*'11: SPUDC algorithm for MCs with identical transition matrices with near-logarithmic regret

Prior Work

- Thompson '33 (first work on MAB).
- i.i.d. rewards:
  - Lai & Robbins'85:
    - lower bound of regret: $K \ln t$
    - proposed a policy that achieves an asymptotical upper bound on regret $O(K \ln t)$
  - Anantharam *et al.*'87: extension from single play to multiple plays.
  - Auer *et al.*'02 (UCB1 algorithm): an optimal logarithmic regret is achievable uniformly over time

$$\text{regret}(t) \leq C_1 K \ln t + C_2 \qquad \text{for all } t$$

- Restless Markovian rewards:
  - Tekin and Liu'10: RCA algorithm with logarithmic weak regret
  - Liu *et al.*'10: RUCB algorithm with logarithmic weak regret
  - Dai *et al.*'11: SPUDC algorithm for MCs with identical transition matrices with near-logarithmic regret
- These prior works do not consider dependencies across arms.

Prior Work

- Thompson '33 (first work on MAB).
- i.i.d. rewards:
  - Lai & Robbins'85:
    - lower bound of regret: $K \ln t$
    - proposed a policy that achieves an asymptotical upper bound on regret $O(K \ln t)$
  - Anantharam *et al.*'87: extension from single play to multiple plays.
  - Auer *et al.*'02 (UCB1 algorithm): an optimal logarithmic regret is achievable uniformly over time

$$\text{regret}(t) \leq C_1 K \ln t + C_2 \qquad \text{for all } t$$

- Restless Markovian rewards:
  - Tekin and Liu'10: RCA algorithm with logarithmic weak regret
  - Liu *et al.*'10: RUCB algorithm with logarithmic weak regret
  - Dai *et al.*'11: SPUDC algorithm for MCs with identical transition matrices with near-logarithmic regret
- These prior works do not consider dependencies across arms.
- **MAB with Linear rewards: dependencies!**

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t} \sum_{i=1}^{N} a_i(\tau) X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

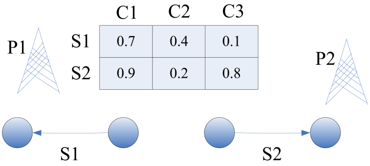
Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



	C1	C2	C3
S1	0.7	0.4	0.1
S2	0.9	0.2	0.8

How to allocate channels to secondary users?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

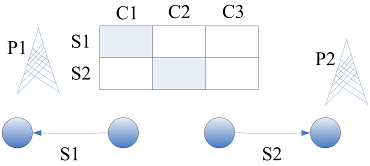
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



How to allocate channels to secondary users? arm 1?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

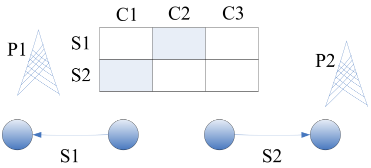
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



How to allocate channels to secondary users? arm 2?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

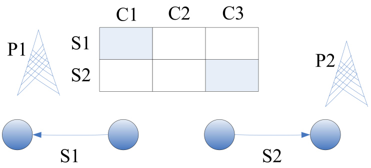
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



How to allocate channels to secondary users? arm 3?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

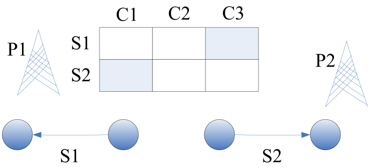
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



How to allocate channels to secondary users? arm 4?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

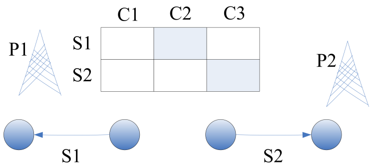
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



How to allocate channels to secondary users? arm 5?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

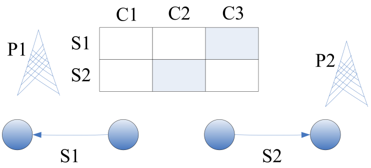
Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



How to allocate channels to secondary users? arm 6?

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

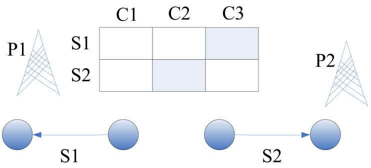
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



Q channels, M coordinated secondary users.

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

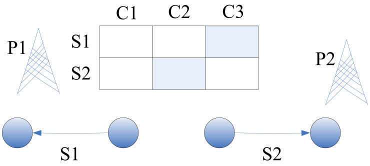
$\longrightarrow$

Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



Q channels, M coordinated secondary users.$\rightarrow$ **only $Q \times M$ unknown variables!**

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

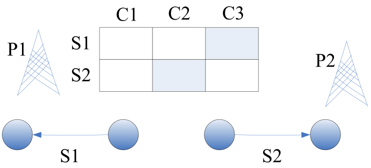
Stochastic Maximum Weighted Matching (MWM)

$$\max \quad R_{\mathbf{a}}^{MWM}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{|E|} a_i(\tau)W_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \text{ is a matching}$$

where $\{W_i(\tau)\}$ are unknown edge weights.

Application: learning multiuser channel allocations in cognitive radio networks.



Q channels, M coordinated secondary users.$\rightarrow$ **only $Q \times M$ unknown variables!**$\rightarrow$ **$P(Q, M)$ matchings (arms)! (e.g. $9 \times 5 = 45$, however $P(9,5) = 15120$)**

Application Examples

General Formulation

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned}$$

where $\{X_i(\tau)\}$ are unknown random variables.

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

Stochastic Shortest Path (SP) Routing

$$\min \quad C_{\mathbf{a}}^{SP}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i\in E} a_i(\tau)D_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is an s-t path}$$

where $\{D_i(\tau)\}$ are unknown link costs.

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t} \sum_{i=1}^{N} a_i(\tau) X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

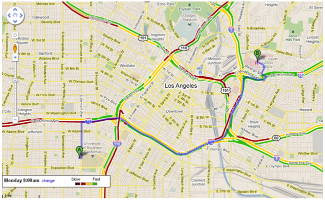
Stochastic Shortest Path (SP) Routing

$$\min \quad C_{\mathbf{a}}^{SP}(t) = \mathbb{E}[\sum_{\tau=1}^{t} \sum_{i \in E} a_i(\tau) D_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \text{ is an s-t path}$$

where $\{D_i(\tau)\}$ are unknown link costs.

Application: motivating example #1.



Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t} \sum_{i=1}^{N} a_i(\tau) X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

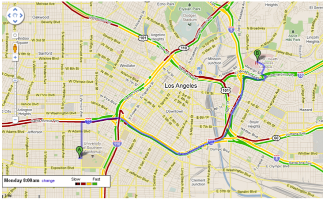
Stochastic Shortest Path (SP) Routing

$$\min \quad C_{\mathbf{a}}^{SP}(t) = \mathbb{E}[\sum_{\tau=1}^{t} \sum_{i \in E} a_i(\tau) D_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \text{ is an s-t path}$$

where $\{D_i(\tau)\}$ are unknown link costs.

Application: motivating example #1.



Similarly, $|E|$ edges

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

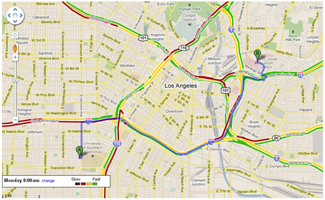
$\longrightarrow$

Stochastic Shortest Path (SP) Routing

$$\min \quad C_{\mathbf{a}}^{SP}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i\in E} a_i(\tau)D_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \text{ is an s-t path}$$

where $\{D_i(\tau)\}$ are unknown link costs.

Application: motivating example #1.



Similarly, $|E|$ edges$\rightarrow$ **only $|E|$ unknown variables!**

Application Examples

General Formulation

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $\{X_i(\tau)\}$ are unknown random variables.

$\longrightarrow$

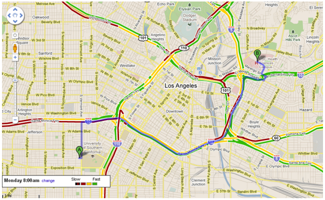
Stochastic Shortest Path (SP) Routing

$$\min \quad C_{\mathbf{a}}^{SP}(t) = \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i\in E} a_i(\tau)D_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \text{ is an s-t path}$$

where $\{D_i(\tau)\}$ are unknown link costs.

Application: motivating example #1.



Similarly, $|E|$ edges$\rightarrow$ **only $|E|$ unknown variables!**$\rightarrow$ **# paths (arms): exponential in $|E|$**

Challenges (1)

A K-armed classic MAB with single play ($K = |\mathcal{F}|$):

MAB with Linear Rewards

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

$\longrightarrow$

Classic MAB with single play

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t} Y_{\mathbf{a}}(\tau)]$$
$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $Y_{\mathbf{a}}(\tau) = \sum_{i=1}^{N} a_i(\tau)X_i(\tau)$, $K = |\mathcal{F}|$ (e.g. $K = P(N, M)$).

Challenges (1)

A K-armed classic MAB with single play ($K = |\mathcal{F}|$):

MAB with Linear Rewards

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t} \sum_{i=1}^{N} a_i(\tau) X_i(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned}$$

$\longrightarrow$

Classic MAB with single play

$$\begin{aligned} \max \quad & \mathbb{E}[\sum_{\tau=1}^{t} Y_{\mathbf{a}}(\tau)] \\ s.t. \quad & \mathbf{a}(\tau) \in \mathcal{F} \end{aligned}$$

where $Y_{\mathbf{a}}(\tau) = \sum_{i=1}^{N} a_i(\tau) X_i(\tau)$, $K = |\mathcal{F}|$ (e.g. $K = P(N, M)$).

- arms #: **exponentially** in N
  - Exponential storage.
  - Exponential computation time.
  - The upper bound of regret grows exponentially.

Challenges (1)

A K-armed classic MAB with single play ($K = |\mathcal{F}|$):

MAB with Linear Rewards

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t}\sum_{i=1}^{N} a_i(\tau)X_i(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

- **a more efficient and better algorithm is needed!**

$\longrightarrow$

Classic MAB with single play

$$\max \quad \mathbb{E}[\sum_{\tau=1}^{t} Y_{\mathbf{a}}(\tau)]$$

$$s.t. \quad \mathbf{a}(\tau) \in \mathcal{F}$$

where $Y_{\mathbf{a}}(\tau) = \sum_{i=1}^{N} a_i(\tau)X_i(\tau)$, $K = |\mathcal{F}|$ (e.g. $K = P(N, M)$).

- arms #: **exponentially** in N
  - Exponential storage.
  - Exponential computation time.
  - The upper bound of regret grows exponentially.

Challenges (2)

Challenges due to restless Markovian rewards:

Challenges (2)

Challenges due to restless Markovian rewards:

- transitions occur no matter played or not (every time slot)

Challenges (2)

Challenges due to restless Markovian rewards:

- transitions occur no matter played or not (every time slot)
- the current state while starting to play a Markov chain depends not only on the transition probabilities, but also on the policy

Challenges (2)

Challenges due to restless Markovian rewards:

- transitions occur no matter played or not (every time slot)
- the current state while starting to play a Markov chain depends not only on the transition probabilities, but also on the policy
- the policy design for the restless case is much more difficult

Outline

1. Introduction
   - Motivating Examples
   - General Formulation: MAB with Linear Rewards
   - Preliminaries
   - Problem Formulation
   - Applications
   - Challenges

2. Combinatorial Learning with Restless Markov Rewards (CLRMR)
   - Contribution
   - Proposed Algorithms
   - Analysis of Regret
   - An Extension
   - Simulations

3. Conclusion

Our Contribution

A new algorithm for this more general problem (parameterized by $\mathcal{F}$):

Combinatorial Learning with Restless Markov Rewards (CLRMR)

- only $O(N)$ storage
- achieves regret of $O(N^4 \ln t)$ (uniformly)
- polynomial running time whenever the underlying problem (which corresponds to $\mathcal{F}$) is in P (or admits approximation algorithms)

Our Contribution

A new algorithm for this more general problem (parameterized by $\mathcal{F}$):

Combinatorial Learning with Restless Markov Rewards (CLRMR)

- only $O(N)$ storage
- achieves regret of $O(N^4 \ln t)$ (uniformly)
- polynomial running time whenever the underlying problem (which corresponds to $\mathcal{F}$) is in P (or admits approximation algorithms)

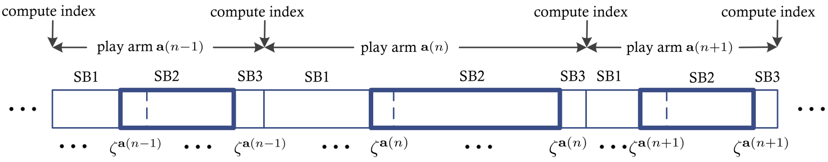
It is the first to show how to efficiently implement online learning for stochastic combinatorial network optimization when edge weights are dynamically evolving as restless Markovian processes.

Key ideas

1. only use info. from regenerative cycle (of the multidimensional Markov chain $\{X^{\mathbf{a}}(n)\}$)

Key ideas

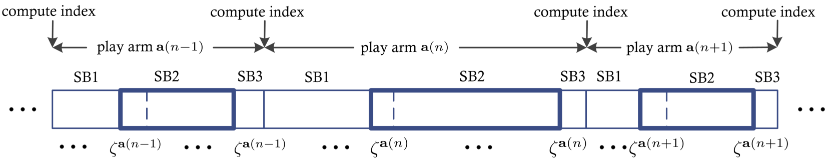
1. only use info. from regenerative cycle (of the multidimensional Markov chain $\{X^{\mathbf{a}}(n)\}$)



3 sub-blocks: SB1, SB2 and SB3

Key ideas

1. only use info. from regenerative cycle (of the multidimensional Markov chain $\{X^{\mathbf{a}}(n)\}$)

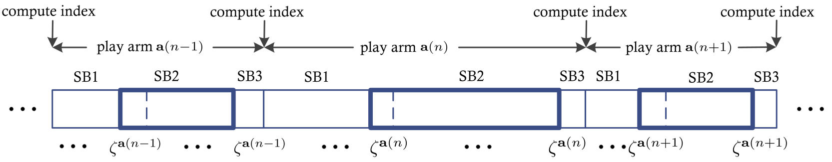


3 sub-blocks: SB1, SB2 and SB3

2. Utilize dependencies to improve efficiency

Key ideas

1. only use info. from regenerative cycle (of the multidimensional Markov chain $\{X^{\mathbf{a}}(n)\}$)



3 sub-blocks: SB1, SB2 and SB3

2. Utilize dependencies to improve efficiency

Storage

- store and use the observations for each MC
- for MC $\{X^i(n)\}$, 3 N-dimensional vectors:
  - $\bar{z}_2^i$: sample mean of observed values in SB2
  - m_2^i: # of observed times in SB2
  - ζ^i: a pre-specified state (to determine the regenerative cycle)

How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

```
// INITIALIZATION
t = 1, t_2 = 1;
∀i = 1, ···, N, m_2^i = 0, z̄_2^i = 0;
for b = 1 to N do
    t := t + 1, t_2 := t_2 + 1;
    Play any arm a such that b ∈ A_a; denote (x_i)_{i∈A_a} as the observed state vector for arm a;
    ∀i ∈ A_{a(n)}, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play arm a; denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    end while
end for
// MAIN LOOP
while 1 do
    // SB1 STARTS
    t := t + 1;
    Play an arm a which maximizes
        max_{a∈F} Σ_{i∈A_a} a_i ( z̄_2^i + sqrt( L ln t_2 / m_2^i ) );
    where L is a constant.
    Denote (x_i)_{i∈A_a} as the observed state vector;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
    end while
    // SB2 STARTS
    t_2 := t_2 + 1;
    ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    end while
    // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
    b := b + 1, t := t + 1;
end while
```

Maximization criterion in SB1:

$$\max_{\mathbf{a}\in\mathcal{F}} \sum_{i\in\mathcal{A}_{\mathbf{a}}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

Update rule:

$$\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r^i_{x_i}}{m_2^i + 1}, \quad m_2^i := m_2^i + 1$$

How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

// INITIALIZATION
$t = 1$, $t_2 = 1$;
$\forall i = 1, \cdots, N$, $m_2^i = 0$, $\bar{z}_2^i = 0$;
for $b = 1$ to N **do**
  $t := t + 1$, $t_2 := t_2 + 1$;
  Play any arm $\mathbf{a}$ such that $b \in \mathcal{A}_{\mathbf{a}}$; denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector for arm $\mathbf{a}$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play arm $\mathbf{a}$; denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
end for
// MAIN LOOP
while 1 **do**
  // SB1 STARTS
  $t := t + 1$;
  Play an arm $\mathbf{a}$ which maximizes

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_{\mathbf{a}}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

  where L is a constant.
  Denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
  end while
  // SB2 STARTS
  $t_2 := t_2 + 1$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
  // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
  $b := b + 1$, $t := t + 1$;
end while

Initialization: play arms s.t. each MC is observed at least once.

How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

```
// INITIALIZATION
t = 1, t_2 = 1;
∀i = 1, ··· , N, m_2^i = 0, z̄_2^i = 0;
for b = 1 to N do
    t := t + 1, t_2 := t_2 + 1;
    Play any arm a such that b ∈ A_a; denote (x_i)_{i∈A_a} as the observed state vector for arm a;
    ∀i ∈ A_{a(n)}, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play arm a; denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    end while
end for
// MAIN LOOP
while 1 do
    // SB1 STARTS
    t := t + 1;
    Play an arm a which maximizes
        max_{a∈F} Σ_{i∈A_a} a_i ( z̄_2^i + sqrt( L ln t_2 / m_2^i ) );
    where L is a constant.
    Denote (x_i)_{i∈A_a} as the observed state vector;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
    end while
    // SB2 STARTS
    t_2 := t_2 + 1;
    ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    end while
    // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
    b := b + 1, t := t + 1;
end while
```

Initialization: play arms s.t. each MC is observed at least once.

↓

Main loop:

How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

// INITIALIZATION
$t = 1$, $t_2 = 1$;
$\forall i = 1, \cdots, N$, $m_2^i = 0$, $\bar{z}_2^i = 0$;
for $b = 1$ to N **do**
  $t := t + 1$, $t_2 := t_2 + 1$;
  Play any arm $\mathbf{a}$ such that $b \in \mathcal{A}_{\mathbf{a}}$; denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector for arm $\mathbf{a}$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play arm $\mathbf{a}$; denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
end for
// MAIN LOOP
while 1 **do**
  // SB1 STARTS
  $t := t + 1$;
  Play an arm $\mathbf{a}$ which maximizes

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_{\mathbf{a}}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

  where L is a constant.
  Denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
  end while
  // SB2 STARTS
  $t_2 := t_2 + 1$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
  // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
  $b := b + 1$, $t := t + 1$;
end while

compute index
↓

...

Initialization: play arms s.t. each MC is observed at least once.

↓

Main loop:

//SB1

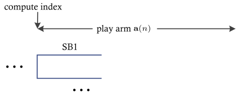
- decide which arm to play in this block:
  pick **a** which solves the maximization problem

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_{\mathbf{a}}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

```
// INITIALIZATION
t = 1, t_2 = 1;
∀i = 1, ··· , N, m_2^i = 0, z̄_2^i = 0;
for b = 1 to N do
    t := t + 1, t_2 := t_2 + 1;
    Play any arm a such that b ∈ A_a; denote (x_i)_{i∈A_a} as the observed state vector for arm a;
    ∀i ∈ A_{a(n)}, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; z̄_2^i := (z̄_2^i m_2^i + r_{x_i}^i)/(m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play arm a; denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r_{x_i}^i)/(m_2^i + 1), m_2^i := m_2^i + 1;
    end while
end for
// MAIN LOOP
while 1 do
    // SB1 STARTS
    t := t + 1;
    Play an arm a which maximizes
        max_{a∈F} Σ_{i∈A_a} a_i ( z̄_2^i + sqrt( L ln t_2 / m_2^i ) );
    where L is a constant.
    Denote (x_i)_{i∈A_a} as the observed state vector;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
    end while
    // SB2 STARTS
    t_2 := t_2 + 1;
    ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r_{x_i}^i)/(m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r_{x_i}^i)/(m_2^i + 1), m_2^i := m_2^i + 1;
    end while
    // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
    b := b + 1, t := t + 1;
end while
```



Initialization: play arms s.t. each MC is observed at least once.

↓

Main loop:

//SB1

- decide which arm to play in this block:
  pick **a** which solves the maximization problem

$$\max_{\mathbf{a}\in\mathcal{F}} \sum_{i\in\mathcal{A}_\mathbf{a}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

- keep playing **a**

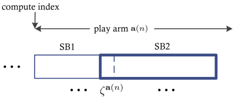
How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

// INITIALIZATION
$t = 1$, $t_2 = 1$;
$\forall i = 1, \cdots, N$, $m_2^i = 0$, $\bar{z}_2^i = 0$;
for $b = 1$ to N **do**
  $t := t + 1$, $t_2 := t_2 + 1$;
  Play any arm $\mathbf{a}$ such that $b \in \mathcal{A}_{\mathbf{a}}$; denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector for arm $\mathbf{a}$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play arm $\mathbf{a}$; denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
end for
// MAIN LOOP
while 1 **do**
  // SB1 STARTS
  $t := t + 1$;
  Play an arm $\mathbf{a}$ which maximizes

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_{\mathbf{a}}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

  where L is a constant.
  Denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
  end while
  // SB2 STARTS
  $t_2 := t_2 + 1$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}} \neq (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_{\mathbf{a}}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
  // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
  $b := b + 1$, $t := t + 1$;
end while



Initialization: play arms s.t. each MC is observed at least once.

↓

Main loop:

//SB1

- decide which arm to play in this block:
  pick **a** which solves the maximization problem

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_{\mathbf{a}}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

- keep playing **a**

//SB2

- when $\zeta^{\mathbf{a}} = (\zeta^i)_{i \in \mathcal{A}_{\mathbf{a}}}$ occurs, keep playing **a**, update $\bar{z}_2^i$, m_2^i after each play

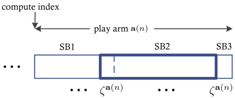
How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

// INITIALIZATION
$t = 1$, $t_2 = 1$;
$\forall i = 1, \cdots, N$, $m_2^i = 0$, $\bar{z}_2^i = 0$;
for $b = 1$ to N **do**
  $t := t + 1$, $t_2 := t_2 + 1$;
  Play any arm $\mathbf{a}$ such that $b \in \mathcal{A}_\mathbf{a}$; denote $(x_i)_{i \in \mathcal{A}_\mathbf{a}}$ as the observed state vector for arm $\mathbf{a}$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_\mathbf{a}} \neq (\zeta^i)_{i \in \mathcal{A}_\mathbf{a}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play arm $\mathbf{a}$; denote $(x_i)_{i \in \mathcal{A}_\mathbf{a}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
end for
// MAIN LOOP
while 1 **do**
  // SB1 STARTS
  $t := t + 1$;
  Play an arm $\mathbf{a}$ which maximizes

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_\mathbf{a}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

  where L is a constant.
  Denote $(x_i)_{i \in \mathcal{A}_\mathbf{a}}$ as the observed state vector;
  while $(x_i)_{i \in \mathcal{A}_\mathbf{a}} \neq (\zeta^i)_{i \in \mathcal{A}_\mathbf{a}}$ **do**
    $t := t + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_\mathbf{a}}$ as the observed state vector;
  end while
  // SB2 STARTS
  $t_2 := t_2 + 1$;
  $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  while $(x_i)_{i \in \mathcal{A}_\mathbf{a}} \neq (\zeta^i)_{i \in \mathcal{A}_\mathbf{a}}$ **do**
    $t := t + 1$, $t_2 := t_2 + 1$;
    Play an arm $\mathbf{a}$ and denote $(x_i)_{i \in \mathcal{A}_\mathbf{a}}$ as the observed state vector;
    $\forall i \in \mathcal{A}_{\mathbf{a}(n)}$, $\bar{z}_2^i := \frac{\bar{z}_2^i m_2^i + r_{x_i}^i}{m_2^i + 1}$, $m_2^i := m_2^i + 1$;
  end while
  // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
  $b := b + 1$, $t := t + 1$;
end while



Initialization: play arms s.t. each MC is observed at least once.

↓

Main loop:

//SB1

- decide which arm to play in this block:
  pick $\mathbf{a}$ which solves the maximization problem

$$\max_{\mathbf{a} \in \mathcal{F}} \sum_{i \in \mathcal{A}_\mathbf{a}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

- keep playing $\mathbf{a}$

//SB2

- when $\zeta^\mathbf{a} = (\zeta^i)_{i \in \mathcal{A}_\mathbf{a}}$ occurs, keep playing $\mathbf{a}$, update $\bar{z}_2^i$, m_2^i after each play

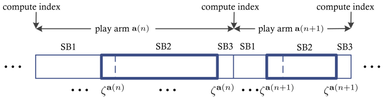
//SB3

- when $\zeta^\mathbf{a} = (\zeta^i)_{i \in \mathcal{A}_\mathbf{a}}$ occurs again, stop playing

How the CLRMR Algorithm Works

Algorithm 1 Combinatorial Learning with Restless Markov Reward (CLRMR)

```
// INITIALIZATION
t = 1, t_2 = 1;
∀i = 1, ···, N, m_2^i = 0, z̄_2^i = 0;
for b = 1 to N do
    t := t + 1, t_2 := t_2 + 1;
    Play any arm a such that b ∈ A_a; denote (x_i)_{i∈A_a} as the observed state vector for arm a;
    ∀i ∈ A_{a(n)}, let ζ^i be the first state observed for Markov chain i if ζ^i has never been set; z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play arm a; denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    end while
end for
// MAIN LOOP
while 1 do
    // SB1 STARTS
    t := t + 1;
    Play an arm a which maximizes
        max_{a∈F} Σ_{i∈A_a} a_i ( z̄_2^i + sqrt( L ln t_2 / m_2^i ) );
    where L is a constant.
    Denote (x_i)_{i∈A_a} as the observed state vector;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
    end while
    // SB2 STARTS
    t_2 := t_2 + 1;
    ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    while (x_i)_{i∈A_a} ≠ (ζ^i)_{i∈A_a} do
        t := t + 1, t_2 := t_2 + 1;
        Play an arm a and denote (x_i)_{i∈A_a} as the observed state vector;
        ∀i ∈ A_{a(n)}, z̄_2^i := (z̄_2^i m_2^i + r^i_{x_i}) / (m_2^i + 1), m_2^i := m_2^i + 1;
    end while
    // SB3 IS THE LAST PLAY IN THE WHILE LOOP. THEN A BLOCK COMPLETES.
    b := b + 1, t := t + 1;
end while
```



Initialization: play arms s.t. each MC is observed at least once.

↓

Main loop:

//SB1

- decide which arm to play in this block: pick **a** which solves the maximization problem

$$\max_{\mathbf{a}\in\mathcal{F}} \sum_{i\in\mathcal{A}_\mathbf{a}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L \ln t_2}{m_2^i}} \right);$$

- keep playing **a**

//SB2

- when $\zeta^{\mathbf{a}} = (\zeta^i)_{i\in\mathcal{A}_\mathbf{a}}$ occurs, keep playing **a**, update $\bar{z}_2^i$, m_2^i after each play

//SB3

- when $\zeta^{\mathbf{a}} = (\zeta^i)_{i\in\mathcal{A}_\mathbf{a}}$ occurs again, stop playing

Upper Bound of Regret

- Traditional approach:
  bound expected # times each non-optimal arm is played & sum over all arms → bound on regret

Upper Bound of Regret

- Traditional approach:
  bound expected # times each non-optimal arm is played & sum over all arms $\rightarrow$ bound on regret
- Bound is linear in # arms

Upper Bound of Regret

- Traditional approach:
  bound expected # times each non-optimal arm is played & sum over all arms → bound on regret
- Bound is linear in # arms
- But: in CLRMR, we have exponentially many arms!

Upper Bound of Regret

- Traditional approach:
  bound expected # times each non-optimal arm is played & sum over all arms → bound on regret
- Bound is linear in # arms
- But: in CLRMR, we have exponentially many arms!
- Can we do better?

Upper Bound of Regret

- Traditional approach:
  bound expected # times each non-optimal arm is played & sum over all arms → bound on regret
- Bound is linear in # arms
- But: in CLRMR, we have exponentially many arms!
- Can we do better?
- Yes! We prove a tighter bound: $O(N^4 \ln t)$ (or $O(N^3 L \ln t)$).

Upper Bound of Regret

Theorem

When using any constant $L \geq 56(H+1)S_{\max}^2 r_{\max}^2 \hat{\pi}_{\max}^2/\epsilon_{\min}$*, the regret of CLRMR is at most*

$$\mathfrak{R}^{CLRMR}(t) \leq Z_3 \ln t + Z_4$$

where

$$Z_3 = Z_1 + Z_5 \frac{4NLH^2 a_{\max}^2}{\Delta_{\min}^2}, \quad Z_4 = Z_2 + \gamma^*(\frac{1}{\pi_{\min}} + M_{\max} + 1) + Z_5(N + \frac{\pi NHS_{\max}}{3\pi_{\min}})$$

and

$$Z_1 = \Delta_{\max}\left(\frac{1}{\Pi_{\min}} + M_{\max} + 1\right)\frac{4NLH^2 a_{\max}^2}{\Delta_{\min}^2}, \quad Z_2 = \Delta_{\max}\left(\frac{1}{\Pi_{\min}} + M_{\max} + 1\right)\left(N + \frac{\pi NHS_{\max}}{3\pi_{\min}}\right),$$

$$Z_5 = \gamma_{\max}^{\triangle}(\frac{1}{\Pi_{\min}} + M_{\max} + 1 - \frac{1}{\pi_{\max}}) + \gamma^* M_{\max}^*$$

Notations:

- H : $\max_{\mathbf{a}} |\mathcal{A}_{\mathbf{a}}|$. Note that $H \leq N$
- $\hat{\pi}_x^i$: $\max\{\pi_x^i, 1-\pi_x^i\}$
- $\hat{\pi}_{\max}$: $\max_{i,x\in S^i} \hat{\pi}_x^i$
- $\pi_{\max}$: $\max_{i,x\in S^i} \pi_x^i$
- ϵ^i: eigenvalue gap, defined as $1-\lambda_2$, where λ_2 is the second largest eigenvalue of the multiplicative symmetrization of P^i
- $\epsilon_{\min}$: $\min_i \epsilon^i$
- $S_{\max}$: $\max_i |S^i|$
- $r_{\max}$: $\max_{i,x\in S^i} r_x^i$
- $a_{\max}$: $\max_{i\in\mathcal{A}_{\mathbf{a}},\mathbf{a}\in\mathcal{F}} a_i$
- $\Delta_{\mathbf{a}}$: $\gamma^* - \gamma^{\mathbf{a}}$
- $\Delta_{\min}$: $\min_{\gamma^{\mathbf{a}}\leq\gamma^*} \Delta_{\mathbf{a}}$
- $\Delta_{\max}$: $\max_{\gamma^{\mathbf{a}}\leq\gamma^*} \Delta_{\mathbf{a}}$
- $\Pi_z^{\mathbf{a}}$: steady state distribution for state z of $\{X^{\mathbf{a}}(n)\}$
- $\Pi_{\min}^{\mathbf{a}}$: $\min_{z\in S^{\mathbf{a}}} \Pi_z^{\mathbf{a}}$
- $\Pi_{\min}$: $\min_{\mathbf{a},z\in S^{\mathbf{a}}} \Pi_z^{\mathbf{a}}$
- $\gamma_{\max}^{\triangle}$: $\max_{\gamma^{\mathbf{a}}\leq\gamma^*} \gamma^{\mathbf{a}}$
- $M_{z_1,z_2}^{\mathbf{a}}$: mean hitting time of state z_2 starting from an initial state z_1 for $\{X^{\mathbf{a}}(n)\}$
- $M_{\max}^{\mathbf{a}}$: $\max_{z_1,z_2\in S^{\mathbf{a}}} M_{z_1,z_2}^{\mathbf{a}}$
- $M_{\max}$: $\max_{\gamma^{\mathbf{a}}\leq\gamma^*} M_{\max}^{\mathbf{a}}$

Upper Bound of Regret

Theorem

When using any constant $L \geq 56(H+1)S_{\max}^2 r_{\max}^2 \hat{\pi}_{\max}^2/\epsilon_{\min}$*, the regret of CLRMR is at most*

$$O(N^4 \ln t)$$

Notations:

- H : $\max_{\mathbf{a}} |\mathcal{A}_{\mathbf{a}}|$. Note that $H \leq N$
- $\hat{\pi}_x^i$: $\max\{\pi_x^i, 1-\pi_x^i\}$
- $\hat{\pi}_{\max}$: $\max_{i,x\in S^i} \hat{\pi}_x^i$
- $\pi_{\max}$: $\max_{i,x\in S^i} \pi_x^i$
- ϵ^i: eigenvalue gap, defined as $1-\lambda_2$, where λ_2 is the second largest eigenvalue of the multiplicative symmetrization of P^i
- $\epsilon_{\min}$: $\min_i \epsilon^i$
- $S_{\max}$: $\max_i |S^i|$
- $r_{\max}$: $\max_{i,x\in S^i} r_x^i$
- $a_{\max}$: $\max_{i\in\mathcal{A}_{\mathbf{a}},\mathbf{a}\in\mathcal{F}} a_i$
- $\Delta_{\mathbf{a}}$: $\gamma^* - \gamma^{\mathbf{a}}$
- $\Delta_{\min}$: $\min_{\gamma^{\mathbf{a}}\leq\gamma^*} \Delta_{\mathbf{a}}$
- $\Delta_{\max}$: $\max_{\gamma^{\mathbf{a}}\leq\gamma^*} \Delta_{\mathbf{a}}$
- $\Pi_z^{\mathbf{a}}$: steady state distribution for state z of $\{X^{\mathbf{a}}(n)\}$
- $\Pi_{\min}^{\mathbf{a}}$: $\min_{z\in S^{\mathbf{a}}} \Pi_z^{\mathbf{a}}$
- $\Pi_{\min}$: $\min_{\mathbf{a},z\in S^{\mathbf{a}}} \Pi_z^{\mathbf{a}}$
- $\gamma_{\max}^{\triangle}$: $\max_{\gamma^{\mathbf{a}}\leq\gamma^*} \gamma^{\mathbf{a}}$
- $M_{z_1,z_2}^{\mathbf{a}}$: mean hitting time of state z_2 starting from an initial state z_1 for $\{X^{\mathbf{a}}(n)\}$
- $M_{\max}^{\mathbf{a}}$: $\max_{z_1,z_2\in S^{\mathbf{a}}} M_{z_1,z_2}^{\mathbf{a}}$
- $M_{\max}$: $\max_{\gamma^{\mathbf{a}}\leq\gamma^*} M_{\max}^{\mathbf{a}}$

An extension of CLRMR

When (a bound of) $S_{\max}$, $r_{\max}$, $\hat{\pi}_{\max}$ or ϵ_{min} is unknown, L cannot be determined. What shall we do?

An extension of CLRMR

When (a bound of) $S_{\max}$, $r_{\max}$, $\hat{\pi}_{\max}$ or ϵ_{min} is unknown, L cannot be determined. What shall we do?

An extension of CLRMR: using any arbitrarily slowly diverging non-decreasing sequence $L(t)$ such that $L(t) \leq t$ for any t.
(replacing the maximization in CLRMR accordingly with

$$\max_{\mathbf{a} \in \mathcal{F}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L(n(t_2)) \ln t_2}{m_2^i}} \right)$$

where $n(t_2)$ is the time when total number of time slots spent in SB2 is t_2)

An extension of CLRMR

When (a bound of) $S_{\max}$, $r_{\max}$, $\hat{\pi}_{\max}$ or ϵ_{min} is unknown, L cannot be determined. What shall we do?

An extension of CLRMR: using any arbitrarily slowly diverging non-decreasing sequence $L(t)$ such that $L(t) \leq t$ for any t.
(replacing the maximization in CLRMR accordingly with

$$\max_{\mathbf{a} \in \mathcal{F}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L(n(t_2)) \ln t_2}{m_2^i}} \right)$$

where $n(t_2)$ is the time when total number of time slots spent in SB2 is t_2)

Theorem

The expected regret under the CLRMR policy with using $L(t)$ is at most

$$\mathfrak{R}^{CLRMR-LN}(t) \leq Z_6 L(t) \ln t + Z_7 \qquad (2)$$

where Z_6 and Z_7 are constants.

An extension of CLRMR

When (a bound of) $S_{\max}$, $r_{\max}$, $\hat{\pi}_{\max}$ or ϵ_{min} is unknown, L cannot be determined. What shall we do?

An extension of CLRMR: using any arbitrarily slowly diverging non-decreasing sequence $L(t)$ such that $L(t) \leq t$ for any t.
(replacing the maximization in CLRMR accordingly with

$$\max_{\mathbf{a} \in \mathcal{F}} a_i \left(\bar{z}_2^i + \sqrt{\frac{L(n(t_2)) \ln t_2}{m_2^i}} \right)$$

where $n(t_2)$ is the time when total number of time slots spent in SB2 is t_2)

Theorem

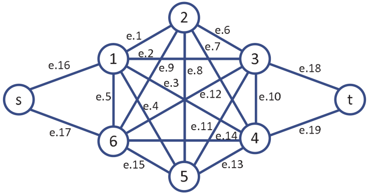
The expected regret under the CLRMR policy with using $L(t)$ is at most

$$O(N^3 L(t) \ln t)$$

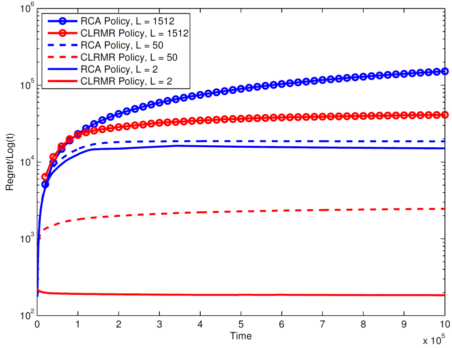
Simulation Results (1)

Application: Stochastic Shortest Path

- 19 links, 260 acyclic paths



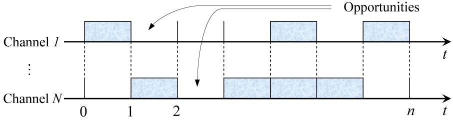
Link	p_{01}, p_{10}	Link	p_{01}, p_{10}	Link	p_{01}, p_{10}
e.1	0.2, 0.8	e.8	0.3, 0.8	e.15	0.1, 0.8
e.2	0.3, 0.9	e.9	0.1, 0.9	e.16	0.8, 0.1
e.3	0.2, 0.7	e.10	0.9, 0.1	e.17	0.2, 0.7
e.4	0.7, 0.1	e.11	0.3, 0.8	e.18	0.9, 0.1
e.5	0.3, 0.9	e.12	0.2, 0.7	e.19	0.3, 0.8
e.6	0.2, 0.7	e.13	0.8, 0.1		
e.7	0.2, 0.8	e.14	0.4, 0.8		

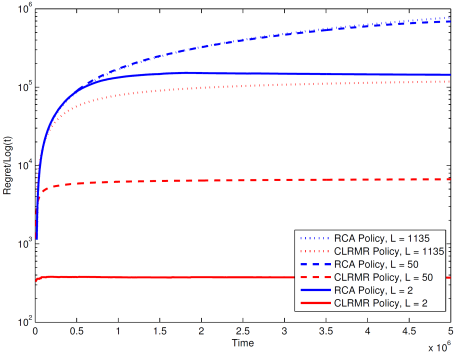


Simulation Results (2)

Application: Channel Allocations in CRN

- 9 orthogonal channels, 5 secondary users





	ch.1	ch.2	ch.3	ch.4	ch.5	ch.6	ch.7	ch.8	ch.9
u.1	0.5,0.6	0.2,0.7	0.2,0.9	0.8,0.1	0.2,0.7	0.3,0.7	0.2,0.9	0.2,0.7	0.1,0.9
u.2	0.3,0.8	0.1,0.9	0.2,0.8	0.3,0.7	0.3,0.6	0.2,0.8	0.4,0.7	0.2,0.8	0.9,0.2
u.3	0.8,0.1	0.2,0.7	0.3,0.7	0.2,0.8	0.5,0.6	0.2,0.7	0.2,0.7	0.2,0.8	0.1,0.9
u.4	0.3,0.9	0.2,0.8	0.2,0.9	0.4,0.6	0.9,0.2	0.2,0.9	0.2,0.9	0.2,0.9	0.2,0.9
u.5	0.5,0.6	0.2,0.7	0.3,0.9	0.2,0.7	0.5,0.5	0.2,0.7	0.8,0.1	0.3,0.9	0.3,0.9

Outline

1 Introduction
- Motivating Examples
- General Formulation: MAB with Linear Rewards
- Preliminaries
- Problem Formulation
- Applications
- Challenges

2 Combinatorial Learning with Restless Markov Rewards (CLRMR)
- Contribution
- Proposed Algorithms
- Analysis of Regret
- An Extension
- Simulations

3 Conclusion

Conclusion

More Works on MAB with Linear Rewards:

Problems	Random Process	Proposed Algorithms	Regret Bound*
MAB with Linear Rewards	i.i.d.	LLR	$O(N^4 \ln t)$
		LLR-K	$O(N^4 \ln t)$
		LLR with β-approximation	$O(N^4 \ln t)^\natural$

Notes:

$*$. Upper bounds on regret are achieved uniformly.

$\natural$. β-approximation regret.

Conclusion

More Works on MAB with Linear Rewards:

Problems	Random Process	Proposed Algorithms	Regret Bound*
MAB with Linear Rewards	i.i.d.	LLR	$O(N^4 \ln t)$
		LLR-K	$O(N^4 \ln t)$
		LLR with β-approximation	$O(N^4 \ln t)^\natural$
MAB with Linear Rewards	Rested Markovian	MLMR	$O(N^4 \ln t)^\sharp$
	Rested Markovian		$O(L(t)N^3 \ln t)^\dagger$

Notes:

*. Upper bounds on regret are achieved uniformly.

♮. β-approximation regret.

♯. weak regret; an upper bound on L is known.

†. $L(t)$ is any arbitrarily slowly diverging non-decreasing sequence.

Conclusion

More Works on MAB with Linear Rewards:

Problems	Random Process	Proposed Algorithms	Regret Bound*
MAB with Linear Rewards	i.i.d.	LLR	$O(N^4 \ln t)$
		LLR-K	$O(N^4 \ln t)$
		LLR with β-approximation	$O(N^4 \ln t)^{\natural}$
MAB with Linear Rewards	Rested Markovian	MLMR	$O(N^4 \ln t)^{\sharp}$
	Rested Markovian		$O(L(t)N^3 \ln t)^{\dagger}$
MAB with Linear Rewards	Restless Markovian	CLRMR	$O(N^4 \ln t)^{\sharp}$
	Restless Markovian		$O(L(t)N^3 \ln t)^{\dagger}$

Notes:

$*$. Upper bounds on regret are achieved uniformly.

$\natural$. β-approximation regret.

$\sharp$. weak regret; an upper bound on L is known.

$\dagger$. $L(t)$ is any arbitrarily slowly diverging non-decreasing sequence.

Conclusion

More Works on MAB with Linear Rewards:

Problems	Random Process	Proposed Algorithms	Regret Bound*
MAB with Linear Rewards	i.i.d.	LLR	$O(N^4 \ln t)$
		LLR-K	$O(N^4 \ln t)$
		LLR with β-approximation	$O(N^4 \ln t)^{\natural}$
MAB with Linear Rewards	Rested Markovian	MLMR	$O(N^4 \ln t)^{\sharp}$
	Rested Markovian		$O(L(t) N^3 \ln t)^{\dagger}$
MAB with Linear Rewards	Restless Markovian	CLRMR	$O(N^4 \ln t)^{\sharp}$
	Restless Markovian		$O(L(t) N^3 \ln t)^{\dagger}$

Notes:
*. Upper bounds on regret are achieved uniformly.
$\natural$. β-approximation regret.
$\sharp$. weak regret; an upper bound on L is known.
$\dagger$. $L(t)$ is any arbitrarily slowly diverging non-decreasing sequence.

Papers and Collaborators:

- SECON'12, DySPAN'10, IEEE/ACM Trans. Networking, Globecom'11, Machine Learning (under submission), Infocom'12 (mini-conf), arXiv(under submission)
- joint work with Bhaskar Krishnamachari, Mingyan Liu, Rahul Jain.

Conclusion (2)

Broad applications:

- Sensor Networks
- Cognitive Radio Networks
- Web Search
- Internet Advertising
- Energy Distribution Networks
- Social Economical Networks
- ......

Thanks!

ygai@usc.edu