Batched Dueling Bandits

Arpit Agarwal [1]   Rohan Ghuge [2]   Viswanath Nagarajan [2]

[1]Data Science Institute, Columbia University.

[2]Department of Industrial and Operations Engineering, University of Michigan.

July 18, 2022

Motivation I: Web-Search Ranking

https://www.expedia.com › ... › Maryland
Top Hotels in Baltimore, MD from $76 - Expedia
Check Baltimore (and vicinity) hotel prices · Canopy by Hilton Baltimore Harbor Point · DoubleTree **Hotel Baltimore** - BWI Airport · Baltimore Marriott Waterfront.
Accommodation: 673 hotels   Highest Price: $182
Number of reviews: 9622

https://www.kayak.com › ... › Hotels in Maryland
16 Best Hotels in Baltimore. Hotels from $59/night - KAYAK
Baltimore hotels near The **Baltimore** Convention Center ; La Quinta Inn & Suites by Wyndham **Baltimore** Downtown - **Baltimore** - Bedroom. La Quinta Inn & Suites by ...
Average price (weekend night): $200   Low season: May
Average price (weeknight): $175   High season: March

https://travel.usnews.com › Hotels › USA
25 Best Hotels in Baltimore, MD - US News Travel
Four Seasons **Hotel Baltimore** · Sagamore Pendry Baltimore · Kimpton Hotel Monaco Baltimore Inner Harbor · Royal Sonesta Harbor Court Baltimore · Hotel Indigo ...

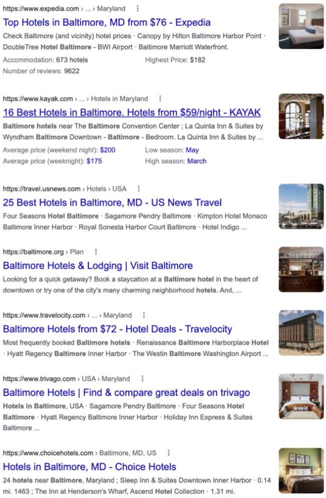
https://baltimore.org › Plan
Baltimore Hotels & Lodging | Visit Baltimore
Looking for a quick getaway? Book a staycation at a **Baltimore hotel** in the heart of downtown or try one of the city's many charming neighborhood **hotels**. And, ...

https://www.travelocity.com › ... › Maryland
Baltimore Hotels from $72 - Hotel Deals - Travelocity
Most frequently booked **Baltimore hotels** · Renaissance **Baltimore** Harborplace **Hotel** · Hyatt Regency **Baltimore** Inner Harbor · The Westin **Baltimore** Washington Airport ...

https://www.trivago.com › USA › Maryland
Baltimore Hotels | Find & compare great deals on trivago
Hotels in Baltimore, USA · Sagamore Pendry Baltimore · Four Seasons **Hotel Baltimore** · Hyatt Regency Baltimore Inner Harbor · Holiday Inn Express & Suites Baltimore ...

https://www.choicehotels.com › Baltimore, MD, US
Hotels in Baltimore, MD - Choice Hotels
24 **hotels** near **Baltimore**, Maryland ; Sleep Inn & Suites Downtown Inner Harbor · 0.14 mi. 1463 ; The Inn at Henderson's Wharf, Ascend **Hotel** Collection · 1.31 mi.

Motivation I: Web-Search Ranking

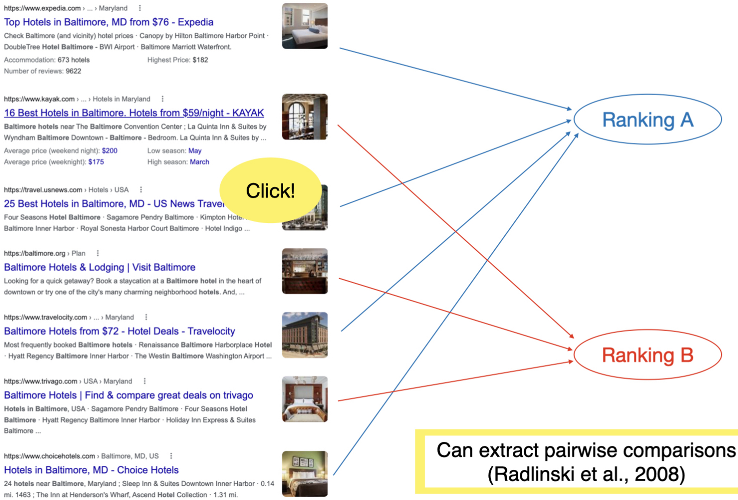


Can extract pairwise comparisons (Radlinski et al., 2008)

Motivation I: Web-Search Ranking

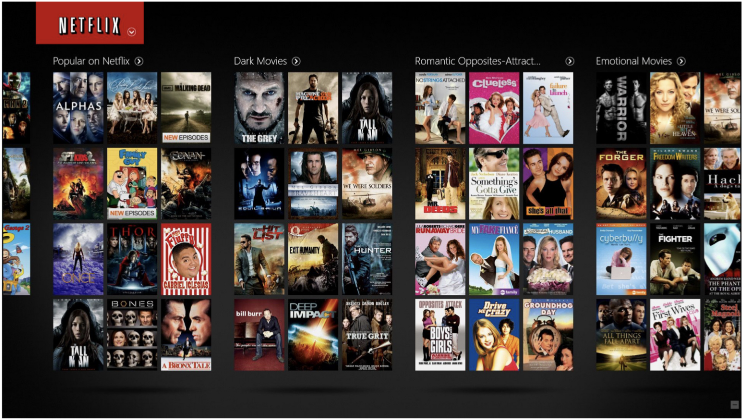
Can extract pairwise comparisons
(Radlinski et al., 2008)

Motivation I: Web-Search Ranking

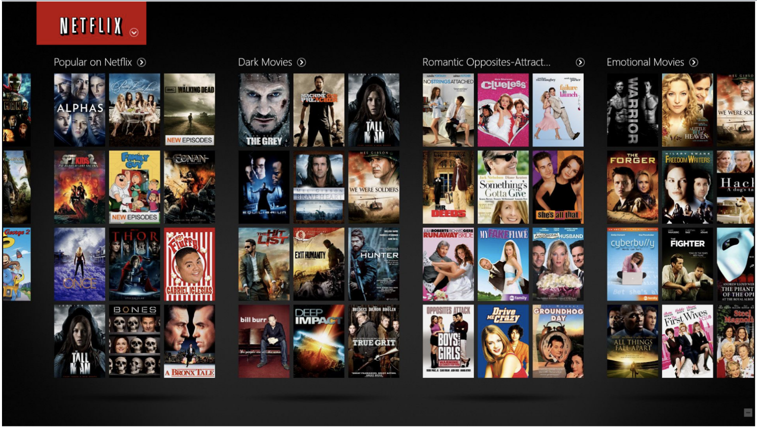


Can extract pairwise comparisons
(Radlinski et al., 2008)

Motivation II: Movie Recommendation



Motivation II: Movie Recommendation



Simultaneously satisfy users and determine best movie

Dueling Bandits

- K arms

Dueling Bandits

- K arms
- time horizon T

Dueling Bandits

- K arms
- time horizon T
- in trial $t \in [T]$:

  select pair (i_t, j_t)

Dueling Bandits

- K arms
- time horizon T
- in trial $t \in [T]$:

  select pair (i_t, j_t)
  observe noisy comparison

Dueling Bandits

- K arms
- time horizon T
- in trial $t \in [T]$:
  select pair (i_t, j_t)
  observe noisy comparison
- noisy comparison:
  $\Pr(i \text{ beats } j) = P_{i,j}$
  comparisons are independent

Dueling Bandits

- K arms
- time horizon T
- in trial $t \in [T]$:

  select pair (i_t, j_t)
  observe noisy comparison
- noisy comparison:

  $\Pr(i \text{ beats } j) = P_{i,j}$
  comparisons are independent
  $P_{i,j} = \frac{1}{2} + \epsilon(i,j)$: measure of distinguishability

Dueling Bandits

- K arms
- time horizon T
- in trial $t \in [T]$:
  select pair (i_t, j_t)
  observe noisy comparison
- noisy comparison:
  $\Pr(i \text{ beats } j) = P_{i,j}$
  comparisons are independent
  $P_{i,j} = \frac{1}{2} + \epsilon(i,j)$: measure of distinguishability
- assume $i^* =$ best arm; $\epsilon(i^*, i) \geq 0$ for all i

Dueling Bandits

- K arms
- time horizon T
- in trial $t \in [T]$:

  select pair (i_t, j_t)
  observe noisy comparison
- noisy comparison:

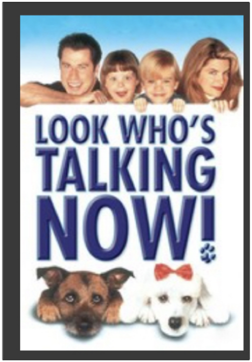
  $\Pr(i \text{ beats } j) = P_{i,j}$
  comparisons are independent
  $P_{i,j} = \frac{1}{2} + \epsilon(i,j)$: measure of distinguishability
- assume $i^* =$ best arm; $\epsilon(i^*, i) \geq 0$ for all i

Goal: perform noisy comparisons that have low regret wrt i^*

Regret: Motivation

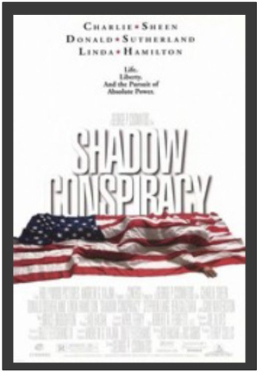
want to maximize user satisfaction

Regret: Motivation

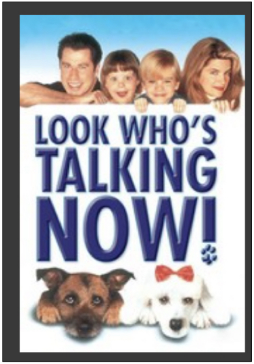


Regret: Motivation

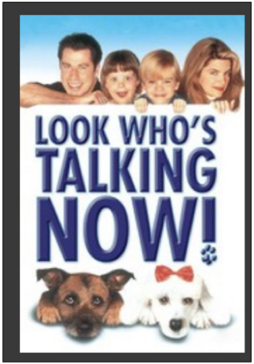
Regret: Motivation



Regret: Motivation



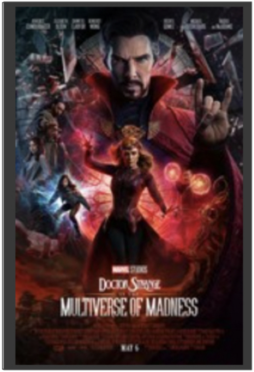
Regret: Motivation



may help in learning; users may be unsatisfied

Regret: Motivation

Regret: Motivation



Regret: Motivation

Regret: Motivation

Regret: Motivation

Regret: Motivation

simultaneously learn and keep users satisfied

Regret

- let i^* = best arm

Regret

- let $i^* =$ best arm
- in trial t:
  - (i_t, j_t) selected

Regret

- let $i^* =$ best arm
- in trial t:

  - (i_t, j_t) selected

  - $r(t) = \epsilon_{i_t} + \epsilon_{j_t}$

Regret

- let $i^* =$ best arm
- in trial t:

  - (i_t, j_t) selected

  - $r(t) = \epsilon_{i_t} + \epsilon_{j_t}$: measures sub-optimality against i^*
  notation: $\epsilon_j = \epsilon(i^*, j)$

Regret

- let $i^* =$ best arm
- in trial t:

  - (i_t, j_t) selected

  - $r(t) = \epsilon_{i_t} + \epsilon_{j_t}$: measures sub-optimality against i^*
  notation: $\epsilon_j = \epsilon(i^*, j)$

  - total regret $R(T) = \sum_t r(t)$

Regret

- let $i^* =$ best arm
- in trial t:

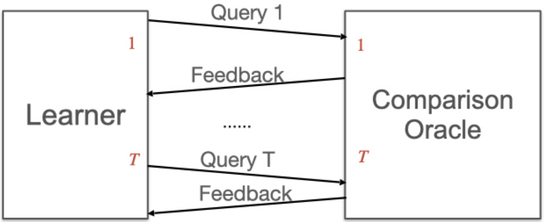
  - (i_t, j_t) selected

  - $r(t) = \epsilon_{i_t} + \epsilon_{j_t}$: measures sub-optimality against i^*
  notation: $\epsilon_j = \epsilon(i^*, j)$

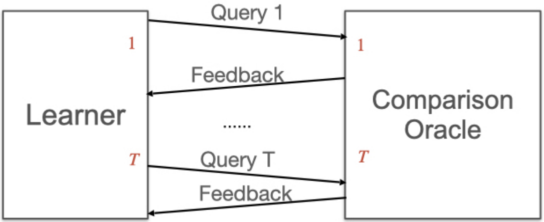
  - total regret $R(T) = \sum_t r(t)$

Perform noisy comparisons with low regret wrt i^*

Full Adaptivity

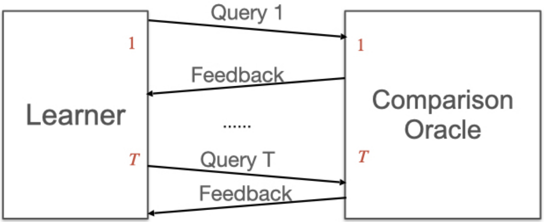


Full Adaptivity



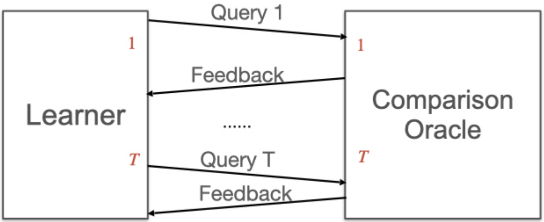
- policy updates one at a time

Full Adaptivity



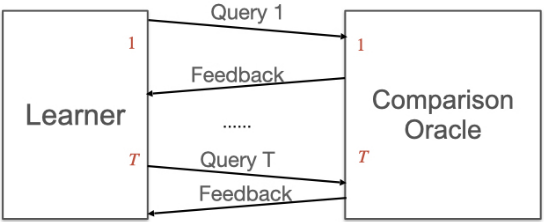
- policy updates one at a time
- can use prior observations to make selection

Full Adaptivity



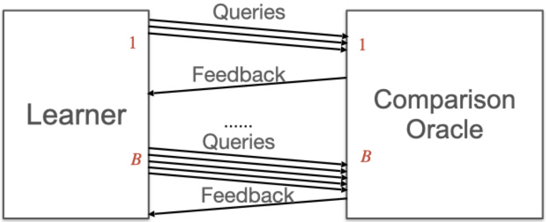
- policy updates one at a time
- can use prior observations to make selection
- may be infeasible in large systems

Full Adaptivity

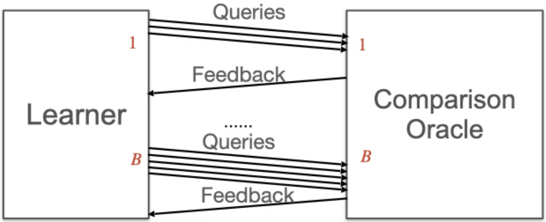


- policy updates one at a time
- can use prior observations to make selection
- may be infeasible in large systems
- requires large computational resources

Limited Adaptivity: Batching

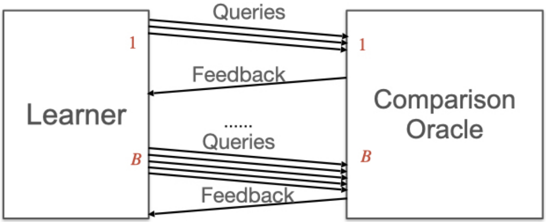


Limited Adaptivity: Batching



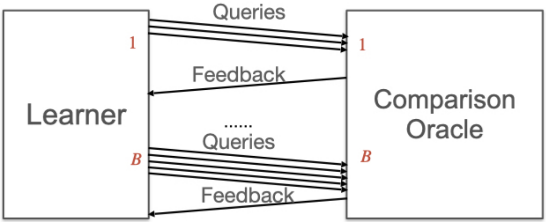
- learner makes multiple comparisons in parallel

Limited Adaptivity: Batching



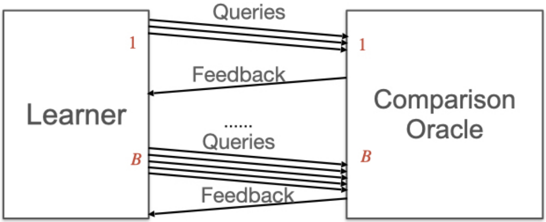
- learner makes multiple comparisons in parallel
- receives all feedback simultaneously

Limited Adaptivity: Batching



- learner makes multiple comparisons in parallel
- receives all feedback simultaneously
- adaptively selects next batch

Limited Adaptivity: Batching



- learner makes multiple comparisons in parallel
- receives all feedback simultaneously
- adaptively selects next batch

Given number of batches B, perform B batches of noisy comparisons with low regret wrt i^*

Main Results: Informal

- Trade-off b/w # batches and regret under two well-studied pairwise comparison models:

  (1) SST + STI

  (2) Condorcet

Main Results: Informal

- Trade-off b/w # batches and regret under two well-studied pairwise comparison models:
  - (1) SST + STI
  - (2) Condorcet
- $O(BT^{1/B}\log(T))$ regret in $O(B)$ rounds
  - $O(\log^2(T))$ regret in $O(\log(T))$ rounds

  Ignoring dependence on K

Main Results: Informal

- Trade-off b/w # batches and regret under two well-studied pairwise comparison models:
  - (1) SST + STI
  - (2) Condorcet
- $O(BT^{1/B}\log(T))$ regret in $O(B)$ rounds
  - $O(\log^2(T))$ regret in $O(\log(T))$ rounds

  Ignoring dependence on K
- Lower bound: $\Omega(T^{1/B})$ in B rounds

Pairwise Comparison Models

- $\epsilon(i,j) = P_{i,j} - 1/2$

Pairwise Comparison Models

- $\epsilon(i,j) = P_{i,j} - 1/2$
- Condorcet: $\exists\ i^*$ such that $\epsilon(i^*, i) \geq 0$ for $i \neq i^*$

Pairwise Comparison Models

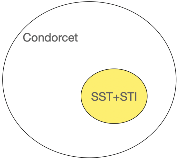
- $\epsilon(i,j) = P_{i,j} - 1/2$
- Condorcet: $\exists\ i^*$ such that $\epsilon(i^*, i) \geq 0$ for $i \neq i^*$
  - there exists a best arm

Pairwise Comparison Models

- $\epsilon(i,j) = P_{i,j} - 1/2$
- Condorcet: $\exists\ i^*$ such that $\epsilon(i^*, i) \geq 0$ for $i \neq i^*$
  - there exists a best arm
- SST + STI: $\exists$ ordering $\succ$ such that for $i \succ j \succ k$:
  - $\epsilon(i,k) \geq \max\{\epsilon(i,j), \epsilon(j,k)\}$ (Strong Stoch. Transitivity)
  - $\epsilon(i,k) \leq \epsilon(i,j) + \epsilon(j,k)$ (Stoch. Triangle Inequality)

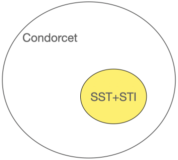
Pairwise Comparison Models

- $\epsilon(i,j) = P_{i,j} - 1/2$
- Condorcet: $\exists\ i^*$ such that $\epsilon(i^*, i) \geq 0$ for $i \neq i^*$
  - there exists a best arm
- SST + STI: $\exists$ ordering $\succ$ such that for $i \succ j \succ k$:
  - $\epsilon(i,k) \geq \max\{\epsilon(i,j), \epsilon(j,k)\}$ (Strong Stoch. Transitivity)
  - $\epsilon(i,k) \leq \epsilon(i,j) + \epsilon(j,k)$ (Stoch. Triangle Inequality)
- Condorcet setting is more general



Pairwise Comparison Models

- $\epsilon(i, j) = P_{i,j} - 1/2$
- Condorcet: $\exists\ i^*$ such that $\epsilon(i^*, i) \geq 0$ for $i \neq i^*$
  - there exists a best arm
- SST + STI: $\exists$ ordering $\succ$ such that for $i \succ j \succ k$:
  - $\epsilon(i, k) \geq \max\{\epsilon(i, j), \epsilon(j, k)\}$ (Strong Stoch. Transitivity)
  - $\epsilon(i, k) \leq \epsilon(i, j) + \epsilon(j, k)$ (Stoch. Triangle Inequality)
- Condorcet setting is more general



- Extensive amount of work on sequential algs: Yue et al. (2012), Yue and Joachims (2011), Zoghi et al. (2014), Komiyama et al. (2015)

Main Results

Theorem 1

There is an algorithm for batched dueling bandits that uses at most B rounds, and if the instance admits a Condorcet winner, the expected regret is bounded by

$$\mathbb{E}[R(T)] \leq 3K T^{1/B} \log\left(6TK^2B\right) \sum_{j:\epsilon_j>0} \frac{1}{\epsilon_j}.$$

Main Results

Theorem 1

There is an algorithm for batched dueling bandits that uses at most B rounds, and if the instance admits a Condorcet winner, the expected regret is bounded by

$$\mathbb{E}[R(T)] \leq 3K T^{1/B} \log\left(6TK^2B\right) \sum_{j:\epsilon_j>0} \frac{1}{\epsilon_j}.$$

- simplified: $O\left(KT^{1/B}\log(T)\sum_j \frac{1}{\epsilon_j}\right)$
- worst-case: $O\left(\frac{K^2T^{1/B}\log(T)}{\epsilon_{\min}}\right)$; $\epsilon_{\min} = \min_{j:\epsilon_j>0}\epsilon_j$
- lower bound result: $\Omega\left(\frac{KT^{1/B}}{B^2\epsilon_{\min}}\right)$

Main Results

Theorem 2

There is an algorithm for batched dueling bandits that uses at most $B+1$ batches, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by

$$\mathbb{E}[R(T)] = \sum_{j:\epsilon_j>0} O\left(\frac{\sqrt{K}\,T^{1/B}\log(T)}{\epsilon_j}\right).$$

Main Results

Theorem 2

There is an algorithm for batched dueling bandits that uses at most $B+1$ batches, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by

$$\mathbb{E}[R(T)] = \sum_{j:\epsilon_j>0} O\left(\frac{\sqrt{K}\,T^{1/B}\log(T)}{\epsilon_j}\right).$$

- worst-case: $O\left(\frac{K^{1.5}T^{1/B}\log(T)}{\epsilon_{\min}}\right)$

Main Results

Theorem 3

There is an algorithm for batched dueling bandits that uses at most $2B+1$ batches, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by

$$\mathbb{E}[R(T)] = O\left(\frac{K B T^{1/B} \log(T)}{\epsilon_{\min}}\right).$$

Main Results

Theorem 3

There is an algorithm for batched dueling bandits that uses at most $2B+1$ batches, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by

$$\mathbb{E}[R(T)] = O\left(\frac{KBT^{1/B}\log(T)}{\epsilon_{\text{min}}}\right).$$

- better dependence on K

Main Results

Theorem 3

There is an algorithm for batched dueling bandits that uses at most $2B + 1$ batches, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by

$$\mathbb{E}[R(T)] = O\left(\frac{K B T^{1/B} \log(T)}{\epsilon_{\min}}\right).$$

- better dependence on K ; additional dependence on B

Comparison to Sequential Algs

Notation: $\epsilon_j = \epsilon(i^*, j)$, $\epsilon_{\min} = \min_{j:\epsilon_j>0} \epsilon_j$

Setting	Fully Adaptive (prior work)	**Our Algorithms** Regret	Rounds
Condorcet	$O\left(K\frac{\log T}{\epsilon_{\min}}\right) + O\left(\frac{K^2}{\epsilon_{\min}}\right)$	$O\left(\frac{K^2 T^{1/B}\log(T)}{\epsilon_{\min}}\right)$	B
SST + STI	$O\left(\frac{K\log(T)}{\epsilon_{\min}}\right)$	$O\left(\frac{KBT^{1/B}\log(T)}{\epsilon_{\min}}\right)$	$2B+1$

Comparison to Sequential Algs

Notation: $\epsilon_j = \epsilon(i^*, j)$, $\epsilon_{\min} = \min_{j:\epsilon_j > 0} \epsilon_j$

Setting	Fully Adaptive (prior work)	**Our Algorithms**	
		Regret	Rounds
Condorcet	$O\left(K \frac{\log T}{\epsilon_{\min}}\right) + O\left(\frac{K^2}{\epsilon_{\min}}\right)$	$O\left(\frac{K^2 \log(T)}{\epsilon_{\min}}\right)$	$\log(T)$
SST + STI	$O\left(\frac{K \log(T)}{\epsilon_{\min}}\right)$	$O\left(\frac{K \log^2(T)}{\epsilon_{\min}}\right)$	$2\log(T) + 1$

Intuition

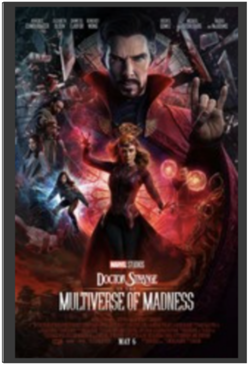

007
GOLDENEYE


CHARLIE SHEEN
DONALD SUTHERLAND
LINDA HAMILTON
Life.
Liberty.
And the Pursuit of
Absolute Power.
SHADOW
CONSPIRACY

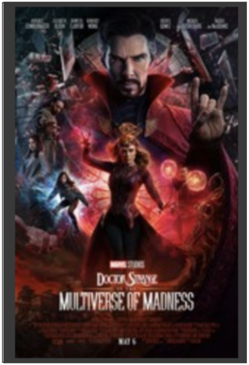
Intuition

few comparisons suffice to decide better option

Intuition



Intuition



may require many comparisons to decide better option

Algorithm

Existence of Condorcet winner; i.e. best arm

Algorithm

Existence of Condorcet winner; i.e. best arm

In batch $r \in [B]$:

- compare all surviving pairs $c_r = T^{r/B}$ times

Algorithm

Existence of Condorcet winner; i.e. best arm

In batch $r \in [B]$:

- compare all surviving pairs $c_r = T^{r/B}$ times
- so we don't waste comparisons on sub-optimal arms

Algorithm

Existence of Condorcet winner; i.e. best arm

In batch $r \in [B]$:

- compare all surviving pairs $c_r = T^{r/B}$ times
- so we don't waste comparisons on sub-optimal arms
- eliminate sub-optimal arms before moving to next batch

Algorithm

Existence of Condorcet winner; i.e. best arm

In batch $r \in [B]$:

- compare all surviving pairs $c_r = T^{r/B}$ times
- so we don't waste comparisons on sub-optimal arms
- eliminate sub-optimal arms before moving to next batch

Elimination criteria:

- set precision $\gamma_r = \sqrt{\log\left(\frac{1}{\delta}\right)/2c_r}$; $\delta \approx T^{-4}$

Algorithm

Existence of Condorcet winner; i.e. best arm

In batch $r \in [B]$:

- compare all surviving pairs $c_r = T^{r/B}$ times
- so we don't waste comparisons on sub-optimal arms
- eliminate sub-optimal arms before moving to next batch

Elimination criteria:

- set precision $\gamma_r = \sqrt{\log\left(\frac{1}{\delta}\right)/2c_r}$; $\delta \approx T^{-4}$
- delete j if $\widehat{P}_{i,j} > 1/2 + \gamma_r$

$$\widehat{P}_{i,j} = \frac{\#\text{ times } i \text{ wins over } j}{\#\text{ times } i \text{ and } j \text{ compared in round } r}$$

Regret Analysis I

- Correct estimate if $|P_{i,j} - \widehat{P}_{i,j}| \leq \gamma_r$: denoted $P_{i,j} \approx_r \widehat{P}_{i,j}$

Regret Analysis I

- Correct estimate if $|P_{i,j} - \widehat{P}_{i,j}| \leq \gamma_r$: denoted $P_{i,j} \approx_r \widehat{P}_{i,j}$
- By Hoeffding: every estimate is correct in every batch with high probability

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \leq \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \leq \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction
  - can use i^* as an anchor to eliminate others

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \leq \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction
  - can use i^* as an anchor to eliminate others
- Suppose j not deleted in batch r: $P_{i^*,j} \leq 1/2 + 2\gamma_r$

$$\epsilon_j \leq 2\gamma_r = 2\sqrt{\frac{\log(1/\delta)}{2c_r}} \Rightarrow c_r \leq \frac{2\log(1/\delta)}{\epsilon_j^2}$$

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \le \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction
  - can use i^* as an anchor to eliminate others
- Suppose j not deleted in batch r: $P_{i^*,j} \le 1/2 + 2\gamma_r$

$$\epsilon_j \le 2\gamma_r = 2\sqrt{\frac{\log(1/\delta)}{2c_r}} \Rightarrow c_r \le \frac{2\log(1/\delta)}{\epsilon_j^2}$$

- Let r be the last such batch; then
  - # comparisons of j and $i^* \le \sum_{\tau=1}^{r+1} c_\tau \le 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2}$

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \leq \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction
  - can use i^* as an anchor to eliminate others
- Suppose j not deleted in batch r: $P_{i^*,j} \leq 1/2 + 2\gamma_r$

$$\epsilon_j \leq 2\gamma_r = 2\sqrt{\frac{\log(1/\delta)}{2c_r}} \Rightarrow c_r \leq \frac{2\log(1/\delta)}{\epsilon_j^2}$$

- Let r be the last such batch; then
  - # comparisons of j and $i^* \leq \sum_{\tau=1}^{r+1} c_\tau \leq 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2}$
  - total comparisons for $j \leq K \cdot 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2} = T_j$

Regret Analysis II

Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \leq \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction
  - can use i^* as an anchor to eliminate others
- Suppose j not deleted in batch r: $P_{i^*,j} \leq 1/2 + 2\gamma_r$

$$\epsilon_j \leq 2\gamma_r = 2\sqrt{\frac{\log(1/\delta)}{2c_r}} \Rightarrow c_r \leq \frac{2\log(1/\delta)}{\epsilon_j^2}$$

- Let r be the last such batch; then
  - # comparisons of j and $i^* \leq \sum_{\tau=1}^{r+1} c_\tau \leq 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2}$
  - total comparisons for $j \leq K \cdot 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2} = T_j$
  - total regret contribution: $\epsilon_j \cdot T_j$

Regret Analysis II

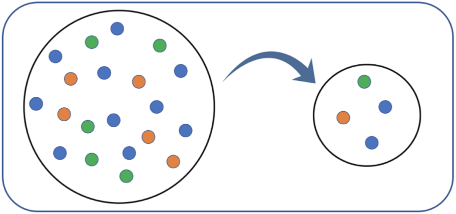
Assumptions: Condorcet winner + $P_{i,j} \approx_r \widehat{P}_{i,j}$
Notation: $\epsilon_j = \epsilon(i^*, j)$

- Recall: if $\widehat{P}_{i,j} > 1/2 + \gamma_r$, delete j
- i^* never deleted: else $P_{i,i^*} \leq \widehat{P}_{i,j} - \gamma_r < 1/2$, contradiction
  - can use i^* as an anchor to eliminate others
- Suppose j not deleted in batch r: $P_{i^*,j} \leq 1/2 + 2\gamma_r$

$$\epsilon_j \leq 2\gamma_r = 2\sqrt{\frac{\log(1/\delta)}{2c_r}} \Rightarrow c_r \leq \frac{2\log(1/\delta)}{\epsilon_j^2}$$

- Let r be the last such batch; then
  - # comparisons of j and $i^* \leq \sum_{\tau=1}^{r+1} c_\tau \leq 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2}$
  - total comparisons for $j \leq K \cdot 2T^{1/B} \cdot \frac{2\log(1/\delta)}{\epsilon_j^2} = T_j$
  - total regret contribution: $\epsilon_j \cdot T_j$
- Summing over all j gives the Condorcet guarantee!

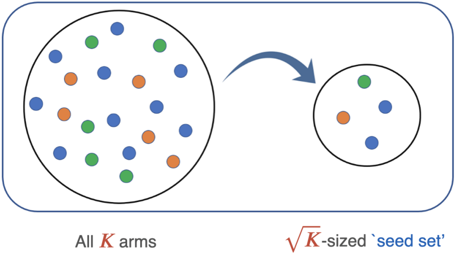
Algorithm+: Seeded Comparisons



All K arms

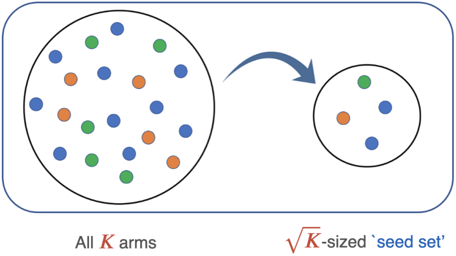
$\sqrt{K}$-sized `seed set'

Algorithm+: Seeded Comparisons



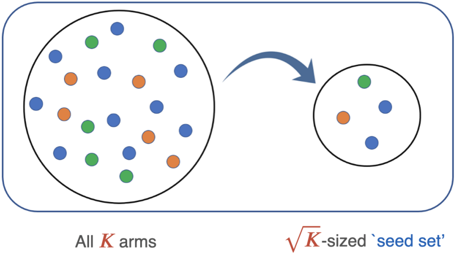
- Compare each seed with every active arm as before

Algorithm+: Seeded Comparisons



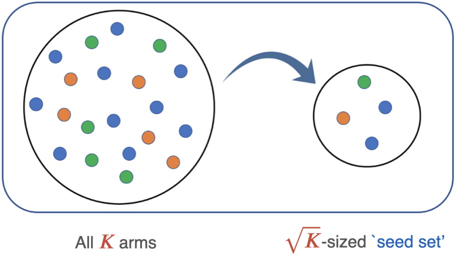
- Compare each seed with every active arm as before
- Eliminate sub-optimal arms

Algorithm+: Seeded Comparisons



- Compare each seed with every active arm as before
- Eliminate sub-optimal arms
- Switch to all pairs policy if $< \sqrt{K}$ arms remain (this is ok!)

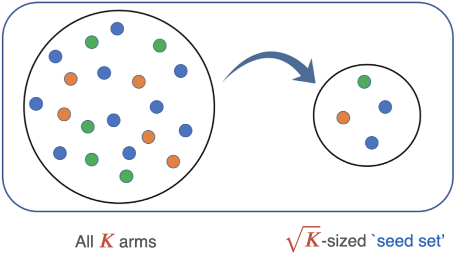
Algorithm+: Seeded Comparisons



- Compare each seed with every active arm as before
- Eliminate sub-optimal arms
- Switch to all pairs policy if $< \sqrt{K}$ arms remain (this is ok!)

> Technical insight: randomly chosen seed set contains a "good" arm that acts as anchor

Algorithm+: Seeded Comparisons



- Compare each seed with every active arm as before
- Eliminate sub-optimal arms
- Switch to all pairs policy if $< \sqrt{K}$ arms remain (this is ok!)

Technical insight: randomly chosen seed set contains a "good" arm that acts as anchor $\rightarrow$ gives $\widetilde{O}(K^{1.5})$ dependence!

Algorithm++: Additional Adaptivity

- Need more ideas to achieve $\widetilde{O}(K)$ dependence

Algorithm++: Additional Adaptivity

- Need more ideas to achieve $\widetilde{O}(K)$ dependence
- Compare seeds amongst themselves; choose empirical best

Algorithm++: Additional Adaptivity

- Need more ideas to achieve $\widetilde{O}(K)$ dependence
- Compare seeds amongst themselves; choose empirical best
  - here we need additional adaptivity
- Use this seed against all active arms

Algorithm++: Additional Adaptivity

- Need more ideas to achieve $\widetilde{O}(K)$ dependence
- Compare seeds amongst themselves; choose empirical best
  - here we need additional adaptivity
- Use this seed against all active arms

Technical insight: best seed serves as good proxy for anchor

Algorithm++: Additional Adaptivity

- Need more ideas to achieve $\widetilde{O}(K)$ dependence
- Compare seeds amongst themselves; choose empirical best
  - here we need additional adaptivity
- Use this seed against all active arms

Technical insight: best seed serves as good proxy for anchor and at most B different best arms

Algorithm++: Additional Adaptivity

- Need more ideas to achieve $\widetilde{O}(K)$ dependence
- Compare seeds amongst themselves; choose empirical best
  - here we need additional adaptivity
- Use this seed against all active arms

Technical insight: best seed serves as good proxy for anchor and at most B different best arms $\rightarrow$
gives $\widetilde{O}(KB)$ dependence!

Computations: Set-up

Datasets used

Computations: Set-up

Datasets used

- ArXiv: Six rankers
- Sushi
- Synthetic data based on BTL model
- Synthetic data based on Hard Instances

Computations: Set-up

Datasets used

- ArXiv: Six rankers
- Sushi
- Synthetic data based on BTL model
- Synthetic data based on Hard Instances

Benchmarks

- RMED (Komiyama et al., 2015)
- RUCB (Zoghi et al., 2014)
- BTM (Yue and Joachims, 2011)

Computations: Regret using log(T) batches

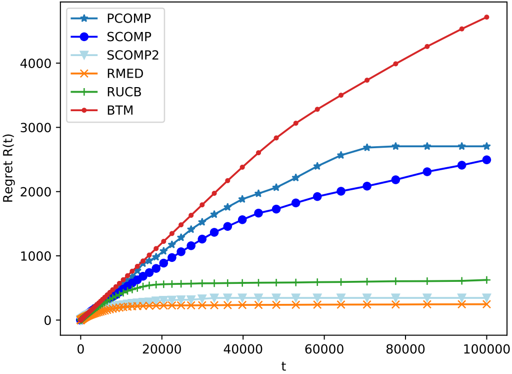


Figure: (a) Six rankers

Computations: Trade-off b/w regret and #batches

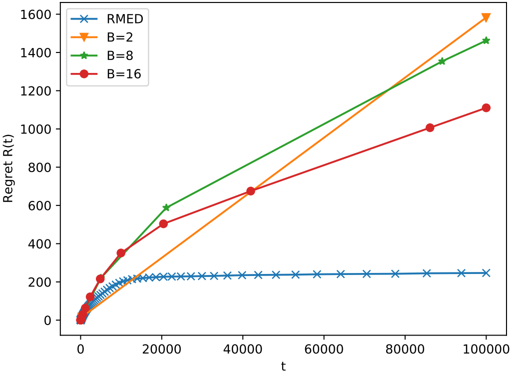


Figure: (a) Six rankers

Conclusion

- Introduce the batched dueling bandit problem

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet
- Also give matching lower bound against # batches
  - $\Omega(T^{1/B})$ for B batches

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet
- Also give matching lower bound against # batches
  - $\Omega(T^{1/B})$ for B batches
- Experiments corroborate our theoretical results

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet
- Also give matching lower bound against # batches
  - $\Omega(T^{1/B})$ for B batches
- Experiments corroborate our theoretical results
- Open Question I: How many batches are needed to exactly match sequential results in (i) SST+STI, and (ii) Condorcet

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet
- Also give matching lower bound against # batches
  - $\Omega(T^{1/B})$ for B batches
- Experiments corroborate our theoretical results
- Open Question I: How many batches are needed to exactly match sequential results in (i) SST+STI, and (ii) Condorcet
- Open Question II: Can we obtain similar results for more general notions of winner; for e.g., von Nuemann winner, Copeland winner, etc.

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet
- Also give matching lower bound against # batches
  - $\Omega(T^{1/B})$ for B batches
- Experiments corroborate our theoretical results
- Open Question I: How many batches are needed to exactly match sequential results in (i) SST+STI, and (ii) Condorcet
- Open Question II: Can we obtain similar results for more general notions of winner; for e.g., von Nuemann winner, Copeland winner, etc.
- Full paper: https://tinyurl.com/batcheddb

Conclusion

- Introduce the batched dueling bandit problem
- Give algorithms that obtain a trade-off b/w #batches and regret for two pairwise comparison models:
  - SST + STI
  - Condorcet
- Also give matching lower bound against # batches
  - $\Omega(T^{1/B})$ for B batches
- Experiments corroborate our theoretical results
- Open Question I: How many batches are needed to exactly match sequential results in (i) SST+STI, and (ii) Condorcet
- Open Question II: Can we obtain similar results for more general notions of winner; for e.g., von Nuemann winner, Copeland winner, etc.
- Full paper: https://tinyurl.com/batcheddb

THANK YOU!