Decentralized Cooperative Stochastic Multiarmed Bandits

David Martínez-Rubio [1] Varun Kanade [1] Patrick Rebeschini [2]

[1]Department of Computer Science - University of Oxford

[2]Department of Statistics - University of Oxford

March 8 2019

Stochastic Multiarmed Bandits

- An agent chooses iteratively an arm among $K \in \mathbb{N}$ possible arms and receives a reward. This happens for T time steps.
- The goal is to get cumulative reward as close as possible to the reward she could have obtained with the best fixed action, in hindsight.

Stochastic Multiarmed Bandits

- An agent chooses iteratively an arm among $K \in \mathbb{N}$ possible arms and receives a reward. This happens for T time steps.
- The goal is to get cumulative reward as close as possible to the reward she could have obtained with the best fixed action, in hindsight.
- In the stochastic version, rewards are sampled from a distribution which is arm-dependent. In the adversarial one, rewards are set by an adversary.
- Optimal algorithms have been developed for both the stochastic and the adversarial versions, e.g. the Upper Confidence Bound algorithm (UCB) for the stochastic version.

Stochastic Multiarmed Bandits

- An agent chooses iteratively an arm among $K \in \mathbb{N}$ possible arms and receives a reward. This happens for T time steps.
- The goal is to get cumulative reward as close as possible to the reward she could have obtained with the best fixed action, in hindsight.
- In the stochastic version, rewards are sampled from a distribution which is arm-dependent. In the adversarial one, rewards are set by an adversary.
- Optimal algorithms have been developed for both the stochastic and the adversarial versions, e.g. the Upper Confidence Bound algorithm (UCB) for the stochastic version.
- The goal can be rephrased in terms of minimizing regret:

$$R(T) = \sup_a \sum_{t=1}^{T} r_t(a) - \sum_{t=1}^{T} r_t(a_t)$$

Stochastic Multiarmed Bandits

- An agent chooses iteratively an arm among $K \in \mathbb{N}$ possible arms and receives a reward. This happens for T time steps.
- The goal is to get cumulative reward as close as possible to the reward she could have obtained with the best fixed action, in hindsight.
- In the stochastic version, rewards are sampled from a distribution which is arm-dependent. In the adversarial one, rewards are set by an adversary.
- Optimal algorithms have been developed for both the stochastic and the adversarial versions, e.g. the Upper Confidence Bound algorithm (UCB) for the stochastic version.
- The goal can be rephrased in terms of minimizing regret:

$$R(T) = \sup_a \sum_{t=1}^{T} r_t(a) - \sum_{t=1}^{T} r_t(a_t)$$

In the stochastic case, in expectation, if Δ_k is the difference between the best mean and the mean for arm k:

$$R(T) = \mathbf{E}\left[\sum_{t=1}^{T} \Delta_{I_t}\right].$$

UCB

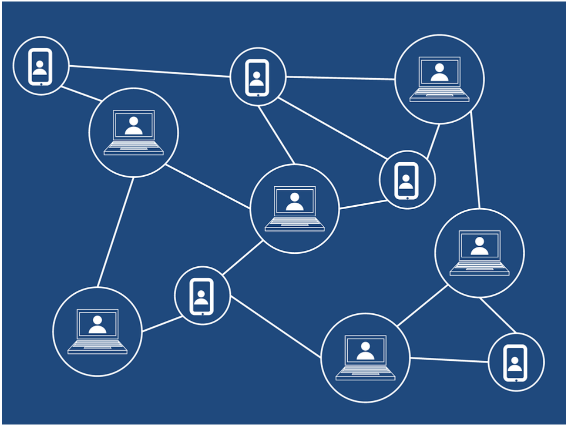
The UCB algorithm is an optimal algorithm for the multiarmed stochastic bandits problem. Fix a time step t and define:

- μ_t^k:= empirical mean observed for arm k.
- n_t^k:= number of times arm k was pulled.
- η:= exploration parameter > 1.
- I_t:= action played at time t.
- $UCB_k := \mu_t^k + \sqrt{\frac{2\eta\sigma^2 \ln t}{n_t^k}}$.

Then UCB picks the arm that maximizes UCB_k:

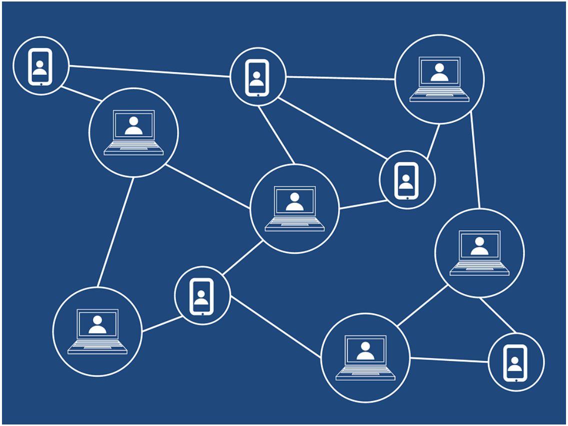
$$I_t = \mathrm{argmax}_{k \in [K]} UCB_k.$$

Decentralized Computation



- The decentralization may be an **inherent restriction of the problem**, as is the case in packet routing or sensor networks, massive ad placement. It could be a choice made to **improve the total running time**.

Decentralized Computation



- The decentralization may be an **inherent restriction of the problem**, as is the case in packet routing or sensor networks, massive ad placement. It could be a choice made to **improve the total running time**.
- Focus should be graph-dependent algorithms, so we can cover the case in which we design the network and the case in which the network is a restriction.

Question

How do we solve decentralized bandit problems?

Question

How do we solve decentralized bandit problems?

The aim is to:

Question

How do we solve decentralized bandit problems?

The aim is to:

- Get regret as close as possible to the best regret attainable with a centralized algorithm.

Question

How do we solve decentralized bandit problems?

The aim is to:

- Get regret as close as possible to the best regret attainable with a centralized algorithm.
- Consider the batched bandit problem as a baseline.

Question

How do we solve decentralized bandit problems?

The aim is to:

- Get regret as close as possible to the best regret attainable with a centralized algorithm.
- Consider the batched bandit problem as a baseline.
- Assume as little non-local information as possible.

Question

How do we solve decentralized bandit problems?

The aim is to:

- Get regret as close as possible to the best regret attainable with a centralized algorithm.
- Consider the batched bandit problem as a baseline.
- Assume as little non-local information as possible.

Adversarial case:

- **No communication case.** Regret incurred if separate optimal algorithms are run: $N\sqrt{KT}$.
- A lower bound: $N\sqrt{T}$.
- There is an algorithm achieving regret $N(\sqrt{K^{1/2}T\log K} + \sqrt{K}\log T)$ (**?**).

Gossip communication

- Nodes of a network send messages only to their neighbors.
- Many decentralized problems can be reduced to approximate averaging (averaging gradients for optimization, averaging samples from a distribution).

Gossip communication

- Nodes of a network send messages only to their neighbors.
- Many decentralized problems can be reduced to approximate averaging (averaging gradients for optimization, averaging samples from a distribution).
- Compute iteratively a weighted average of your value and the ones sent by your neighbors. If $x \in \mathbb{R}^N$ contains the values of the nodes, this operation is Px, for a *communication matrix* P. We just need P^s to converge to $\mathbb{1}\mathbb{1}^\top/N$.

Gossip communication

- Nodes of a network send messages only to their neighbors.
- Many decentralized problems can be reduced to approximate averaging (averaging gradients for optimization, averaging samples from a distribution).
- Compute iteratively a weighted average of your value and the ones sent by your neighbors. If $x \in \mathbb{R}^N$ contains the values of the nodes, this operation is Px, for a *communication matrix* P. We just need P^s to converge to $\mathbb{1}\mathbb{1}^\top/N$.
- Properties of P:
  - Supported on graph.
  - $P\mathbb{1} = \mathbb{1}; \qquad \mathbb{1}^\top P = \mathbb{1}^\top; \qquad |\lambda_2| < 1$

    $\iff P^s \xrightarrow[s\to\infty]{} \mathbb{1}\mathbb{1}^\top/N$ **(?)**
  - For acceleration eigenvalues must be real.

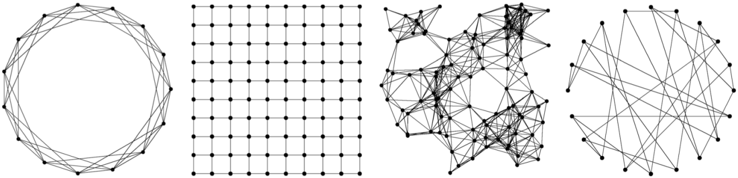
Gossip communication

Graphs with smaller spectral gap will benefit less from decentralization.

Example: time speedup of the dual averaging optimization algorithm with respect to non distributed computation is (**?**):

Graph	Cycle	Sq. Grid	Random Geometric Graph	Expander
Speedup	$\tilde{O}(1)$	$\tilde{O}(\sqrt{N})$	$\tilde{O}(\sqrt{N})$ *	$\tilde{O}(N)$

*Speedup with high probability for a random geometric graph on $[0,1]^2$ with connectivity radius $\Omega(\sqrt{\log^{1+\varepsilon} n/n})$ (for any $\varepsilon > 0$).



Model and Problem

- N agents in a graph with communication matrix P.
- Same K-armed bandit problem at each node.
- T time steps.
- Reward distributions are subgaussian with proxy σ^2 with means $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$.
- Regret is defined as

$$R(T) = TN\mu_1 - \mathbf{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N}\mu_{I_{t,i}}\right] = \sum_{k=1}^{K}\Delta_k \mathbf{E}\left[n_T^k\right],$$

  The expectation is taken with respect to the algorithm and the rewards.
- poly(K) values are allowed to be communicated at each time step.
- We want the algorithm to be decentralized: only N and an upper bound on the spectral gap of P is assumed to be known to the agents, besides their corresponding row of P.

Decentralized UCB?

$$UCB_k := \mu_t^k + \sqrt{\frac{2\eta\sigma^2 \ln t}{n_t^k}}.$$

UCB needs to compute μ_t^k and n_t^k. In a decentralized setting could be approximated with gossip communication. It is an averaging problem.

Difficulties of the adaptation to the decentralized setting:

1. New rewards are obtained at each time step, the average is not only of one value per node.
2. We can only compute approximations of the averages.
3. It turns out that for getting good approximations we need to delay the rewards given to the UCB algorithm.

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = Px_t + y_t$ with $x_1 = 0$

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = Px_t + y_t$ with $x_1 = 0$
Problem: Poor approximation due to last-added values not sufficiently mixed

$$x_{t+1} = \underbrace{y_t + Py_{t-1} + \cdots}_{\text{NOT well-mixed}} + \underbrace{\cdots + P^{t-2}y_2 + P^{t-1}y_1}_{\text{well-mixed}}$$

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ $\quad$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = Px_t + y_t$ $\qquad$ with $x_1 = 0$
Problem: Poor approximation due to last-added values not sufficiently mixed

$$x_{t+1} = \underbrace{y_t + Py_{t-1} + \cdots}_{\text{NOT well-mixed}} + \underbrace{\cdots + P^{t-2}y_2 + P^{t-1}y_1}_{\text{well-mixed}}$$

IDEA: Take decisions only based on well-mixed (delayed) information

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ $\quad$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = Px_t + y_t$ $\qquad$ with $x_1 = 0$
Problem: Poor approximation due to last-added values not sufficiently mixed

$$x_{t+1} = \underbrace{y_t + Py_{t-1} + \cdots}_{\text{NOT well-mixed}} + \underbrace{\cdots + P^{t-2}y_2 + P^{t-1}y_1}_{\text{well-mixed}}$$

IDEA: Take decisions only based on well-mixed (delayed) information

- Use mixing to prescribe delay

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ $\quad$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = Px_t + y_t$ $\qquad$ with $x_1 = 0$
Problem: Poor approximation due to last-added values not sufficiently mixed

$$x_{t+1} = \underbrace{y_t + Py_{t-1} + \cdots}_{\text{NOT well-mixed}} + \underbrace{\cdots + P^{t-2}y_2 + P^{t-1}y_1}_{\text{well-mixed}}$$

IDEA: Take decisions only based on well-mixed (delayed) information

- Use mixing to prescribe delay
- Delay does not increase regret much: τ-delay increases regret by $\tau \sum_k \Delta_k$

Running Consensus

Running Consensus (?). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = Px_t + y_t$ with $x_1 = 0$
Problem: Poor approximation due to last-added values not sufficiently mixed

$$x_{t+1} = \underbrace{y_t + Py_{t-1} + \cdots}_{\text{NOT well-mixed}} + \underbrace{\cdots + P^{t-2}y_2 + P^{t-1}y_1}_{\text{well-mixed}}$$

IDEA: Take decisions only based on well-mixed (delayed) information

- Use mixing to prescribe delay
- Delay does not increase regret much: τ-delay increases regret by $\tau \sum_k \Delta_k$

Acceleration: Compute a polynomial of P that minimizes all eigenvalues but $\lambda_1 = 1$.

Running Consensus

Running Consensus (**?**). Values sequentially added to the network
$y_1, \ldots, y_t \in \mathbb{R}^N$ $\quad$ ($2K$ of these (μ_t^k and n_t^k for $k = 1, \ldots, K$))

Running consensus approximation: $x_{t+1} = P x_t + y_t$ $\qquad$ with $x_1 = 0$
Problem: Poor approximation due to last-added values not sufficiently mixed

$$x_{t+1} = \underbrace{y_t + P y_{t-1} + \cdots}_{\text{NOT well-mixed}} + \underbrace{\cdots + P^{t-2} y_2 + P^{t-1} y_1}_{\text{well-mixed}}$$

IDEA: Take decisions only based on well-mixed (delayed) information

- Use mixing to prescribe delay
- Delay does not increase regret much: τ-delay increases regret by $\tau \sum_k \Delta_k$

Acceleration: Compute a polynomial of P that minimizes all eigenvalues but $\lambda_1 = 1$.

- Use rescaled Chebyshev polynomials.

Delayed average: a small-variance estimator

- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} \right\rceil$ we have $\left\| P^C - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$, (**?**).
- By rescaling Chebyshev's polynomials we can find the polynomial $q_s(x)$ of degree s with minimum supremum in the interval $[-|\lambda_2|, |\lambda_2|]$ that satisfies $q_s(1) = 1$, similarly as in (**?**). So $q_s(P)y_t$ can be computed with s gossip iterations.
- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\sqrt{\ln(1/|\lambda_2|)}} \right\rceil$ we have $\left\| q_C(P) - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$.

Delayed average: a small-variance estimator

- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} \right\rceil$ we have $\left\| P^C - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$, (**?**).
- By rescaling Chebyshev's polynomials we can find the polynomial $q_s(x)$ of degree s with minimum supremum in the interval $[-|\lambda_2|, |\lambda_2|]$ that satisfies $q_s(1) = 1$, similarly as in (**?**). So $q_s(P)y_t$ can be computed with s gossip iterations.
- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\sqrt{\ln(1/|\lambda_2|)}} \right\rceil$ we have $\left\| q_C(P) - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$.
- **Run the algorithm in stages of size C.**

Delayed average: a small-variance estimator

- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} \right\rceil$ we have $\left\| P^C - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$, (**?**).
- By rescaling Chebyshev's polynomials we can find the polynomial $q_s(x)$ of degree s with minimum supremum in the interval $[-|\lambda_2|, |\lambda_2|]$ that satisfies $q_s(1) = 1$, similarly as in (**?**). So $q_s(P)y_t$ can be computed with s gossip iterations.
- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\sqrt{\ln(1/|\lambda_2|)}} \right\rceil$ we have $\left\| q_C(P) - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$.
- **Run the algorithm in stages of size C.**
    - **Use well mixed information to decide the pull.**

Delayed average: a small-variance estimator

- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} \right\rceil$ we have $\left\| P^C - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$, (**?**).
- By rescaling Chebyshev's polynomials we can find the polynomial $q_s(x)$ of degree s with minimum supremum in the interval $[-|\lambda_2|, |\lambda_2|]$ that satisfies $q_s(1) = 1$, similarly as in (**?**). So $q_s(P)y_t$ can be computed with s gossip iterations.
- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\sqrt{\ln(1/|\lambda_2|)}} \right\rceil$ we have $\left\| q_C(P) - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$.
- **Run the algorithm in stages of size C.**
  - **Use well mixed information to decide the pull.**
  - **Rewards obtained in the previous stage mix using the running consensus protocol.**

Delayed average: a small-variance estimator

- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} \right\rceil$ we have $\left\| P^C - \mathbb{1}^T\mathbb{1}/N \right\|_2 \leq \varepsilon/N$, (**?**).
- By rescaling Chebyshev's polynomials we can find the polynomial $q_s(x)$ of degree s with minimum supremum in the interval $[-|\lambda_2|, |\lambda_2|]$ that satisfies $q_s(1) = 1$, similarly as in (**?**). So $q_s(P)y_t$ can be computed with s gossip iterations.
- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\sqrt{\ln(1/|\lambda_2|)}} \right\rceil$ we have $\left\| q_C(P) - \mathbb{1}^T\mathbb{1}/N \right\|_2 \leq \varepsilon/N$.
- **Run the algorithm in stages of size C.**
  - **Use well mixed information to decide the pull.**
  - **Rewards obtained in the previous stage mix using the running consensus protocol.**
  - **Keep rewards of current stage in a third variable for mixing them in the next one.**

Delayed average: a small-variance estimator

- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} \right\rceil$ we have $\left\| P^C - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$, (?).
- By rescaling Chebyshev's polynomials we can find the polynomial $q_s(x)$ of degree s with minimum supremum in the interval $[-|\lambda_2|, |\lambda_2|]$ that satisfies $q_s(1) = 1$, similarly as in (?). So $q_s(P)y_t$ can be computed with s gossip iterations.
- If $C = \left\lceil \frac{\ln(N/\varepsilon)}{\sqrt{\ln(1/|\lambda_2|)}} \right\rceil$ we have $\left\| q_C(P) - \mathbb{1}^T \mathbb{1}/N \right\|_2 \leq \varepsilon/N$.

- **Run the algorithm in stages of size C.**
  - **Use well mixed information to decide the pull.**
  - **Rewards obtained in the previous stage mix using the running consensus protocol.**
  - **Keep rewards of current stage in a third variable for mixing them in the next one.**
  - **At the end of the stage add the recently well mixed rewards to the rest of well mixed rewards.**

Pseudocode

Algorithm 1 Decentralized Delayed UCB at node i.

$\zeta \leftarrow$ Sample from each arm.
mixed$_i \leftarrow (\zeta/N, \mathbb{1}/N)$, mixing$_i \leftarrow (\zeta, \mathbb{1})$, new$_i \leftarrow \mathbf{0}$.
$t \leftarrow K, s \leftarrow K$
while $t \leq T$ **do**
  Obtain $\widehat{\mu}^k_{t,i}, \widehat{n}^k_{t,i}, k = 1, \ldots, K$ from mixed$_i$.
  $k^* \leftarrow \arg\max_{k \in \{1,\ldots,K\}} \left\{ \widehat{\mu}^k_{t,i} + \sqrt{\frac{2\eta\sigma^2 \ln s}{\widehat{n}^k_{t,i}}} \right\}$
  for C iterations **do**
    Perform one mixing step to mixing$_i$.
    Play arm k^*, update new$_i$.
    $t \leftarrow t + 1$
  end for
  $s \leftarrow (t - C)N$
  mixed$_i \leftarrow$ mixed$_i$+mixing$_i$.
  mixing$_i \leftarrow$ new$_i$.
  new$_i \leftarrow \mathbf{0}$.
end while

Regret

[M.R., Kanade, Rebeschini '19]

DDUCB incurs the following regret:

1. Accelerated communication:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N \ln(N)}{\sqrt{\ln(1/|\lambda_2|)}} \sum_{k=1}^{K} \Delta_k.$$

2. Unaccelerated communication:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \sum_{k=1}^{K} \Delta_k.$$

The red term in both cases is NC, up to constants.
Acceleration needs less delay in the difficult regimes ($|\lambda_2|$ close to 1).

Interpretation of the regret

DDUCB (acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N \ln(N)}{\sqrt{\ln(1/|\lambda_2|)}} \sum_{k=1}^{K} \Delta_k.$$

Interpretation of the regret

DDUCB (acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N\ln(N)}{\sqrt{\ln(1/|\lambda_2|)}} \sum_{k=1}^{K} \Delta_k.$$

Centralized algorithm

$$R(TN) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \sum_{k=1}^{K} \Delta_k.$$

Interpretation of the regret

DDUCB (acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N \ln(N)}{\sqrt{\ln(1/|\lambda_2|)}} \sum_{k=1}^{K} \Delta_k.$$

Centralized algorithm

$$R(TN) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \sum_{k=1}^{K} \Delta_k.$$

No communication:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{N\sigma^2 \ln(T)}{\Delta_k} + N \sum_{k=1}^{K} \Delta_k$$

Interpretation of the regret

DDUCB (acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N \ln(N)}{\sqrt{\ln(1/|\lambda_2|)}} \sum_{k=1}^{K} \Delta_k.$$

Centralized algorithm

$$R(TN) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \sum_{k=1}^{K} \Delta_k.$$

No communication:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{N\sigma^2 \ln(T)}{\Delta_k} + N \sum_{k=1}^{K} \Delta_k$$

Lower bound (straighforward):

$$R(T) = \Omega\left(\sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \left(\frac{N}{K} + 1\right) \sum_{k=1}^{K} \Delta_k\right)$$

Interpretation of the regret

DDUCB (no acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \sum_{k=1}^{K} \Delta_k.$$

Interpretation of the regret

DDUCB (no acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N\ln(N)}{\ln(1/|\lambda_2|)} \sum_{k=1}^{K} \Delta_k.$$

DDUCB without hiding the stages hyperparameter ε and the exploration parameter η:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\eta(1+\varepsilon)\sigma^2 \ln(TN)}{\Delta_k} + \left(\frac{N\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} + \frac{\eta}{\eta-1}\right) \sum_{k=1}^{K} \Delta_k.$$

Interpretation of the regret

DDUCB (no acceleration):

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\sigma^2 \ln(TN)}{\Delta_k} + \frac{N\ln(N)}{\ln(1/|\lambda_2|)} \sum_{k=1}^{K} \Delta_k.$$

DDUCB without hiding the stages hyperparameter ε and the exploration parameter η:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \frac{\eta(1+\varepsilon)\sigma^2 \ln(TN)}{\Delta_k} + \left(\frac{N\ln(N/\varepsilon)}{\ln(1/|\lambda_2|)} + \frac{\eta}{\eta-1}\right) \sum_{k=1}^{K} \Delta_k.$$

Algorithm in (**?**), coopUCB:

$$R(T) \lesssim \sum_{k:\Delta_k>0} \sum_{j=1}^{N} \frac{\gamma(1+\varepsilon_c^j)}{N\Delta_k} \ln(TN) + N\left(\sqrt{N}\sum_{j=2}^{N} \frac{|\lambda_j|}{1-|\lambda_j|} + \frac{\gamma}{\gamma-1}\right) \sum_{k=1}^{K} \Delta_k$$

Comparison with previous work

- **?** designed and analyzed coopUCB.
- coopUCB runs a decentralized UCB but with no delays.
- Size of confidence intervals increases to compensate for the inaccuracy of the estimation.
- It needs the knowledge of more data about the graph (whole spectrum of P and its eigenvectors).

Comparison with previous work

For coopUCB, the algorithm in (?) the regret is:

$$R(T) \lesssim \alpha \sum_{k:\Delta_k>0} \frac{\sigma^2 \log TN}{\Delta_k} + \beta \sum_{k=1}^{K} \Delta_k$$

and

$$1 \lesssim \alpha \quad ; \quad \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \lesssim \beta.$$

Comparison with previous work

For coopUCB, the algorithm in (**?**) the regret is:

$$R(T) \lesssim \alpha \sum_{k:\Delta_k>0} \frac{\sigma^2 \log TN}{\Delta_k} + \beta \sum_{k=1}^{K} \Delta_k$$

and

$$1 \lesssim \alpha \quad ; \quad \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \lesssim \beta.$$

- Communication is trivial in the case of a complete graph. It is a batched bandit problem.

Comparison with previous work

For coopUCB, the algorithm in (**?**) the regret is:

$$R(T) \lesssim \alpha \sum_{k:\Delta_k>0} \frac{\sigma^2 \log TN}{\Delta_k} + \beta \sum_{k=1}^{K} \Delta_k$$

and

$$1 \lesssim \alpha \quad ; \quad \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \lesssim \beta.$$

- Communication is trivial in the case of a complete graph. It is a batched bandit problem.
- Asymptotic regret for that case and highly connected graphs is the same for DDUCB and coopUCB.

Comparison with previous work

For coopUCB, the algorithm in (**?**) the regret is:

$$R(T) \lesssim \alpha \sum_{k:\Delta_k>0} \frac{\sigma^2 \log TN}{\Delta_k} + \beta \sum_{k=1}^{K} \Delta_k$$

and

$$1 \lesssim \alpha \quad ; \quad \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \lesssim \beta.$$

- Communication is trivial in the case of a complete graph. It is a batched bandit problem.
- Asymptotic regret for that case and highly connected graphs is the same for DDUCB and coopUCB.
- We obtain a graph independent constant in the first term and a better bound in the second term for general graphs.

Comparison with previous work

For coopUCB, the algorithm in (?) the regret is:

$$R(T) \lesssim \alpha \sum_{k:\Delta_k>0} \frac{\sigma^2 \log TN}{\Delta_k} + \beta \sum_{k=1}^{K} \Delta_k$$

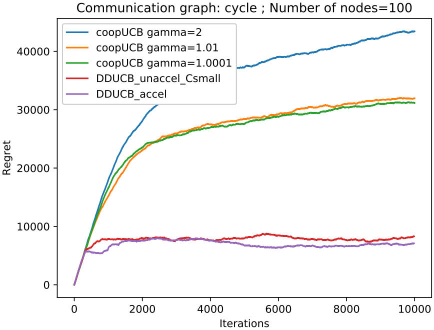
and

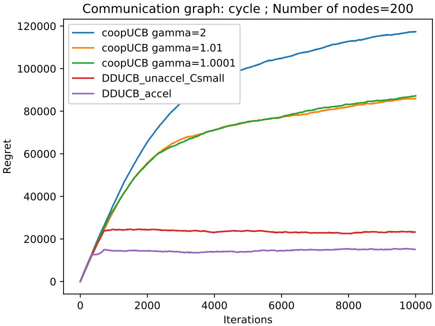
$$1 \lesssim \alpha \quad ; \quad \frac{N \ln(N)}{\ln(1/|\lambda_2|)} \lesssim \beta.$$

Example:

Graph: Cycle	α	β
[Landgren et al. '16]	$\Theta(N^2)$	$\Theta(N^{7/2})$
DDUCB (unacc)	$\Theta(1)$	$\Theta(N^3 \log(N))$
DDUCB (acc)	$\Theta(1)$	$\Theta(N^2 \log(N))$

Simulations

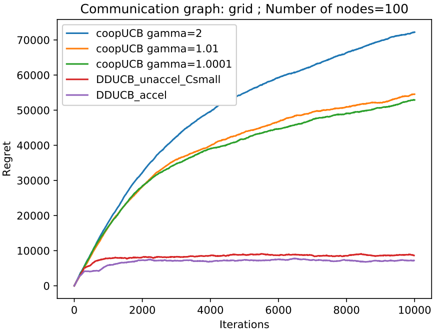


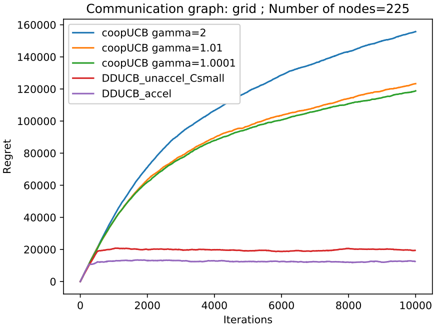


Cycle with $N = 100$ (left) and $N = 200$ (right).

$K = 17$; optimal arm $\mathcal{N}(1, 1)$; sub-optimal arms $\mathcal{N}(0.8, 1)$

Simulations





Square grid with $N = 100$ (left) and $N = 225$ (right).

$K = 17$; optimal arm $\mathcal{N}(1, 1)$; sub-optimal arms $\mathcal{N}(0.8, 1)$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$
- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$
- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell$$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$
- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$
- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\textbf{pulls from arm } k \\ \textbf{up to } d_t\}}} \text{weight}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \leq \frac{\varepsilon}{N}$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$
- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \leq \frac{\varepsilon}{N}$
  - This yields $n_{t,v}^k \geq (1 - \varepsilon) n_{d_t}^k$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$

- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \leq \frac{\varepsilon}{N}$
  - This yields $n_{t,v}^k \geq (1 - {}^{k}\varepsilon) n_{d_t}$
  - Delay does not increase regret by much: τ-delay yields increase $\tau \sum_k \Delta_k$

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$

- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\ell \in \{\text{pulls from arm } k \text{ up to } d_t\}} \text{weight}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \leq \frac{\varepsilon}{N}$
  - This yields $n_{t,v}^k \geq (1 -{}^{k}\varepsilon) n_{d_t}$
  - Delay does not increase regret by much: τ-delay yields increase $\tau \sum_k \Delta_k$

- **Control prob. of playing sub-optimal arm given agents' information:**

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$

- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell \qquad \frac{m_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell \cdot \text{reward}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \leq \frac{\varepsilon}{N}$
  - This yields $n_{t,v}^k \geq (1 - {}^k\varepsilon) n_{d_t}$
  - Delay does not increase regret by much: τ-delay yields increase $\tau \sum_k \Delta_k$

- **Control prob. of playing sub-optimal arm given agents' information:**

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$
- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell \qquad \frac{m_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell \cdot \text{reward}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \le \frac{\varepsilon}{N}$
  - This yields $n_{t,v}^k \ge (1 -^{k}\varepsilon) n_{d_t}$
  - Delay does not increase regret by much: τ-delay yields increase $\tau \sum_k \Delta_k$
- **Control prob. of playing sub-optimal arm given agents' information:**
  - Control on weights yields $\mathbf{P}\left(\frac{m_{t,v}^k}{n_{t,v}^k} - \mu_k \ge \sqrt{\frac{2\eta\sigma^2 \log s}{n_{t,v}^k}}\right) \le \frac{1}{s^{\eta+1}}$ (and analogous inequality for the other side).

Proof Sketch

- Network Regret $= \sum_{k=1}^{K} \Delta_k \mathbf{E}\left[n_T^k\right]$

- **Relate network's quantities (n_t^k) to agents' approximations ($n_{t,v}^k$):**
  - Variables used by agents to make decisions, using delayed information:

$$\frac{n_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell \qquad \frac{m_{t,v}^k}{N} = \sum_{\substack{\ell \in \{\text{pulls from arm } k \\ \text{up to } d_t\}}} \text{weight}_\ell \cdot \text{reward}_\ell$$

  - $\text{weight}_\ell = q_C(P)_{vw}$ for some $w \in V$, $\text{delay}(t) := d_t := K + \lfloor \frac{t-K}{C} \rfloor C - C$
  - Accelerated mixing yields $|\text{weight}_\ell - \frac{1}{N}| \leq \frac{\varepsilon}{N}$
  - This yields $n_{t,v}^k \geq (1-\varepsilon) n_{d_t}^k$
  - Delay does not increase regret by much: τ-delay yields increase $\tau \sum_k \Delta_k$

- **Control prob. of playing sub-optimal arm given agents' information:**
  - Control on weights yields $\mathbf{P}\left(\frac{m_{t,v}^k}{n_{t,v}^k} - \mu_k \geq \sqrt{\frac{2\eta\sigma^2 \log s}{n_{t,v}^k}}\right) \leq \frac{1}{s^{\eta+1}}$ (and analogous inequality for the other side).
  - This yields $\mathbf{P}\left(I_{t,v} = k \,|\, n_{t,v}^k > \frac{16\eta\sigma^2 \ln(N d_t)}{\Delta_k^2}\right) \leq \frac{2}{(N d_t)^\eta}$

Variants

1. Add the local information obtained during a stage to the UCB estimation.

Variants

1. Add the local information obtained during a stage to the UCB estimation.
2. Make neighbors send you their local information during the stage and use it as above.

Variants

1. Add the local information obtained during a stage to the UCB estimation.
2. Make neighbors send you their local information during the stage and use it as above.
3. In accelerated mixing, keep mixing rewards with the unaccelerated method, after a mixing stage.

Variants

1. Add the local information obtained during a stage to the UCB estimation.
2. Make neighbors send you their local information during the stage and use it as above.
3. In accelerated mixing, keep mixing rewards with the unaccelerated method, after a mixing stage.
4. No need to know $|\lambda_2|$, just an upper bound. Regret increases accordingly.

Variants

1. Add the local information obtained during a stage to the UCB estimation.
2. Make neighbors send you their local information during the stage and use it as above.
3. In accelerated mixing, keep mixing rewards with the unaccelerated method, after a mixing stage.
4. No need to know $|\lambda_2|$, just an upper bound. Regret increases accordingly.
5. Reduce communication at the expense of increasing delay: send the same information in more steps. (Analysis is straightforward since we worked with an arbitrary fixed delay).

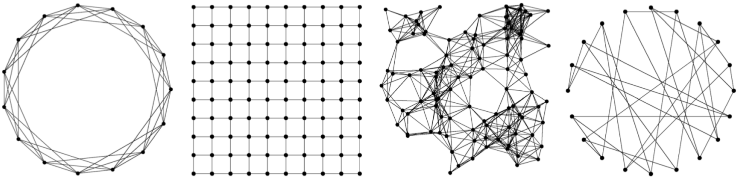
Spectral gap and regret

Graphs with greater spectral gap benefit more from decentralization.

Terms of the regret for DDUCB, if regret is
$R(T) \lesssim \alpha \sum_{k:\Delta_k>0} \frac{\sigma^2 \log TN}{\Delta_k} + \beta \sum_{k=1}^{K} \Delta_k$

Graph	Cycle	Square Grid	Random Geom. Graph	Expander
Samples per it.	N	N	N	N
α	$O(1)$	$O(1)$	$O(1)$	$O(1)$
β	$\tilde{O}(N^2)$	$\tilde{O}(N^{3/2})$	$\tilde{O}(N^{3/2})$ *	$\tilde{O}(N)$

*Second term of regret with high probability for a random geometric graph on $[0,1]^2$ with connectivity radius $\Omega(\sqrt{\log^{1+\varepsilon} n/n})$ (for any $\varepsilon > 0$).



Conclusion

- New algorithm based on graph-dependent mixing delay.
  - Optimal in complete graphs and improvement for graphs with geometry.

Conclusion

- New algorithm based on graph-dependent mixing delay.
  - Optimal in complete graphs and improvement for graphs with geometry.
- First algorithm with size-independent $\log T$-term in regret.

Conclusion

- New algorithm based on graph-dependent mixing delay.
  - Optimal in complete graphs and improvement for graphs with geometry.
- First algorithm with size-independent $\log T$-term in regret.
- Communication can be accelerated using Chebyshev polynomials.

Conclusion

- New algorithm based on graph-dependent mixing delay.
  - Optimal in complete graphs and improvement for graphs with geometry.
- First algorithm with size-independent $\log T$-term in regret.
- Communication can be accelerated using Chebyshev polynomials.
- **Trade-off** between regret and communication (graph & delay decisions).