Combinatorial Pure Exploration in Multi-Armed Bandits

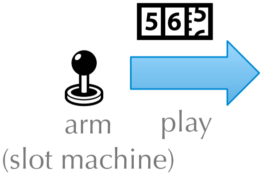
Shouyuan Chen[1] Tian Lin[2] Irwin King[1] Michael R. Lyu[1] Wei Chen[3]

[1] CUHK [2] Tsinghua University [3] Microsoft Research Asia

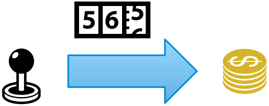
Single-armed bandit

arm
(slot machine)

Single-armed bandit



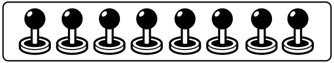
Single-armed bandit



sampled independently from
an **unknown** distribution
(reward distribution)

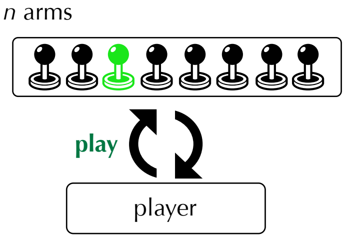
Multi-armed bandit

n arms



1. each arm has an **unknown** reward distribution
2. the reward distributions can be **different**.

Multi-armed bandit



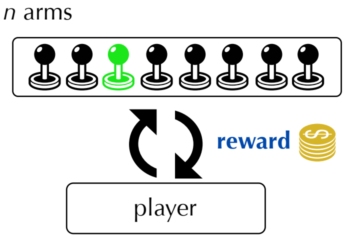
a game on multiple rounds...

rules

for round $t = 1, \ldots, T$

- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit



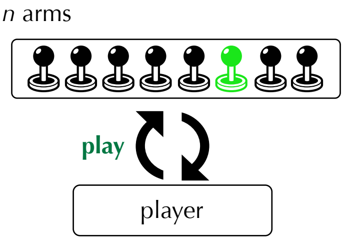
a game on multiple rounds...

rules

for round $t = 1, \dots, T$

- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit



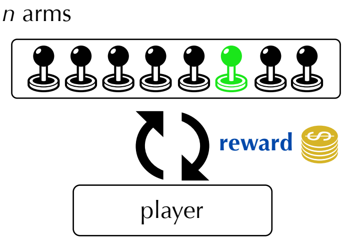
a game on multiple rounds...

rules

for round $t = 1, \ldots, T$

- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit



a game on multiple rounds...

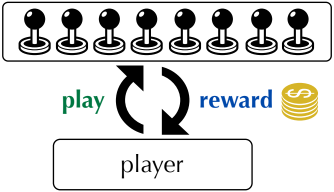
rules

for round $t = 1, \ldots, T$

- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit

n arms



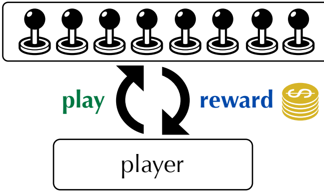
rules

for round $t = 1, \ldots, T$

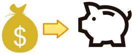
- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit

n arms



in the end...

take all rewards 

goal: maximize the cumulative reward

exploitation v.s. exploration

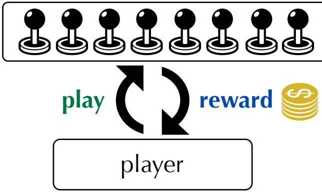
rules

for round $t = 1, \ldots, T$

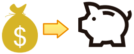
- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit

n arms



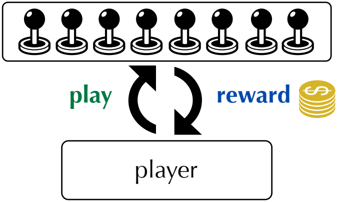
in the end...

take all rewards 

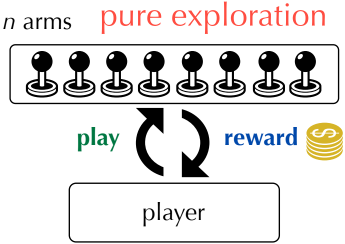
goal: maximize the cumulative reward

exploitation v.s. exploration

n arms pure exploration



Multi-armed bandit



rules

for round $t = 1, \ldots, T$

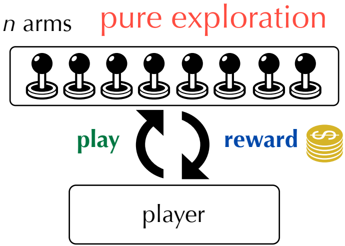
- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

Multi-armed bandit

rules

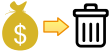
for round $t = 1, \ldots, T$

- plays arm $i_t \in [n]$
- receives reward $X_{it} \sim \phi_i$

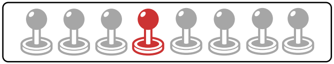


in the end...

(1) forfeit all rewards
(2) output **1** arm

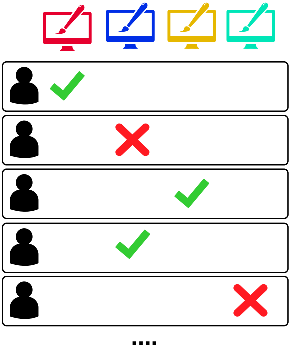


goal: find the single **best arm**
(largest expected reward)



Pure Exploration of MAB

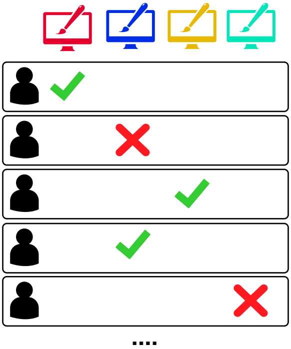
A/B testing, clinical trials, wireless network, crowdsourcing, ...



- n arms = n variants
- play arm i = a page view on the i-th variant
- reward = a click on the ads
- finding the best arm = finding the variant with the highest average ads clicks

Pure Exploration of MAB

A/B testing, clinical trials, wireless network, crowdsourcing, ...



- n arms = n variants
- play arm i = a page view on the i-th variant
- reward = a click on the ads
- finding the best arm = finding the variant with the highest average ads clicks

extensions

- find top-k arms [KS10,GGL12,KTPS12,BWV13,KK13,ZCL14]
- find top arms in disjoint groups of arms [GGLB11,GGL12,BWV13]

Pure exploration: two settings

fixed budget

- play for T rounds.
- report the best arm after finished.
- **goal**: minimize the probability of error $\Pr[\text{out} \neq i_*]$

fixed confidence

- play for any number of rounds.
- report the best arm after finished
- guarantee that probability of error $\Pr[\text{out} \neq i_*] < \delta$.
- **goal**: minimize the number of rounds (sample complexity).

Combinatorial Pure Exploration of MAB

Combinatorial Pure Exploration (CPE)

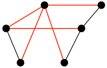
- play one arm at each round
- find the optimal **set** of arms M_* satisfying certain constraint

$$M_* = \arg\max_{M \in \mathcal{M}} \sum_{i \in M} w(i)$$

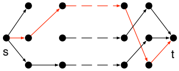
  - maximize the sum of expected rewards of arms in the set

size-k-sets

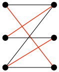
spanning trees



paths

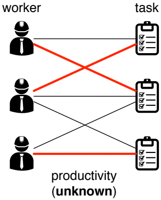


matchings



Motivating Examples

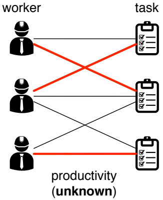
- matching



Goal:
1) estimate the productivities from tests.
2) find the optimal **1-1 assignment**.

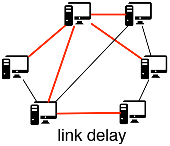
Motivating Examples

- matching



Goal:
1) estimate the productivities from tests.
2) find the optimal **1-1 assignment**.

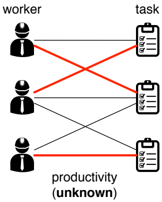
- spanning trees and paths



Goal:
1) estimate the delays from measurements
2) find the minimum spanning tree
or shortest path.

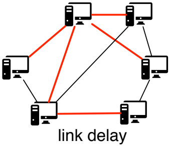
Motivating Examples

- matching



Goal:
1) estimate the productivities from tests.
2) find the optimal **1-1 assignment**.

- spanning trees and paths



Goal:
1) estimate the delays from measurements
2) find the minimum spanning tree
or shortest path.

- size-k-sets
  - finding the top-k arms.

Our Results

- algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.

Our Results

- algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.
- upper bounds
  - sample complexity / probability of error.

Our Results

- algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.
- upper bounds
  - sample complexity / probability of error.
- lower bound
  - algorithms are optimal (within log factors) for many types of $\mathcal{M}$ (in particular, bases of a matroid).

Our Results

- algorithms
  - two general learning algorithms for a wide range of $\mathcal{M}$.
- upper bounds
  - sample complexity / probability of error.
- lower bound
  - algorithms are optimal (within log factors) for many types of $\mathcal{M}$ (in particular, bases of a matroid).
- compared with existing work
  - the first lower bound for the top-k problem
  - the first upper and lower bounds for other combinatorial constraints.

CLUCB: Fixed confidence algorithm

input

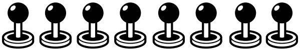
- confidence: $\delta \in (0, 1)$
- access to a maximization oracle: $\text{Oracle}(\cdot) : \mathbb{R}^n \to \mathcal{M}$
  - $\text{Oracle}(v) = \max_{M \in \mathcal{M}} \sum_{i \in M} v(i)$ for weights $v \in \mathbb{R}^n$

output

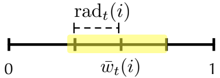
- a set of arms: $M \in \mathcal{M}$.

CLUCB: Fixed confidence algorithm

all arms



maintain: for all i and t



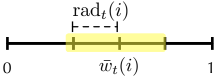
notations

- for each arm $i \in [n]$ in each round t
  - empirical mean: $\bar{w}_t(i)$
  - confidence radius: $\mathrm{rad}_t(i)$ (proportional to $1/\sqrt{n_t(i)}$)

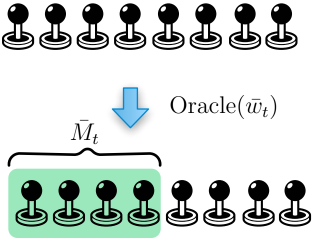
CLUCB: Fixed confidence algorithm

all arms

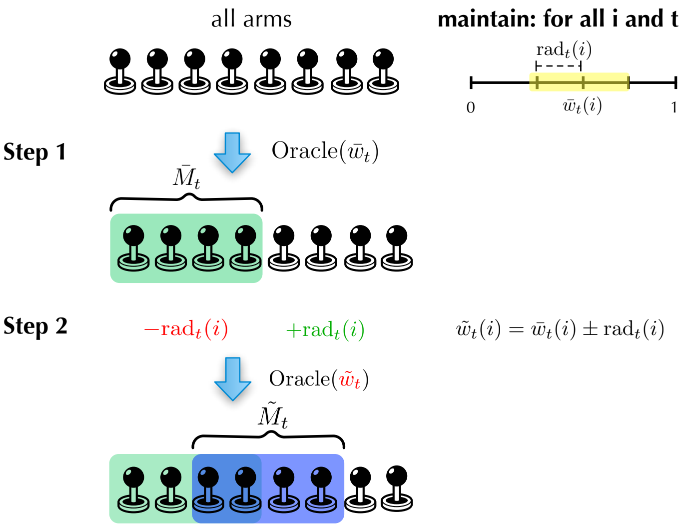
maintain: for all i and t



Step 1

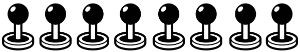


CLUCB: Fixed confidence algorithm

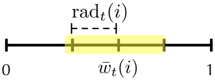


CLUCB: Fixed confidence algorithm

all arms

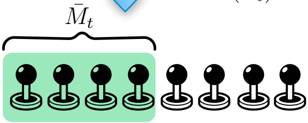


maintain: for all i and t



Step 1

$\mathrm{Oracle}(\bar{w}_t)$

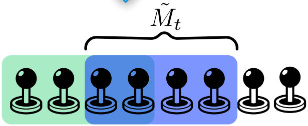


Step 2

$-\mathrm{rad}_t(i)$ $+\mathrm{rad}_t(i)$

$$\tilde{w}_t(i) = \bar{w}_t(i) \pm \mathrm{rad}_t(i)$$

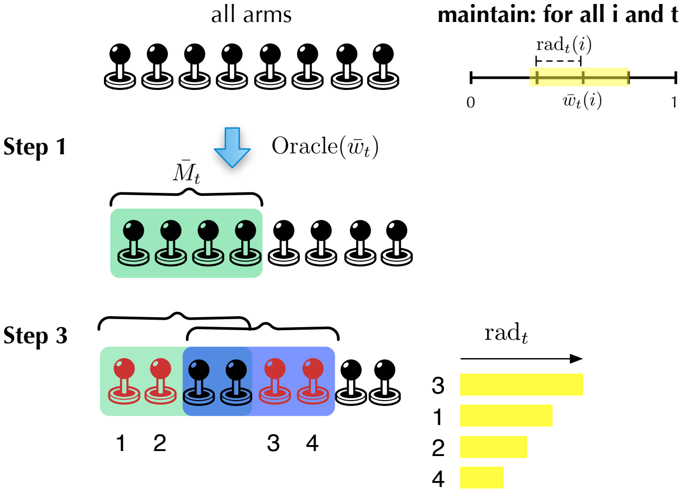
$\mathrm{Oracle}(\tilde{w}_t)$



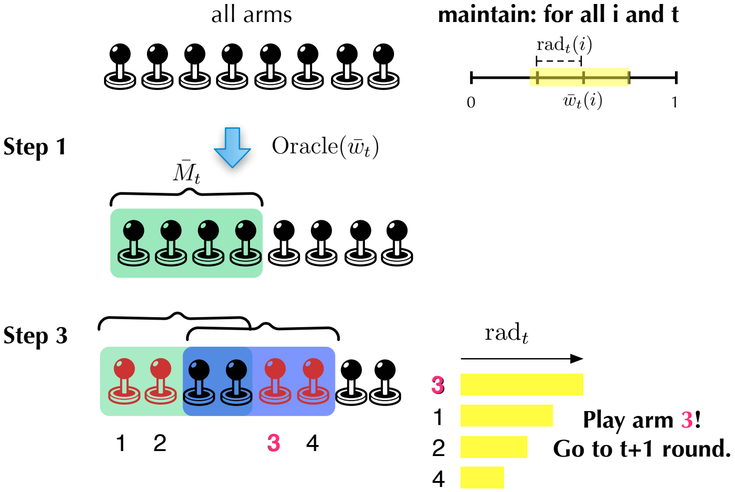
If: $\bar{M}_t = \tilde{M}_t$

Then: Stop and output $\bar{M}_t$

CLUCB: Fixed confidence algorithm



CLUCB: Fixed confidence algorithm



CLUCB: Sample Complexity

Our sample complexity bound depends on two quantities.

- **H**: depends on expected rewards
- width($\mathcal{M}$): depends on the structure of $\mathcal{M}$

CLUCB: Sample Complexity

Theorem (Upper bound)

With probability at least $1-\delta$, CLUCB algorithm:

1. *correctly outputs the optimal set M_**
2. *uses at most $O(\mathrm{width}(\mathcal{M})^2\mathbf{H}\log(n\mathbf{H}/\delta))$ rounds.*

Our sample complexity bound depends on two quantities.

- $\mathbf{H}$: depends on expected rewards
- $\mathrm{width}(\mathcal{M})$: depends on the structure of $\mathcal{M}$

Results at a glance

Theorem (Upper bound)

With probability at least $1-\delta$, CLUCB algorithm:

1. *outputs the optimal set* $M_* \triangleq \arg\max_{M \in \mathcal{M}} w(M)$.
2. *uses at most* $O(\mathrm{width}(\mathcal{M})^2 \mathbf{H} \log(n\mathbf{H}/\delta))$ *rounds.*

Theorem (Lower bound)

Given any expected rewards, any δ-correct algorithm must use at least $\Omega(\mathbf{H} \log(1/\delta))$ rounds. (An algorithm $\mathbb{A}$ is δ-correct algorithm, if $\mathbb{A}$'s probability of error is at most δ for any instances)

Example (Sample Complexities)

- k**-sets, spanning trees, bases of a matroid**: $\tilde{O}(\mathbf{H})$ optimal!
- **matchings, paths (in DAG)**: $\tilde{O}(|V|^2\mathbf{H})$.
- **in general**: $\tilde{O}(n^2\mathbf{H})$

$\mathbf{H}$ and gaps

- Δ_e: gap of arm $e \in [n]$

$$\Delta_e = \begin{cases} w(M_*) - \max_{M \in \mathcal{M}: e \in M} w(M) & \text{if } e \notin M_*, \\ w(M_*) - \max_{M \in \mathcal{M}: e \notin M} w(M) & \text{if } e \in M_* \end{cases}$$

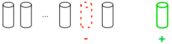
  - stability of the optimality of M_* wrt. arm e.
- $\mathbf{H} = \sum_{e \in [n]} \Delta_e^{-2}$
  - for the top-K problem: recover the previous definition of $\mathbf{H}$.

Width and exchange class

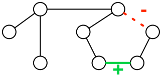
Intuitions

- we need a unifying method of analyzing different $\mathcal{M}$
  - an exchange class is a "proxy" for the structure of $\mathcal{M}$.
- an exchange class is a collection of "patches" that are used to interpolate between valid sets.

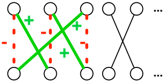
size-k-sets



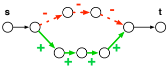
spanning tree



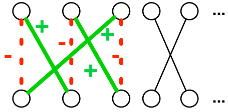
matching



path



Width and exchange class



definition

width($\mathcal{B}$): the size of the largest "patch"

$$\mathrm{width}(\mathcal{B}) = \max_{(b_+, b_-) \in \mathcal{B}} |b_+| + |b_-|.$$

width($\mathcal{M}$): the width of the "thinnest" exchange class

$$\mathrm{width}(\mathcal{M}) = \min_{\mathcal{B} \in \mathrm{Exchange}(\mathcal{M})} \mathrm{width}(\mathcal{B}),$$

Example (Widths)

- ***k*-sets, spanning trees, bases of a matroid**: $\mathrm{width}(\mathcal{M}) = 2$.
- **matchings, paths (in DAG)**: $\mathrm{width}(\mathcal{M}) = O(|V|)$.
- **in general**: $\mathrm{width}(\mathcal{M}) \leq n$

CSAR: Fixed budget algorithm

input

- budget: T (play for at most T rounds)
- access to a maximization oracle

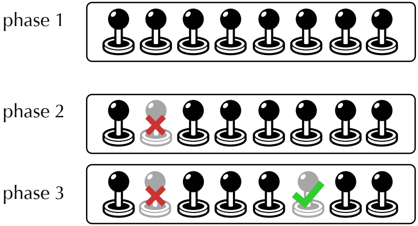
output

- a set of arms: $M \in \mathcal{M}$.

overview:

- break the T rounds into n phases.

CSAR: Fixed budget algorithm



in each phase (n phases in total):

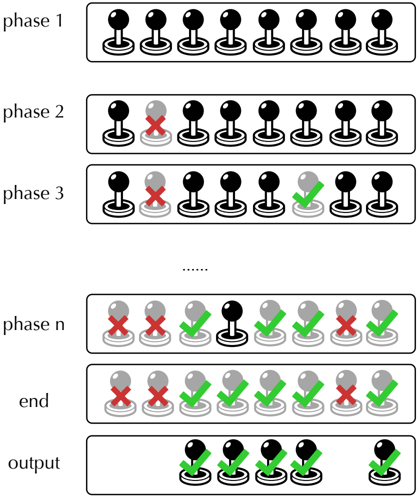
- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.

active: neither accepted nor rejected. require more samples

accepted: include in the output

rejected: exclude from the output

CSAR: Fixed budget algorithm



in each phase (n phases in total):

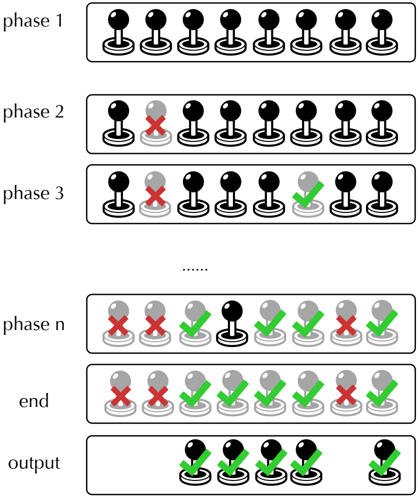
- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.

active: neither accepted nor rejected. require more samples

accepted: include in the output

rejected: exclude from the output

CSAR: Fixed budget algorithm



in each phase (*n* phases in total):

- 1 arm is accepted or rejected.
- active arms are sampled for a same number of times.

active: neither accepted nor rejected. require more samples

accepted: include in the output

rejected: exclude from the output

problem: which arm to accept or reject?

CSAR: Fixed budget algorithm

problem: which arm to accept or reject?

- accept/reject the arm with the largest **empirical gap**.

$$\bar{\Delta}_e = \begin{cases} \bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \notin \bar{M}_t, \\ \bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \notin M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t \end{cases}$$

  - $\mathcal{M}_t = \{M : M \in \mathcal{M}, A_t \subseteq M, B_t \cap M = \emptyset\}$.
  - A_t: accepted arms, B_t: rejected arms (up to phase t).

CSAR: Fixed budget algorithm

problem: which arm to accept or reject?

- accept/reject the arm with the largest **empirical gap**.

$$\bar{\Delta}_e = \begin{cases} \bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \notin \bar{M}_t, \\ \bar{w}_t(\bar{M}_t) - \max_{M \in \mathcal{M}_t : e \notin M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t \end{cases}$$

  - $\mathcal{M}_t = \{M : M \in \mathcal{M}, A_t \subseteq M, B_t \cap M = \emptyset\}$.
  - A_t: accepted arms, B_t: rejected arms (up to phase t).
  - $\bar{\Delta}_e$ can be computed using a maximization oracle.

CSAR: Fixed budget algorithm

problem: which arm to accept or reject?

- accept/reject the arm with the largest **empirical gap**.

$$\bar{\Delta}_e = \begin{cases} \bar{w}_t(\bar{M}_t) - \max\limits_{M \in \mathcal{M}_t : e \in M} \bar{w}_t(M) & \text{if } e \notin \bar{M}_t, \\ \bar{w}_t(\bar{M}_t) - \max\limits_{M \in \mathcal{M}_t : e \notin M} \bar{w}_t(M) & \text{if } e \in \bar{M}_t \end{cases}$$

  - $\mathcal{M}_t = \{M : M \in \mathcal{M}, A_t \subseteq M, B_t \cap M = \emptyset\}$.
  - A_t: accepted arms, B_t: rejected arms (up to phase t).
  - $\bar{\Delta}_e$ can be computed using a maximization oracle.

- recall the (unknown) **gap** of arm e:

$$\Delta_e = \begin{cases} w(M_*) - \max_{M \in \mathcal{M} : e \in M} w(M) & \text{if } e \notin M_*, \\ w(M_*) - \max_{M \in \mathcal{M} : e \notin M} w(M) & \text{if } e \in M_* \end{cases}$$

CSAR: Probability of error

Theorem (Probability of error of CSAR)

Given any budget $T > n$, CSAR correctly outputs the optimal set M_* with probability at least

$$1 - 2^{\tilde{O}\left(-\frac{T}{\mathrm{width}(\mathcal{M})^2 \mathbf{H}}\right)}$$

and uses at most T rounds.

CSAR: Probability of error

Theorem (Probability of error of CSAR)

Given any budget $T > n$, CSAR correctly outputs the optimal set M_* with probability at least

$$1 - 2^{\tilde{O}\left(-\frac{T}{\text{width}(\mathcal{M})^2 \mathbf{H}}\right)}$$

and uses at most T rounds.

Remark: To guarantee a constant probability of error of δ, both CSAR and CLUCB need $T = \tilde{O}(\text{width}(\mathcal{M})^2 \mathbf{H})$ rounds.

Summary

- combinatorial pure exploration: a general framework that covers many pure exploration problems in MAB.
  - find top-k arms, optimal spanning trees, matchings or paths.
- two general algorithms for the problem
  - only need a maximization oracle for $\mathcal{M}$.
  - comparable performance guarantees.
- our algorithm is optimal (up to log factors) for matroids.
  - including k-sets and spanning trees.

Future work

- tighten the bounds for matching, paths and other combinatorial constraints
- support approximate maximization oracles
- non-linear rewards

Thank you!

Exchange class: Formal definition

Exchange set

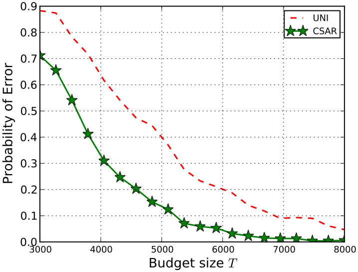
An **exchange set** b is an ordered pair of disjoint sets $b = (b_+, b_-)$ where $b_+ \cap b_- = \emptyset$ and $b_+, b_- \subseteq [n]$.
Let M be any set. We also define two operators:

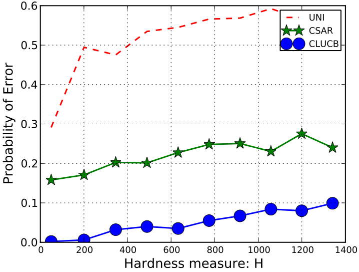
- $M \oplus b \triangleq M \backslash b_- \cup b_+$.
- $M \ominus b \triangleq M \backslash b_+ \cup b_-$.

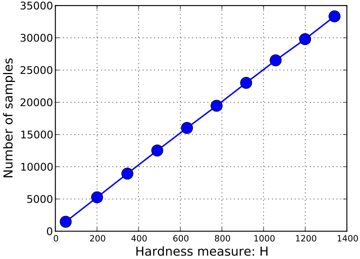
Exchange class

We call a collection of exchange sets $\mathcal{B}$ an **exchange class for** $\mathcal{M}$ if $\mathcal{B}$ satisfies the following property. For any $M, M' \in \mathcal{M}$ such that $M \neq M'$ and for any $e \in (M \backslash M')$, there exists an exchange set $(b_+, b_-) \in \mathcal{B}$ which satisfies five constraints: **(a)** $e \in b_-$, **(b)** $b_+ \subseteq M' \backslash M$, **(c)** $b_- \subseteq M \backslash M'$, **(d)** $(M \oplus b) \in \mathcal{M}$ and **(e)** $(M' \ominus b) \in \mathcal{M}$.

Experiments of CPE







Width and exchange class

definition

Let $\mathcal{B}$ be an exchange class.

$$\text{width}(\mathcal{B}) = \max_{(b_+, b_-) \in \mathcal{B}} |b_+| + |b_-|.$$

Let $\text{Exchange}(\mathcal{M})$ denote the family of all possible exchange classes for $\mathcal{M}$. We define the width of $\mathcal{M}$ to be the width of the thinnest exchange class

$$\text{width}(\mathcal{M}) = \min_{\mathcal{B} \in \text{Exchange}(\mathcal{M})} \text{width}(\mathcal{B}),$$

where $\text{Exchange}(\mathcal{M})$ is the family of all possible exchange classes for $\mathcal{M}$.