Flickering Multi-Armed Bandits

Learning under Evolving Connectivity Constraints

8th Annual Learning for Dynamics & Control Conference (L4DC 2026)

Oral Presentation









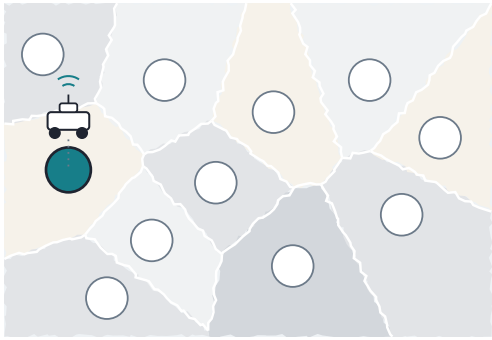
Sourav Chakraborty[1⋆] **Amit Kiran Rege**[1⋆] Claire Monteleoni[1,2] Lijun Chen[1]

[1]University of Colorado Boulder [2]INRIA Paris [⋆]Equal contribution

A Robot Scout After a Disaster

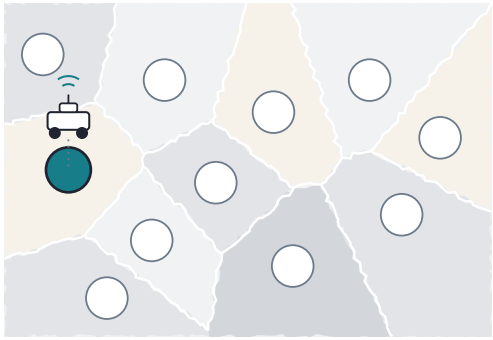
A Robot Scout After a Disaster

- A robot scouts a disaster zone to place a relay at the **best** site. Can use open streets.



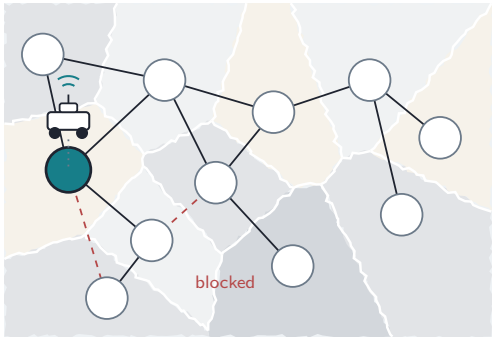
A Robot Scout After a Disaster

- A robot scouts a disaster zone to place a relay at the **best** site. Can use open streets.
- Each site has an **unknown** signal quality, learned only by visiting.



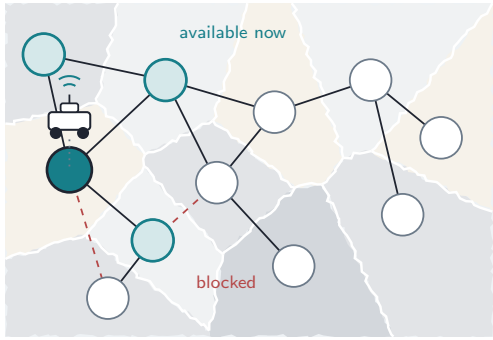
A Robot Scout After a Disaster

- A robot scouts a disaster zone to place a relay at the **best** site. Can use open streets.
- Each site has an **unknown** signal quality, learned only by visiting.
- Debris blocks and reopens streets, so the map **flickers** every round.



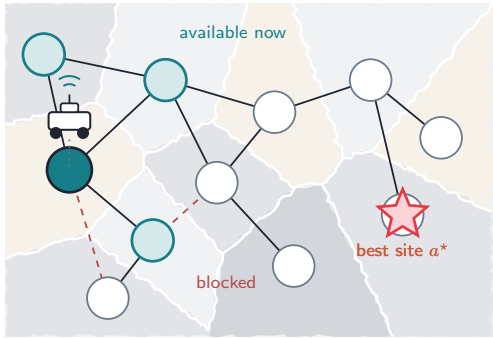
A Robot Scout After a Disaster

- A robot scouts a disaster zone to place a relay at the **best** site. Can use open streets.
- Each site has an **unknown** signal quality, learned only by visiting.
- Debris blocks and reopens streets, so the map **flickers** every round.
- It can move only to an **adjacent** site.



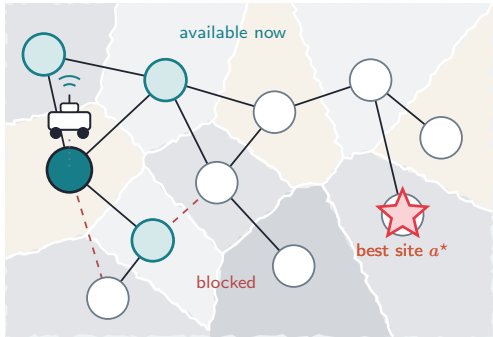
A Robot Scout After a Disaster

- A robot scouts a disaster zone to place a relay at the **best** site. Can use open streets.
- Each site has an **unknown** signal quality, learned only by visiting.
- Debris blocks and reopens streets, so the map **flickers** every round.
- It can move only to an **adjacent** site.
- The best site $a^\star$ may be **far** but must still find it and **commit**.



A Robot Scout After a Disaster

- A robot scouts a disaster zone to place a relay at the **best** site. Can use open streets.
- Each site has an **unknown** signal quality, learned only by visiting.
- Debris blocks and reopens streets, so the map **flickers** every round.
- It can move only to an **adjacent** site.
- The best site $a^\star$ may be **far** but must still find it and **commit**.



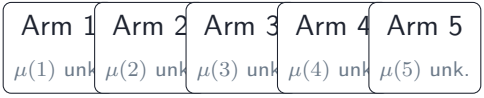
This looks like a **multi-armed bandit** problem with a twist. We'll recall classical bandits next.

Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment

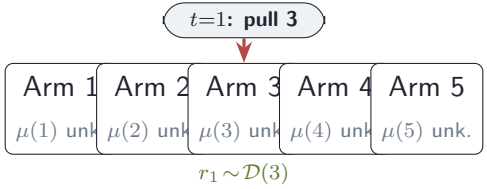
Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment



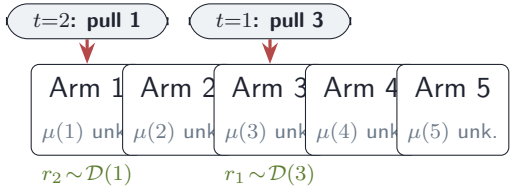
Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment



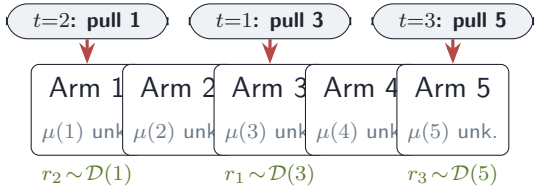
Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment



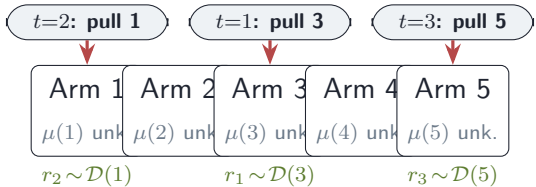
Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment



Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment



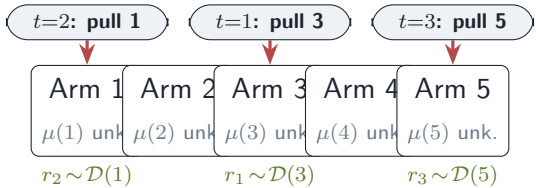
Score it (Expected Regret):

$$\mathbb{E}\left[R(T)\right] = \sum_{t=1}^{T} \left(\mu(a^\star) - \mu(a_t)\right)$$

(reward gap to the best arm, summed over time)

Classical Bandits: Pull, Observe, Repeat

Stripping away the map for a moment



Score it (Expected Regret):

$$\mathbb{E}\left[R(T)\right] = \sum_{t=1}^{T} \left(\mu(a^\star) - \mu(a_t)\right)$$

(reward gap to the best arm, summed over time)

Goal: a strategy $a_1, \ldots, a_T$ balancing *explore* vs. *exploit*, such that

$$\lim_{T\to\infty} \frac{\mathbb{E}[R(T)]}{T} = 0$$

(sublinear regret)

The Same Problem, Redrawn as a Graph

Contribution I: The Flickering Multi-Armed Bandit Model

The Same Problem, Redrawn as a Graph

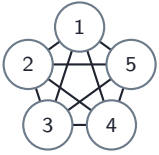
Contribution I: The Flickering Multi-Armed Bandit Model

the same n arms



≡

drawn as a graph



The Same Problem, Redrawn as a Graph

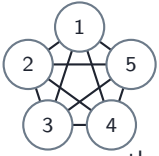
Contribution I: The Flickering Multi-Armed Bandit Model

the same n arms



$\equiv$

drawn as a graph



You can pull *any* arm at every time.

$\equiv$

From *any* arm, **every** other arm is reachable in one step.

The Same Problem, Redrawn as a Graph

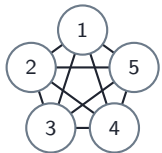
Contribution I: The Flickering Multi-Armed Bandit Model

the same n arms



$\equiv$

drawn as a graph



You can pull *any* arm at every time.

$\equiv$

From *any* arm, **every** other arm is reachable in one step.

What if the graph isn't complete? $\rightarrow$ only a *subset* of nodes reachable in a step

The Same Problem, Redrawn as a Graph

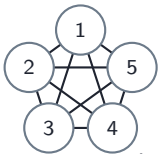
Contribution I: The Flickering Multi-Armed Bandit Model

the same n arms



$\equiv$

drawn as a graph



You can pull *any* arm at every time.

$\equiv$

From *any* arm, **every** other arm is reachable in one step.

What if the graph isn't complete? $\rightarrow$ only a *subset* of nodes reachable in a step

What if that graph also changes every round? $\rightarrow$ *flickering*.

The Same Problem, Redrawn as a Graph

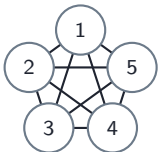
Contribution I: The Flickering Multi-Armed Bandit Model

the same n arms



$\equiv$

drawn as a graph



You can pull *any* arm at every time.

$\equiv$

From *any* arm, **every** other arm is reachable in one step.

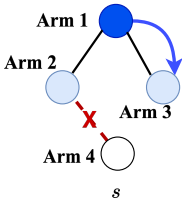
What if the graph isn't complete? $\rightarrow$ only a *subset* of nodes reachable in a step

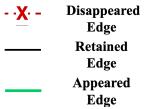
What if that graph also changes every round? $\rightarrow$ *flickering*.

In plain bandit language: the set of **available arms changes every round** and arm availability depends on **the last pulled arm.**

The Graph Flickers: Three Snapshots

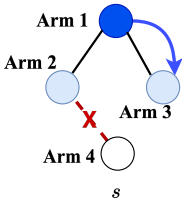
The Graph Flickers: Three Snapshots

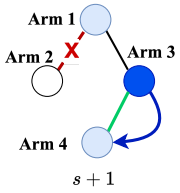


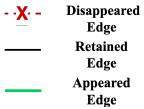


The **dark blue node** is the agent's **previously selected arm**, and the **light blue nodes** are the **current arm options** for selecting in this time step.

The Graph Flickers: Three Snapshots

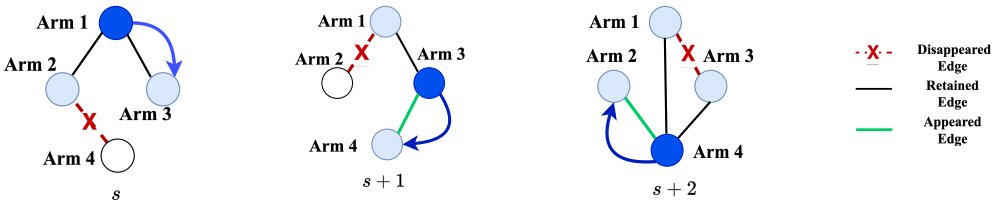






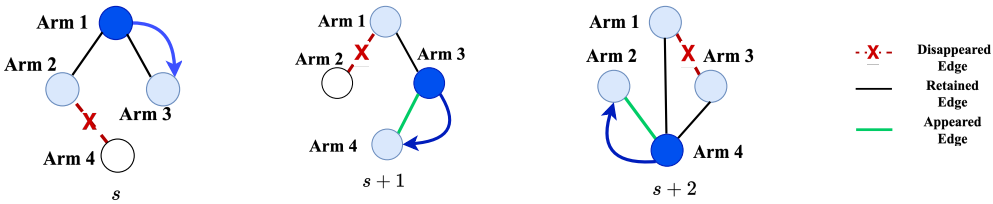
The **dark blue node** is the agent's **previously selected arm**, and the **light blue nodes** are the **current arm options** for selecting in this time step.

The Graph Flickers: Three Snapshots



The **dark blue node** is the agent's **previously selected arm**, and the **light blue nodes** are the **current arm options** for selecting in this time step.

The Graph Flickers: Three Snapshots



The **dark blue node** is the agent's **previously selected arm**, and the **light blue nodes** are the **current arm options** for selecting in this time step.

How do we model this graph evolution?

Modeling the Flickering Connectivity

Memoryless vs. Sticky Graph Evolution

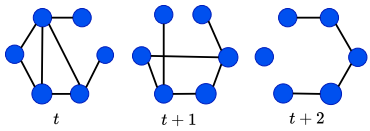
Modeling the Flickering Connectivity

Memoryless vs. Sticky Graph Evolution

Model I: i.i.d. Erdős–Rényi.

At each round, every potential edge is formed via an independent Bernoulli trial with success probability p.

Every snapshot is a completely fresh draw.



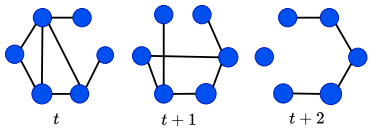
Modeling the Flickering Connectivity

Memoryless vs. Sticky Graph Evolution

Model I: i.i.d. Erdős–Rényi.

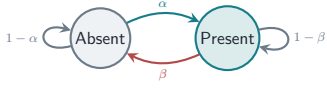
At each round, every potential edge is formed via an independent Bernoulli trial with success probability p.

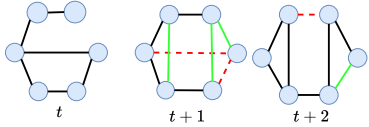
Every snapshot is a completely fresh draw.



Model II: Edge-Markovian

Each edge evolves independently via a two-state Markov chain shown below:





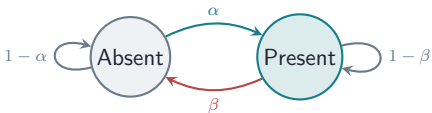
Edge-Markovian Graphs Are Markov Chains

From one edge to the whole graph

Edge-Markovian Graphs Are Markov Chains

From one edge to the whole graph

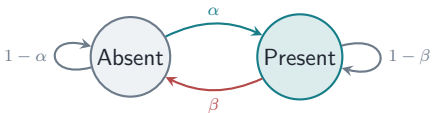
One edge: a two-state Markov chain



Edge-Markovian Graphs Are Markov Chains

From one edge to the whole graph

One edge: a two-state Markov chain

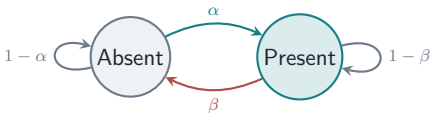


$$P_e = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

Edge-Markovian Graphs Are Markov Chains

From one edge to the whole graph

One edge: a two-state Markov chain



$$P_e = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

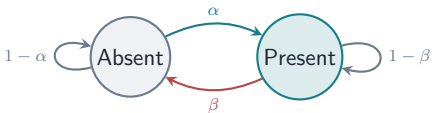
Whole graph: product chain over all edges

$$X_t = \left(X_t^e\right)_{e \in \mathcal{E}} \in \{0,1\}^{\binom{n}{2}}$$

Edge-Markovian Graphs Are Markov Chains

From one edge to the whole graph

One edge: a two-state Markov chain



$$P_e = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

Whole graph: product chain over all edges

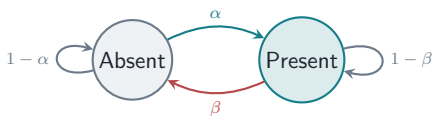
$$X_t = \left(X_t^e\right)_{e \in \mathcal{E}} \in \{0,1\}^{\binom{n}{2}}$$

$$K(x,y) = \prod_{e \in \mathcal{E}} P_e(x_e, y_e)$$

Edge-Markovian Graphs Are Markov Chains

From one edge to the whole graph

One edge: a two-state Markov chain



$$P_e = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

Whole graph: product chain over all edges

$$X_t = \left(X_t^e\right)_{e \in \mathcal{E}} \in \{0,1\}^{\binom{n}{2}}$$

$$K(x,y) = \prod_{e \in \mathcal{E}} P_e(x_e, y_e)$$

$$p_\infty = \Pr(e \in E_t \text{ at stationarity}) = \frac{\alpha}{\alpha+\beta}.$$

Burn-in Is Not Enough

Good snapshots are not fresh snapshots

Burn-in Is Not Enough

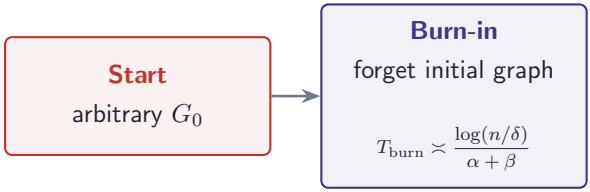
Good snapshots are not fresh snapshots

Start

arbitrary G_0

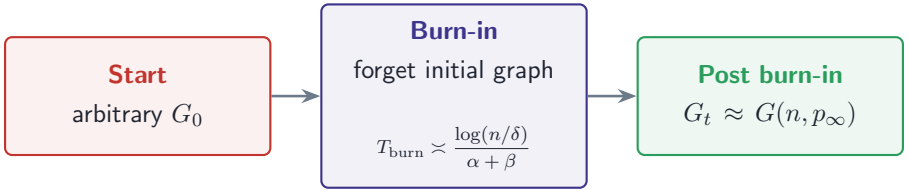
Burn-in Is Not Enough

Good snapshots are not fresh snapshots



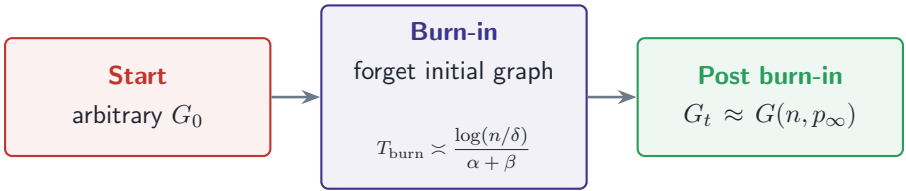
Burn-in Is Not Enough

Good snapshots are not fresh snapshots



Burn-in Is Not Enough

Good snapshots are not fresh snapshots



What burn-in gives

$$G_t \approx G(n, p_\infty)$$

each individual snapshot is well behaved

What it does not give

$$G_t \not\perp G_{t+1}$$

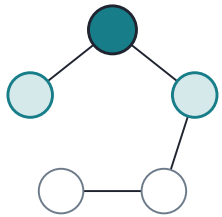
consecutive graphs are still correlated

Our Algorithm: A Lazy Random Walk, Then Commit

Contribution II: A Simple, Robust Algorithm

Our Algorithm: A Lazy Random Walk, Then Commit

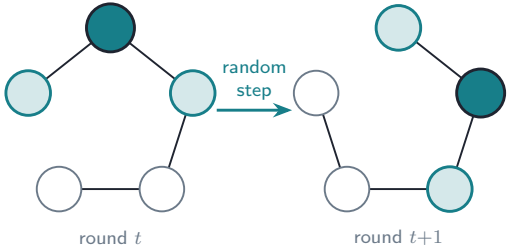
Contribution II: A Simple, Robust Algorithm



round t

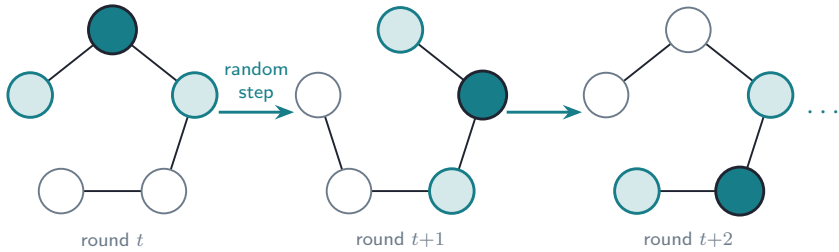
Our Algorithm: A Lazy Random Walk, Then Commit

Contribution II: A Simple, Robust Algorithm



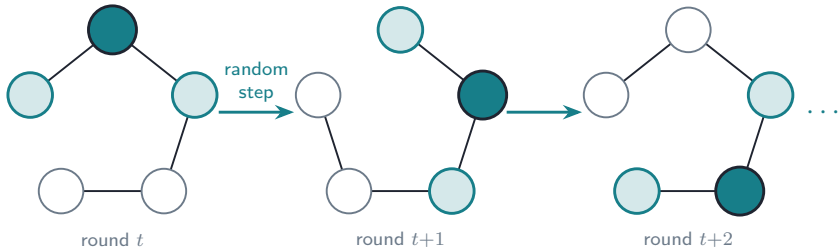
Our Algorithm: A Lazy Random Walk, Then Commit

Contribution II: A Simple, Robust Algorithm



Our Algorithm: A Lazy Random Walk, Then Commit

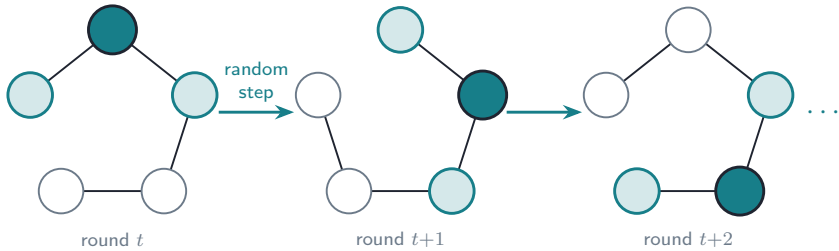
Contribution II: A Simple, Robust Algorithm



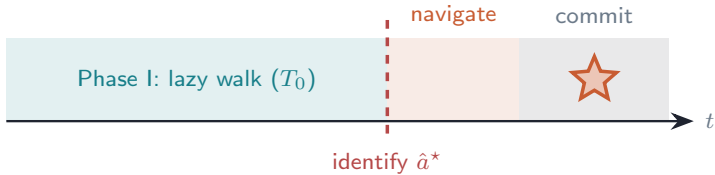
We run this lazy walk for T_0 rounds, then **commit** to the best arm seen.

Our Algorithm: A Lazy Random Walk, Then Commit

Contribution II: A Simple, Robust Algorithm



We run this lazy walk for T_0 rounds, then **commit** to the best arm seen.



Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1 - \delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1 - \delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

$$R(T) \leq$$

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1 - \delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

$$R(T) \;\leq\; \underbrace{c_1\, n \log \tfrac{nT}{\delta} \;+\; c_2\, \frac{n \log(n/\delta)}{\Delta_{\min}^2}}_{\text{exploration}}$$

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1 - \delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

$$R(T) \;\leq\; \underbrace{c_1\, n \log \tfrac{nT}{\delta} \;+\; c_2\, \frac{n \log(n/\delta)}{\Delta_{\min}^2}}_{\text{exploration}} \;+\; \underbrace{c_3\, n \log \tfrac{n}{\delta}}_{\text{navigation}}\,.$$

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1 - \delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

$$R(T) \;\leq\; \underbrace{c_0 \frac{\log(nT/\delta)}{\alpha + \beta}}_{\text{burn-in (Markov only)}} + \underbrace{c_1\, n \log \tfrac{nT}{\delta} \;+\; c_2 \frac{n \log(n/\delta)}{\Delta_{\min}^2}}_{\text{exploration}} + \underbrace{c_3\, n \log \tfrac{n}{\delta}}_{\text{navigation}}.$$

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1 - \delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

$$R(T) \;\leq\; \underbrace{c_0 \frac{\log(nT/\delta)}{\alpha + \beta}}_{\text{burn-in (Markov only)}} + \underbrace{c_1\, n \log \tfrac{nT}{\delta} \;+\; c_2 \frac{n \log(n/\delta)}{\Delta_{\min}^2}}_{\text{exploration}} + \underbrace{c_3\, n \log \tfrac{n}{\delta}}_{\text{navigation}}.$$

Burn-in and navigation are lower-order, so the bound collapses:

Expected regret, identical for both graph models

$$\mathbb{E}[R(T)] \;=\; \mathcal{O}\left(\frac{n \log(nT)}{\Delta_{\min}^2}\right), \quad \text{sublinear and free of all graph parameters.}$$

Main Result: Sublinear Regret in Three Pieces

Contribution III: A Unified Analysis

- For **both** graph models, with probability $\geq 1-\delta$ ($\Delta_{\min}$ is the gap to the second-best arm):

$$R(T) \;\leq\; \underbrace{c_0 \frac{\log(nT/\delta)}{\alpha+\beta}}_{\text{burn-in (Markov only)}} + \underbrace{c_1\, n \log \tfrac{nT}{\delta} \;+\; c_2 \frac{n \log(n/\delta)}{\Delta_{\min}^2}}_{\text{exploration}} + \underbrace{c_3\, n \log \tfrac{n}{\delta}}_{\text{navigation}}.$$

Burn-in and navigation are lower-order, so the bound collapses:

Expected regret, identical for both graph models

$$\mathbb{E}[R(T)] \;=\; \mathcal{O}\left(\frac{n \log(nT)}{\Delta_{\min}^2}\right), \quad \text{sublinear and free of all graph parameters.}$$

- **Near-optimal**: matches the floor $\Omega\big(n \log T/\Delta_{\min}^2\big)$ up to logs, so locality is essentially free.

Proof Outline

Contribution III: A Unified Analysis

Explore then commit, in four steps:

Proof Outline

Contribution III: A Unified Analysis

Explore then commit, in four steps:

1. **Cover** the arms: reach each one enough times.

Proof Outline

Contribution III: A Unified Analysis

Explore then commit, in four steps:

1. **Cover** the arms: reach each one enough times.
2. **Identify** the best arm.

Proof Outline

Contribution III: A Unified Analysis

Explore then commit, in four steps:

1. **Cover** the arms: reach each one enough times.
2. **Identify** the best arm.
3. **Navigate** to it and commit.

Proof Outline

Contribution III: A Unified Analysis

Explore then commit, in four steps:

1. **Cover** the arms: reach each one enough times.
2. **Identify** the best arm.
3. **Navigate** to it and commit.
4. **Add up** the costs.

Proof Outline

Contribution III: A Unified Analysis

Explore then commit, in four steps:

1. **Cover** the arms: reach each one enough times.
2. **Identify** the best arm.
3. **Navigate** to it and commit.
4. **Add up** the costs.

- Steps 2 to 4 are standard bandit arguments. **Coverage** is the only hard step.

Coverage, in one line

$$T_{\text{exp}} \approx \frac{s_0}{\pi_{\min}}$$

samples needed per arm

over *minimum visit frequency* $\pi_{\min}$

Coverage

Contribution III: A Unified Analysis

Coverage

Contribution III: A Unified Analysis

The whole proof is keeping the minimum visit frequency $\pi_{\min}$ bounded below as the graph gets harder.

Coverage

Contribution III: A Unified Analysis

The whole proof is keeping the minimum visit frequency $\pi_{\min}$ bounded below as the graph gets harder.



Homogeneous ER

every arm looks alike

$$\pi_{\min} = \frac{1}{n}$$

regeneration

Coverage

Contribution III: A Unified Analysis

The whole proof is keeping the minimum visit frequency $\pi_{\min}$ bounded below as the graph gets harder.

the graph gets harder →

Homogeneous ER

every arm looks alike

$$\pi_{\min} = \frac{1}{n}$$

regeneration

Heterogeneous ER

arms differ in reach

$$\pi_{\min} = \frac{1}{n_{\text{eff}}}$$

a low-degree arm costs more

\+ **typicality, Cheeger**

Coverage

Contribution III: A Unified Analysis

The whole proof is keeping the minimum visit frequency $\pi_{\min}$ bounded below as the graph gets harder.

the graph gets harder →

Homogeneous ER

every arm looks alike

$$\pi_{\min} = \frac{1}{n}$$

regeneration

Heterogeneous ER

arms differ in reach

$$\pi_{\min} = \frac{1}{n_{\text{eff}}}$$

a low-degree arm costs more

+ **typicality, Cheeger**

Edge-Markovian

the floor itself drifts

$$\pi_{\min} = \frac{1}{n} - \text{drift}$$

positive if walk out-mixes drift, $\beta \lesssim 1/n$

+ **tracking, burn-in**

Contributions

Three Things We Just Walked Through

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Simple algorithm

a lazy random walk, explore then commit, near-optimal

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Simple algorithm

a lazy random walk, explore then commit, near-optimal

New tools

mixing + concentration tools, matching lower bound

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Simple algorithm

a lazy random walk, explore then commit, near-optimal

New tools

mixing + concentration tools, matching lower bound

Where this goes next:

- **Navigation-aware UCB.** Explore and exploit at once, with optimism that also budgets the cost of travelling to a distant arm.

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Simple algorithm

a lazy random walk, explore then commit, near-optimal

New tools

mixing + concentration tools, matching lower bound

Where this goes next:

- **Navigation-aware UCB.** Explore and exploit at once, with optimism that also budgets the cost of travelling to a distant arm.
- **Minimal conditions.** How fast can the map flicker before learning fails? The threshold is open both in bandits and in the math literature.

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Simple algorithm

a lazy random walk, explore then commit, near-optimal

New tools

mixing + concentration tools, matching lower bound

Where this goes next:

- **Navigation-aware UCB.** Explore and exploit at once, with optimism that also budgets the cost of travelling to a distant arm.
- **Minimal conditions.** How fast can the map flicker before learning fails? The threshold is open both in bandits and in the math literature.
- **Unknown Graph.** Need to balance exploration of the graph with understanding it's dynamic nature.

Contributions

Three Things We Just Walked Through

New model

FMAB: arms flicker on a local, evolving graph

Simple algorithm

a lazy random walk, explore then commit, near-optimal

New tools

mixing + concentration tools, matching lower bound

Where this goes next:

- **Navigation-aware UCB.** Explore and exploit at once, with optimism that also budgets the cost of travelling to a distant arm.
- **Minimal conditions.** How fast can the map flicker before learning fails? The threshold is open both in bandits and in the math literature.
- **Unknown Graph.** Need to balance exploration of the graph with understanding it's dynamic nature.

Takeaway: When you must *move* to learn, a lazy random walk is almost all you need.

Thank you

Questions welcome

"When you must move to learn, a lazy random walk is almost all you need."