

Multi-Player Bandits Revisited

Decentralized Multi-Player Multi-Arm Bandits

Lilian Besson

Advised by Christophe Moy Émilie Kaufmann

PhD Student

Team SCEE, IETR, CentraleSupélec, Rennes
& Team SequeL, CRIStAL, Inria, Lille

ENSAI “BrownBag” Seminar – 24 January 2018



Motivation

We control some communicating devices, they want to access to an access point.

- Insert them in a **crowded wireless network**.
- With a protocol **slotted in both time and frequency**.

Goal

- Maintain a **good Quality of Service**.
- **With no centralized control** as it costs network overhead.

How?

- Devices can choose a different radio channel at each time
↪ learn the best one with sequential algorithm!

Outline

- ② Our model: 3 different feedback levels
- ③ Regret lower bound

- ⑤ Two new multi-player decentralized algorithms
- ⑥ Upper bound on regret for MCTopM
- ⑦ Experimental results

Outline and reference

- ② Our model: 3 different feedback levels
- ③ Regret lower bound

- ⑤ Two new multi-player decentralized algorithms
- ⑥ Upper bound on regret for MCTopM
- ⑦ Experimental results

This is based on our latest article:

- “*Multi-Player Bandits Models Revisited*”, Besson & Kaufmann.
ALT 2018 (Lanzarote, April).

[arXiv:1711.02317](https://arxiv.org/abs/1711.02317), for

Our model

- K radio channels (e.g., 10)
- Discrete and synchronized time $t \geq 1$. Every time frame t is:

(known)

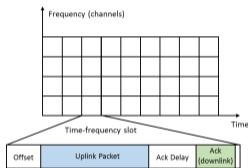


Figure 1: Protocol in time and frequency, with an *Acknowledgement*.

Dynamic device = dynamic radio reconfiguration

- It decides **each time** the channel it uses to send **each packet**.
- It can implement a simple **decision algorithm**.

Our model

“Easy” case

- $M \leq K$ devices **always communicate** and try to access the network, *independently* without centralized supervision,
- Background traffic is *i.i.d.*.

Two variants : with or without *sensing*

- ① *With sensing*: Device first senses for presence of Primary Users that have strict priority (background traffic), then use Ack to detect collisions.

Model the “classical” Opportunistic Spectrum Access problem. Not exactly suited for Internet of Things, but can model ZigBee, and can be analyzed mathematically...

Our model

“Easy” case

- $M \leq K$ devices **always communicate** and try to access the network, *independently* without centralized supervision,
- Background traffic is *i.i.d.*.

Two variants : with or without *sensing*

- ① *With sensing*: Device first senses for presence of Primary Users that have strict priority (background traffic), then use Ack to detect collisions.
Model the “classical” Opportunistic Spectrum Access problem. Not exactly suited for Internet of Things, but can model ZigBee, and can be analyzed mathematically...
- ② *Without sensing*: same background traffic, but cannot sense, so only Ack is used. More suited for “IoT” networks like LoRa or SigFox (Harder to analyze mathematically.)

Background traffic, and rewards

i.i.d. background traffic

- K channels, modeled as Bernoulli (0/1) distributions of mean $\mu_k =$ background traffic from *Primary Users*, bothering the dynamic devices,
- M devices, each uses channel $A^j(t) \in \{1, \dots, K\}$ at time t .

Rewards

$$r^j(t) := Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)}) = \mathbb{1}(\text{uplink \& Ack})$$

- with sensing information $\forall k, Y_{k,t} \stackrel{\text{iid}}{\sim} \text{Bern}(\mu_k) \in \{0, 1\}$,
- collision for device $j : C^j(t) = \mathbb{1}(\text{alone on arm } A^j(t))$.
 \hookrightarrow **combined** binary reward **but not** from two Bernoulli!

3 feedback levels

$$r^j(t) := Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)})$$

- ① “Full feedback”: observe both $Y_{A^j(t),t}$ and $C^j(t)$ separately,
↪ Not realistic enough, we don't focus on it.

3 feedback levels

$$r^j(t) := Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)})$$

- 1 “Full feedback”: observe both $Y_{A^j(t),t}$ and $C^j(t)$ separately,
↪ Not realistic enough, we don't focus on it.
- 2 “Sensing”: first observe $Y_{A^j(t),t}$, then $C^j(t)$ only if $Y_{A^j(t),t} \neq 0$,
↪ Models licensed protocols (ex. ZigBee), our main focus.

3 feedback levels

$$r^j(t) := Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)})$$

- ① “Full feedback”: observe both $Y_{A^j(t),t}$ and $C^j(t)$ separately,
 \hookrightarrow Not realistic enough, we don't focus on it.
- ② “Sensing”: first observe $Y_{A^j(t),t}$, then $C^j(t)$ only if $Y_{A^j(t),t} \neq 0$,
 \hookrightarrow Models licensed protocols (ex. ZigBee), our main focus.
- ③ “No sensing”: observe only the combined $Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)})$,
 \hookrightarrow Unlicensed protocols (ex. LoRaWAN), harder to analyze !

3 feedback levels

$$r^j(t) := Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)})$$

- ① “Full feedback”: observe both $Y_{A^j(t),t}$ and $C^j(t)$ separately,
 \hookrightarrow Not realistic enough, we don't focus on it.
- ② “Sensing”: first observe $Y_{A^j(t),t}$, then $C^j(t)$ only if $Y_{A^j(t),t} \neq 0$,
 \hookrightarrow Models licensed protocols (ex. ZigBee), our main focus.
- ③ “No sensing”: observe only the combined $Y_{A^j(t),t} \times \mathbb{1}(\overline{C^j(t)})$,
 \hookrightarrow Unlicensed protocols (ex. LoRaWAN), harder to analyze !

But all consider the same instantaneous reward $r^j(t)$.

Goal

Problem

- *Goal : minimize packet loss ratio (= maximize nb of received Ack) in a finite-space discrete-time Decision Making Problem.*
- *Solution ? **Multi-Armed Bandit algorithms, decentralized** and used **independently** by each dynamic device.*

Goal

Problem

- *Goal* : minimize packet loss ratio (= maximize nb of received Ack) in a *finite-space discrete-time Decision Making Problem*.
- *Solution* ? **Multi-Armed Bandit algorithms, decentralized** and used **independently** by each dynamic device.

Decentralized reinforcement learning optimization!

- Max transmission rate \equiv **max cumulated rewards** $\max_{\text{algorithm } A} \sum_{t=1}^T \sum_{j=1}^M r^j(t)$.
- Each player wants to **maximize its cumulated reward**,
- With no central control, and no exchange of information,
- Only possible if : each player converges to one of the M best arms, orthogonally (without collisions).

Centralized regret

A measure of success

- Not the network throughput or collision probability,
- We study the **centralized (expected) regret**:

$$R_T(\boldsymbol{\mu}, M, \rho) := \left(\sum_{k=1}^M \mu_k^* \right) T - \mathbb{E}_{\mu} \left[\sum_{t=1}^T \sum_{j=1}^M r^j(t) \right]$$

Notation : μ_k^* is the mean of the k -best arm (k -th largest in $\boldsymbol{\mu}$):

- $\mu_1^* := \max \boldsymbol{\mu}$,
- $\mu_2^* := \max \boldsymbol{\mu} \setminus \{\mu_1^*\}$,
- etc.

Centralized regret

A measure of success

- Not the network throughput or collision probability,
- We study the **centralized** (expected) **regret**:

$$R_T(\boldsymbol{\mu}, M, \rho) := \left(\sum_{k=1}^M \mu_k^* \right) T - \mathbb{E}_{\mu} \left[\sum_{t=1}^T \sum_{j=1}^M r^j(t) \right]$$

Two directions of analysis

- Clearly $R_T = \mathcal{O}(T)$, but we want a sub-linear regret, as small as possible!

Centralized regret

A measure of success

- Not the network throughput or collision probability,
- We study the **centralized** (expected) **regret**:

$$R_T(\boldsymbol{\mu}, M, \rho) := \left(\sum_{k=1}^M \mu_k^* \right) T - \mathbb{E}_{\mu} \left[\sum_{t=1}^T \sum_{j=1}^M r^j(t) \right]$$

Two directions of analysis

- Clearly $R_T = \mathcal{O}(T)$, but we want a sub-linear regret, as small as possible!
- *How good a decentralized algorithm can be in this setting?*
 \hookrightarrow **Lower Bound** on regret, for **any** algorithm !

Centralized regret

A measure of success

- Not the network throughput or collision probability,
- We study the **centralized** (expected) **regret**:

$$R_T(\boldsymbol{\mu}, M, \rho) := \left(\sum_{k=1}^M \mu_k^* \right) T - \mathbb{E}_{\mu} \left[\sum_{t=1}^T \sum_{j=1}^M r^j(t) \right]$$

Two directions of analysis

- Clearly $R_T = \mathcal{O}(T)$, but we want a sub-linear regret, as small as possible!
- *How good a decentralized algorithm can be in this setting?*
 - ↪ **Lower Bound** on regret, for **any** algorithm !
- *How good is my decentralized algorithm in this setting?*
 - ↪ **Upper Bound** on regret, for **one** algorithm !

Lower bound

- ① Decomposition of regret in 3 terms,
- ② Asymptotic lower bound of one term,
- ③ And for regret,
- ④ Proof?

Decomposition on regret

Decomposition

For any algorithm, decentralized or not, we have

$$R_T(\boldsymbol{\mu}, M, \rho) = \sum_{k \in M\text{-worst}} (\mu_M^* - \mu_k) \mathbb{E}_\mu [T_k(T)] \\ + \sum_{k \in M\text{-best}} (\mu_k - \mu_M^*) (T - \mathbb{E}_\mu [T_k(T)]) + \sum_{k=1}^K \mu_k \mathbb{E}_\mu [\mathcal{C}_k(T)].$$

Notations for an arm $k \in \{1, \dots, K\}$:

- $T_k^j(T) := \sum_{t=1}^T \mathbb{1}(A^j(t) = k)$, counts selections by the player $j \in \{1, \dots, M\}$,
- $T_k(T) := \sum_{j=1}^M T_k^j(T)$, counts selections by all M players,
- $\mathcal{C}_k(T) := \sum_{t=1}^T \mathbb{1}(\exists j_1 \neq j_2, A^{j_1}(t) = k = A^{j_2}(t))$, counts collisions.

Decomposition on regret

Decomposition

For any algorithm, decentralized or not, we have

$$R_T(\boldsymbol{\mu}, M, \rho) = \sum_{k \in M\text{-worst}} (\mu_M^* - \mu_k) \mathbb{E}_\mu [T_k(T)] \\ + \sum_{k \in M\text{-best}} (\mu_k - \mu_M^*) (T - \mathbb{E}_\mu [T_k(T)]) + \sum_{k=1}^K \mu_k \mathbb{E}_\mu [C_k(T)].$$

Small regret can be attained if...

- 1 Devices can quickly identify the bad arms M -worst, and not play them too much (*number of sub-optimal selections*),

Decomposition on regret

Decomposition

For any algorithm, decentralized or not, we have

$$R_T(\boldsymbol{\mu}, M, \rho) = \sum_{k \in M\text{-worst}} (\mu_M^* - \mu_k) \mathbb{E}_\mu [T_k(T)] \\ + \sum_{k \in M\text{-best}} (\mu_k - \mu_M^*) (T - \mathbb{E}_\mu [T_k(T)]) + \sum_{k=1}^K \mu_k \mathbb{E}_\mu [C_k(T)].$$

Small regret can be attained if...

- ① Devices can quickly identify the bad arms M -worst, and not play them too much (*number of sub-optimal selections*),
- ② Devices can quickly identify the best arms, and most surely play them (*number of optimal non-selections*),

Decomposition on regret

Decomposition

For any algorithm, decentralized or not, we have

$$R_T(\boldsymbol{\mu}, M, \rho) = \sum_{k \in M\text{-worst}} (\mu_M^* - \mu_k) \mathbb{E}_\mu [T_k(T)] \\ + \sum_{k \in M\text{-best}} (\mu_k - \mu_M^*) (T - \mathbb{E}_\mu [T_k(T)]) + \sum_{k=1}^K \mu_k \mathbb{E}_\mu [\mathcal{C}_k(T)].$$

Small regret can be attained if...

- ① Devices can quickly identify the bad arms M -worst, and not play them too much (*number of sub-optimal selections*),
- ② Devices can quickly identify the best arms, and most surely play them (*number of optimal non-selections*),
- ③ Devices can use orthogonal channels (*number of collisions*).

Lower bound on regret

Lower bound

For any algorithm, decentralized or not, we have

$$R_T(\boldsymbol{\mu}, M, \rho) \geq \sum_{k \in M\text{-worst}} (\mu_M^* - \mu_k) \mathbb{E}_\mu [T_k(T)]$$

Asymptotic Lower Bound on regret I

Theorem 1

[Besson & Kaufmann, 2017]

Sub-optimal arms selections are lower bounded asymptotically,

$$\forall \text{ player } j, \text{ bad arm } k, \quad \liminf_{T \rightarrow +\infty} \frac{\mathbb{E}_{\mu}[T_k^j(T)]}{\log T} \geq \frac{1}{\text{kl}(\mu_k, \mu_M^*)},$$

Where $\text{kl}(x, y) := x \log(\frac{x}{y}) + (1 - x) \log(\frac{1-x}{1-y})$ is the *binary* Kullback-Leibler divergence.

Proof: using technical information theory tools (Kullback-Leibler divergence, change of distributions).

Ref: [Garivier et al, 2016]

Asymptotic Lower Bound on regret II

Theorem 2

[Besson & Kaufmann, 2017]

For any uniformly efficient decentralized policy, and any non-degenerated problem μ ,

$$\liminf_{T \rightarrow +\infty} \frac{R_T(\mu, M, \rho)}{\log(T)} \geq M \times \left(\sum_{k \in M\text{-worst}} \frac{(\mu_M^* - \mu_k)}{\text{kl}(\mu_k, \mu_M^*)} \right).$$

Asymptotic Lower Bound on regret II

Theorem 2

[Besson & Kaufmann, 2017]

For any uniformly efficient decentralized policy, and any non-degenerated problem μ ,

$$\liminf_{T \rightarrow +\infty} \frac{R_T(\mu, M, \rho)}{\log(T)} \geq M \times \left(\sum_{k \in M\text{-worst}} \frac{(\mu_M^* - \mu_k)}{\text{kl}(\mu_k, \mu_M^*)} \right).$$

Remarks

- The centralized *multiple-play* lower bound is the same without the M multiplicative factor...
Ref: [Anantharam et al, 1987]
 \hookrightarrow “price of non-coordination” = M = nb of player?
- Improved state-of-the-art lower bound, but still not perfect: collisions should also be controlled!

Single-player MAB algorithms

- ① Index-based MAB deterministic policies,
- ② Upper Confidence Bound algorithm : UCB_1 ,
- ③ Kullback-Leibler UCB algorithm : kl-UCB.

Upper Confidence Bound algorithm (UCB₁)

- ① For the first K steps ($t = 1, \dots, K$), try each channel *once*.
- ② Then for the next steps $t > K$:
 - $T_k^j(t) := \sum_{s=1}^t \mathbb{1}(A^j(s) = k)$ selections of channel k ,
 - $S_k^j(t) := \sum_{s=1}^t Y_k(s) \mathbb{1}(A^j(s) = k)$ sum of sensing information.

Upper Confidence Bound algorithm (UCB₁)

- ① For the first K steps ($t = 1, \dots, K$), try each channel *once*.
- ② Then for the next steps $t > K$:

- $T_k^j(t) := \sum_{s=1}^t \mathbb{1}(A^j(s) = k)$ selections of channel k ,
- $S_k^j(t) := \sum_{s=1}^t Y_k(s) \mathbb{1}(A^j(s) = k)$ sum of sensing information.

- Compute the index $g_k^j(t) := \underbrace{\frac{S_k^j(t)}{T_k^j(t)}}_{\text{Empirical Mean } \hat{\mu}_k(t)} + \underbrace{\sqrt{\frac{\log(t)}{2 T_k^j(t)}}}_{\text{Upper Confidence Bound}}$,

Upper Confidence Bound algorithm (UCB₁)

- ① For the first K steps ($t = 1, \dots, K$), try each channel *once*.
- ② Then for the next steps $t > K$:

- $T_k^j(t) := \sum_{s=1}^t \mathbb{1}(A^j(s) = k)$ selections of channel k ,

- $S_k^j(t) := \sum_{s=1}^t Y_k(s) \mathbb{1}(A^j(s) = k)$ sum of sensing information.

- Compute the index $g_k^j(t) := \underbrace{\frac{S_k^j(t)}{T_k^j(t)}}_{\text{Empirical Mean } \hat{\mu}_k(t)} + \underbrace{\sqrt{\frac{\log(t)}{2 T_k^j(t)}}}_{\text{Upper Confidence Bound}}$,

- Choose channel $A^j(t) = \arg \max_k g_k^j(t)$,

- Update $T_k^j(t+1)$ and $S_k^j(t+1)$.

Kullback-Leibler UCB algorithm (kl-UCB)

- ① For the first K steps ($t = 1, \dots, K$), try each channel *once*.
- ② Then for the next steps $t > K$:

- $T_k^j(t) := \sum_{s=1}^t \mathbb{1}(A^j(s) = k)$ selections of channel k ,
- $S_k^j(t) := \sum_{s=1}^t Y_k(s) \mathbb{1}(A^j(s) = k)$ sum of sensing information.
- Compute the index $g_k^j(t) := \sup_{q \in [a, b]} \left\{ q : \text{kl} \left(\frac{S_k^j(t)}{T_k^j(t)}, q \right) \leq \frac{\log(t)}{T_k^j(t)} \right\}$,
- Choose channel $A^j(t) = \arg \max_k g_k^j(t)$,
- Update $T_k^j(t+1)$ and $S_k^j(t+1)$.

Kullback-Leibler UCB algorithm (kl-UCB)

- ① For the first K steps ($t = 1, \dots, K$), try each channel *once*.
- ② Then for the next steps $t > K$:

- $T_k^j(t) := \sum_{s=1}^t \mathbb{1}(A^j(s) = k)$ selections of channel k ,
- $S_k^j(t) := \sum_{s=1}^t Y_k(s) \mathbb{1}(A^j(s) = k)$ sum of sensing information.
- Compute the index $g_k^j(t) := \sup_{q \in [a, b]} \left\{ q : \text{kl} \left(\frac{S_k^j(t)}{T_k^j(t)}, q \right) \leq \frac{\log(t)}{T_k^j(t)} \right\}$,
- Choose channel $A^j(t) = \arg \max_k g_k^j(t)$,
- Update $T_k^j(t+1)$ and $S_k^j(t+1)$.

Why bother? kl-UCB is more efficient than UCB₁, and asymptotically optimal for single-player stochastic bandit.

References: [Garivier & Cappé, 2011], [Cappé & Garivier & Maillard & Munos & Stoltz, 2013]

Multi-player decentralized algorithms

- ① Common building blocks of previous algorithms,
- ② First proposal: RandTopM,
- ③ Second proposal: MCTopM,
- ④ Algorithm and illustration.

Algorithms for this easier model

Building blocks : separate the two aspects

- 1 **MAB policy** to learn the best arms (use sensing $Y_{A^j(t),t}$),
- 2 **Orthogonalization scheme** to avoid collisions (use collision indicators $C^j(t)$).

Algorithms for this easier model

Building blocks : separate the two aspects

- ① **MAB policy** to learn the best arms (use sensing $Y_{A^j(t),t}$),
- ② **Orthogonalization scheme** to avoid collisions (use collision indicators $C^j(t)$).

Many different proposals for *decentralized* learning policies

- “State-of-the-art”: RhoRand policy and variants,
- Recent approaches: MEGA and Musical Chair.

[Anandkumar et al, 2011]

[Avner & Mannor, 2015], [Shamir et al, 2016]

Algorithms for this easier model

Building blocks : separate the two aspects

- ① **MAB policy** to learn the best arms (use sensing $Y_{A^j(t),t}$),
- ② **Orthogonalization scheme** to avoid collisions (use collision indicators $C^j(t)$).

Many different proposals for *decentralized* learning policies

- “State-of-the-art”: RhoRand policy and variants,
- Recent approaches: MEGA and Musical Chair.

[Anandkumar et al, 2011]

[Avner & Mannor, 2015], [Shamir et al, 2016]

Our proposals:


[Besson & Kaufmann, 2017]

RandTopM and MCTopM are sort of mixes between RhoRand and Musical Chair, using UCB or more efficient index policies (kl-UCB).

A first decentralized algorithm (naive)

1 Let $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$ and $C^j(1) = \text{False}$

2 **for** $t = 1, \dots, T - 1$ **do**



A first decentralized algorithm (naive)

```
1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  or  $C^j(t)$  then
4      $A^j(t + 1) \sim \mathcal{U}(\widehat{M}^j(t))$  // randomly switch
5   else
6      $A^j(t + 1) = A^j(t)$  // stays on the same arm
7   end
```

A first decentralized algorithm (naive)

```

1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  or  $C^j(t)$  then
4      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t))$  // randomly switch
5   else
6      $A^j(t+1) = A^j(t)$  // stays on the same arm
7   end
8   Play arm  $A^j(t+1)$ , get new observations (sensing and collision),
9   Compute the indices  $g_k^j(t+1)$  and set  $\widehat{M}^j(t+1)$  for next step.
10 end

```

Algorithm 3: A first decentralized learning policy (for a fixed underlying index policy g^j).
 The set $\widehat{M}^j(t)$ is the M best arms according to indexes $g^j(t)$.

RandTopM algorithm

1 Let $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$ and $C^j(1) = \text{False}$

2 **for** $t = 1, \dots, T - 1$ **do**

3 **if** $A^j(t) \notin \widehat{M}^j(t)$ **then**

4 **if** $C^j(t)$ **then**

5 $A^j(t + 1) \sim \mathcal{U}(\widehat{M}^j(t))$

// collision

// randomly switch

RandTopM algorithm

```

1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  then
4     if  $C^j(t)$  then // collision
5        $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t))$  // randomly switch
6     else // aim arm with smaller UCB at  $t - 1$ 
7        $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t) \cap \{k : g_k^j(t-1) \leq g_{A^j(t)}^j(t-1)\})$ 
8     end
9   else

```

RandTopM algorithm

(also works well!)

```

1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  then
4     if  $C^j(t)$  then // collision
5        $A^j(t + 1) \sim \mathcal{U}(\widehat{M}^j(t))$  // randomly switch
6     else // aim arm with smaller UCB at  $t - 1$ 
7        $A^j(t + 1) \sim \mathcal{U}(\widehat{M}^j(t) \cap \{k : g_k^j(t - 1) \leq g_{A^j(t)}^j(t - 1)\})$ 
8     end
9   else
10     $A^j(t + 1) = A^j(t)$  // stays on the same arm
11  end
12  Play arm  $A^j(t + 1)$ , get new observations (sensing and collision),
13  Compute the indices  $g_k^j(t + 1)$  and set  $\widehat{M}^j(t + 1)$  for next step.
14 end

```

MCTopM algorithm

```
1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$  and  $s^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  then // transition (3) or (5)
4      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t) \cap \{k : g_k^j(t-1) \leq g_{A^j(t)}^j(t-1)\})$  // not empty
5      $s^j(t+1) = \text{False}$  // aim arm with smaller UCB at  $t - 1$ 
```

MCTopM algorithm

```
1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$  and  $s^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  then // transition (3) or (5)
4      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t) \cap \{k : g_k^j(t-1) \leq g_{A^j(t)}^j(t-1)\})$  // not empty
5      $s^j(t+1) = \text{False}$  // aim arm with smaller UCB at  $t - 1$ 
6   else if  $C^j(t)$  and  $\overline{s^j(t)}$  then // collision and not fixed
7      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t))$  // transition (2)
8      $s^j(t+1) = \text{False}$ 
```

MCTopM algorithm

```
1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$  and  $s^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  then // transition (3) or (5)
4      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t) \cap \{k : g_k^j(t-1) \leq g_{A^j(t)}^j(t-1)\})$  // not empty
5      $s^j(t+1) = \text{False}$  // aim arm with smaller UCB at  $t - 1$ 
6   else if  $C^j(t)$  and  $\overline{s^j(t)}$  then // collision and not fixed
7      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t))$  // transition (2)
8      $s^j(t+1) = \text{False}$ 
9   else // transition (1) or (4)
10     $A^j(t+1) = A^j(t)$  // stay on the previous arm
11     $s^j(t+1) = \text{True}$  // become or stay fixed on a "chair"
12  end
```

MCTopM algorithm

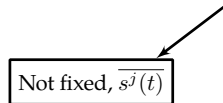
```
1 Let  $A^j(1) \sim \mathcal{U}(\{1, \dots, K\})$  and  $C^j(1) = \text{False}$  and  $s^j(1) = \text{False}$ 
2 for  $t = 1, \dots, T - 1$  do
3   if  $A^j(t) \notin \widehat{M}^j(t)$  then // transition (3) or (5)
4      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t) \cap \{k : g_k^j(t-1) \leq g_{A^j(t)}^j(t-1)\})$  // not empty
5      $s^j(t+1) = \text{False}$  // aim arm with smaller UCB at  $t - 1$ 
6   else if  $C^j(t)$  and  $\overline{s^j(t)}$  then // collision and not fixed
7      $A^j(t+1) \sim \mathcal{U}(\widehat{M}^j(t))$  // transition (2)
8      $s^j(t+1) = \text{False}$ 
9   else // transition (1) or (4)
10     $A^j(t+1) = A^j(t)$  // stay on the previous arm
11     $s^j(t+1) = \text{True}$  // become or stay fixed on a "chair"
12  end
13  Play arm  $A^j(t+1)$ , get new observations (sensing and collision),
14  Compute the indices  $g_k^j(t+1)$  and set  $\widehat{M}^j(t+1)$  for next step.
15 end
```

MCTopM algorithm

(0) Start $t = 0$

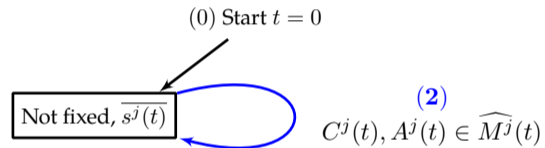
MCTopM algorithm

(0) Start $t = 0$

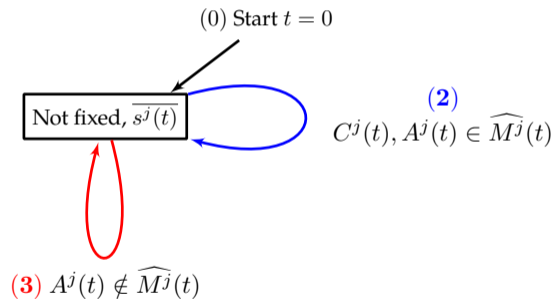


Not fixed, $\overline{s^j(t)}$

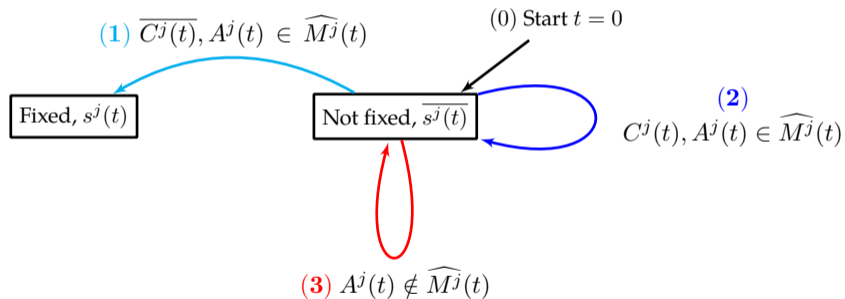
MCTopM algorithm



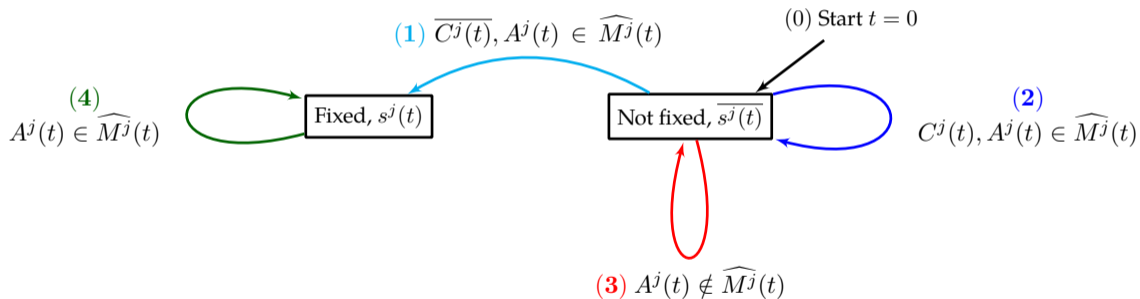
MCTopM algorithm



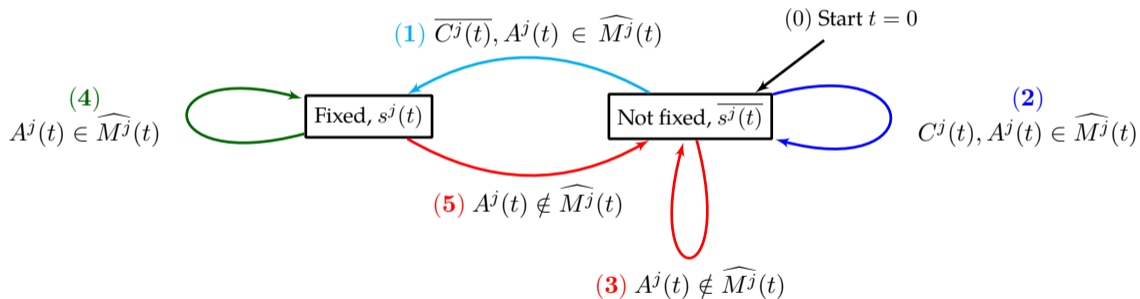
MCTopM algorithm



MCTopM algorithm



MCTopM algorithm



Regret upper bound

- ① Theorems,
- ② Remarks,
- ③ Idea of the proof.

Regret upper bound for MCTopM

Theorem 3

[Besson & Kaufmann, 2017]

One term is controlled by the two others:

$$\sum_{k \in M\text{-best}} (\mu_k - \mu_M^*) (T - \mathbb{E}_\mu[T_k(T)]) \leq (\mu_1^* - \mu_M^*) \left(\sum_{k \in M\text{-worst}} \mathbb{E}_\mu[T_k(T)] + \sum_{k \in M\text{-best}} \mathbb{E}_\mu[C_k(T)] \right)$$

So only need to work on both **sub-optimal selections** and **collisions**.

Regret upper bound for MCTopM

Theorem 3

[Besson & Kaufmann, 2017]

One term is controlled by the two others:

$$\sum_{k \in M\text{-best}} (\mu_k - \mu_M^*) (T - \mathbb{E}_\mu[T_k(T)]) \leq (\mu_1^* - \mu_M^*) \left(\sum_{k \in M\text{-worst}} \mathbb{E}_\mu[T_k(T)] + \sum_{k \in M\text{-best}} \mathbb{E}_\mu[C_k(T)] \right)$$

So only need to work on both sub-optimal selections and collisions.

Theorem 4

[Besson & Kaufmann, 2017]

If all M players use MCTopM with kl-UCB:

$$\forall \boldsymbol{\mu}, \exists G_{M,\boldsymbol{\mu}}, \quad R_T(\boldsymbol{\mu}, M, \rho) \leq G_{M,\boldsymbol{\mu}} \log(T) + o(\log T).$$

Regret upper bound for MCTopM

How?

Control both terms, both are logarithmic at finite horizon:

- Suboptimal selections with the “classical analysis” on kl-UCB indexes.
- Collisions are also controlled with inequalities on the kl-UCB indexes...

Regret upper bound for MCTopM

How?

Control both terms, both are logarithmic at finite horizon:

- Suboptimal selections with the “classical analysis” on kl-UCB indexes.
- Collisions are also controlled with inequalities on the kl-UCB indexes...

Remarks

- The constant $G_{M,\mu}$ scales as M^3 , way better than RhoRand’s constant scaling as $M^2 \binom{2M-1}{M}$,
- We also *minimize the number of channel switching*: interesting as changing arm costs energy in radio systems,

Regret upper bound for MCTopM

How?

Control both terms, both are logarithmic at finite horizon:

- Suboptimal selections with the “classical analysis” on kl-UCB indexes.
- Collisions are also controlled with inequalities on the kl-UCB indexes...

Remarks

- The constant $G_{M,\mu}$ scales as M^3 , way better than RhoRand’s constant scaling as $M^2 \binom{2M-1}{M}$,
- We also *minimize the number of channel switching*: interesting as changing arm costs energy in radio systems,
- For the suboptimal selections, we *match our lower bound* !

Sketch of the proof

- ① Bound the expected number of collisions by M times the number of collisions for non-sitted players,

Sketch of the proof

- ① Bound the expected number of collisions by M times the number of collisions for non-sitted players,
- ② Bound the expected number of **transitions of type (3) and (5)**, by $\mathcal{O}(\log T)$ using the kl-UCB indexes and the forced choice of the algorithm:
$$g_k^j(t-1) \leq g_{k'}^j(t-1), \quad \text{and} \quad g_k^j(t) > g_{k'}^j(t) \quad \text{when switching from } k' \text{ to } k,$$

Sketch of the proof

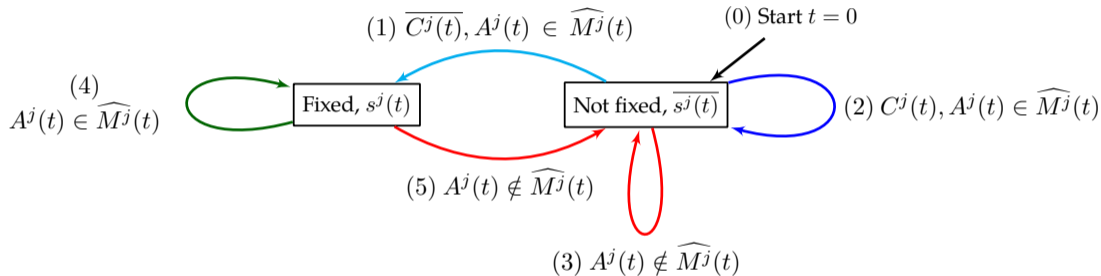
- ① Bound the expected number of collisions by M times the number of collisions for non-sitted players,
- ② Bound the expected number of transitions of type (3) and (5), by $\mathcal{O}(\log T)$ using the kl-UCB indexes and the forced choice of the algorithm:
 $g_k^j(t-1) \leq g_{k'}^j(t-1)$, and $g_k^j(t) > g_{k'}^j(t)$ when switching from k' to k ,
- ③ Bound the expected length of a sequence in the non-sitted state by a constant,

Sketch of the proof

- ① Bound the expected number of collisions by M times the number of collisions for non-sitted players,
- ② Bound the expected number of transitions of type (3) and (5), by $\mathcal{O}(\log T)$ using the kl-UCB indexes and the forced choice of the algorithm:
 $g_k^j(t-1) \leq g_{k'}^j(t-1)$, and $g_k^j(t) > g_{k'}^j(t)$ when switching from k' to k ,
- ③ Bound the expected length of a sequence in the non-sitted state by a constant,
- ④ So most of the times ($\mathcal{O}(T - \log T)$), players are sitted, and no collision happens when they are all sitted!

↪ See our paper for details!

Sketch of the proof



– Time in sitted state is $\mathcal{O}(\log T)$, and collisions are $\leq M$ collisions in sitted state

$\implies \mathcal{O}(\log T)$ collisions.

– Suboptimal selections is $\mathcal{O}(\log T)$ also as $A^j(t + 1)$ is always selected in $\widehat{M^j(t)}$ which is M -best at least $\mathcal{O}(T - \log T)$ (in average).

Experimental results

Experiments on Bernoulli problems $\mu \in [0, 1]^K$.

- ① Lower bound on regret,
- ② Illustration of regret for a single problem and $M = K$,
- ③ Regret for uniformly sampled problems and $M < K$,
- ④ Logarithmic number of collisions,
- ⑤ Logarithmic number of arm switches,
- ⑥ Fairness?

Illustration of the Lower Bound on regret

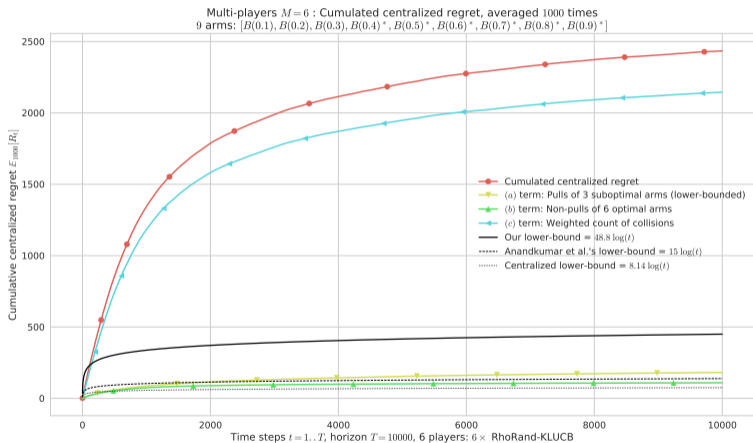


Figure 2: Any such lower bound is **very asymptotic**, usually not satisfied for small horizons. We can see the importance of the collisions!

Constant regret if $M = K$

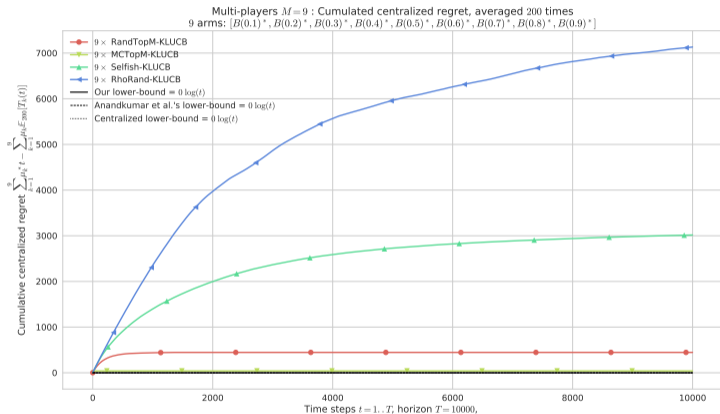


Figure 3: Regret, $M = 9$ players, $K = 9$ arms, horizon $T = 10000$, 200 repetitions. Only **RandTopM** and **MCTopM** achieve constant regret in this saturated case (proved).

Illustration of regret of different algorithms

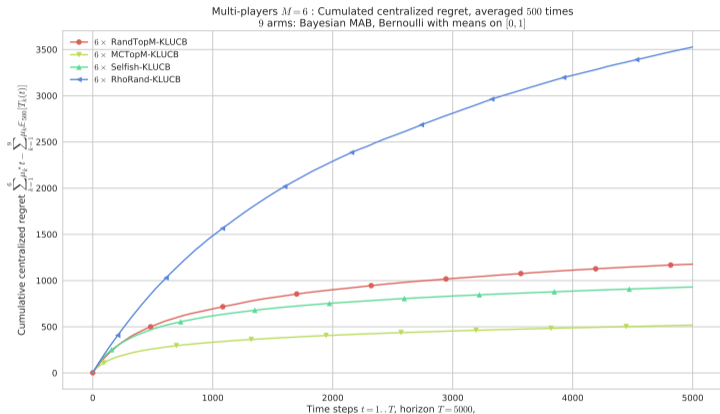


Figure 4: Regret, $M = 6$ players, $K = 9$ arms, horizon $T = 5000$, against 500 problems μ uniformly sampled in $[0, 1]^K$. Conclusion : RhoRand < RandTopM < Selfish < MCTopM in most cases.

Logarithmic number of collisions

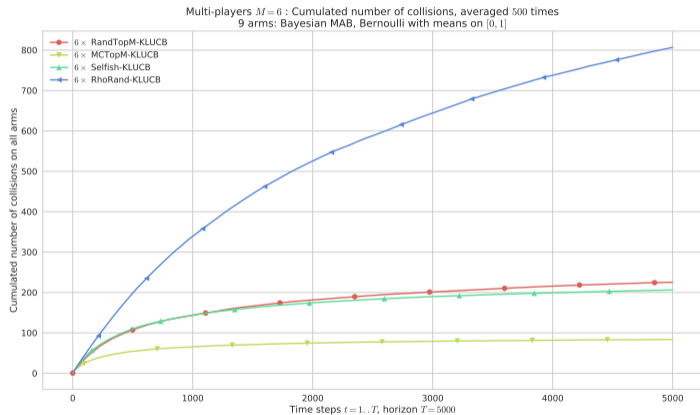


Figure 5: Cumulated number of collisions. Also $\text{RhoRand} < \text{RandTopM} < \text{Selfish} < \text{MCTopM}$.

Logarithmic number of arm switches

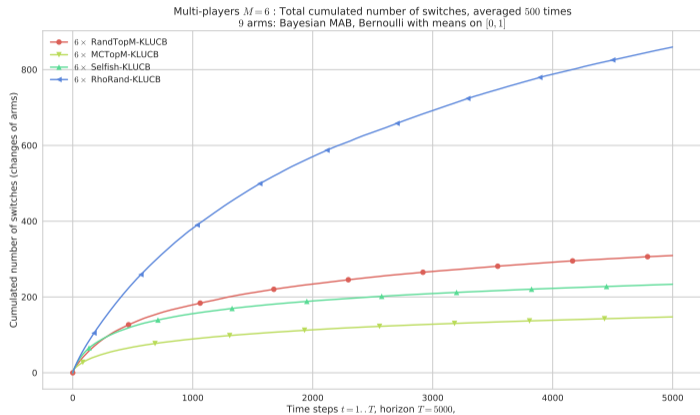


Figure 6: Cumulated number of arm switches. Again $\text{RhoRand} < \text{RandTopM} < \text{Selfish} < \text{MCTopM}$, but no guarantee for RhoRand .

An heuristic, Selfish

For the harder feedback model, without sensing.

- ① An heuristic,
- ② Problems with Selfish,
- ③ Illustration of failure cases.

Selfish heuristic I

Selfish decentralized approach = device don't use sensing:

Selfish

Use UCB_1 (or $kl\text{-}UCB$) indexes on the (non *i.i.d.*) rewards $r^j(t)$ and not on the sensing $Y_{A^j(t)}(t)$.

Reference: [Bonnefoi & Besson et al, 2017]

Works fine...

- More suited to model IoT networks,
- Use less information, and don't know the value of M : we expect Selfish to not have stronger guarantees.
- It works fine in practice!

Selfish heuristic II

But why would it work?

- Sensing feedback were *i.i.d.*, so using UCB_1 to learn the μ_k makes sense,
- But collisions make the rewards not *i.i.d.* !
- Adversarial algorithms should be more appropriate here,
- But empirically, Selfish works much better with kl-UCB than, *e.g.*, Exp3...

Works fine...

- Except... when it fails drastically! ☹️
- In small problems with M and $K = 2$ or 3 , we found small probability of failures (*i.e.*, linear regret), and this prevents from having a generic upper bound on regret for Selfish.

Illustration of failing cases for Selfish

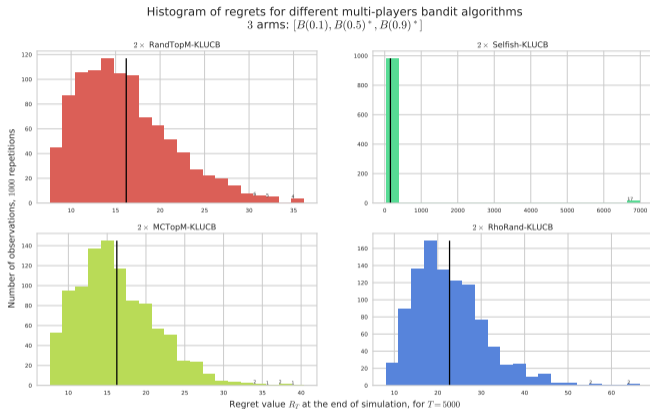


Figure 7: Regret for $M = 2, K = 3, T = 5000$, 1000 repetitions and $\mu = [0.1, 0.5, 0.9]$. Axis x is for regret (different scale for each), and **Selfish** have a small probability of failure (17/1000 cases of $R_T \gg \log T$). The regret for the three other algorithms is very small for this “easy” problem.

Sum-up

Wait, what was the problem ?

- MAB algorithms have guarantees for *i.i.d. settings*,
- But here the collisions cancel the *i.i.d.* hypothesis...
- Not easy to obtain guarantees in this mixed setting (“game theoretic” collisions).

Sum-up

Wait, what was the problem ?

- MAB algorithms have guarantees for *i.i.d. settings*,
- But here the collisions cancel the *i.i.d.* hypothesis...
- Not easy to obtain guarantees in this mixed setting (“game theoretic” collisions).

Theoretical results

- With sensing (“OSA”), we obtained strong results: a lower bound, and an order-optimal algorithm,
- But without sensing (“IoT”), it is harder... our heuristic Selfish usually works but can fail!

Future work

Conclude the Multi-Player OSA analysis

- Remove hypothesis that objects know M ,
- Allow arrival/departure of objects,
- Non-stationarity of background traffic etc.

Future work

Conclude the Multi-Player OSA analysis

- Remove hypothesis that objects know M ,
- Allow arrival/departure of objects,
- Non-stationarity of background traffic etc.

Extend to more objects $M > K$

Extend the theoretical analysis to the large-scale IoT model, first with sensing (*e.g.*, models ZigBee networks), then without sensing (*e.g.*, LoRaWAN networks).

Conclusion I

- In a wireless network with an *i.i.d.* background traffic in K channels,
- M devices can use both sensing and acknowledgement feedback, to learn the most free channels and to find orthogonal configurations.

We showed 😊

- Decentralized bandit algorithms can solve this problem,
- We have a lower bound for any decentralized algorithm,
- And we proposed an order-optimal algorithm, based on kl-UCB and an improved Musical Chair scheme, MCTopM

Conclusion II

But more work is still needed... 😊

- **Theoretical guarantees** are still missing for the “IoT” model (without sensing), and can be improved (slightly) for the “OSA” model (with sensing).
- Maybe study **other emission models**...
- Implement and test this on **real-world radio devices**
↪ demo (in progress) for the ICT 2018 conference!

Thanks! 😊

Any question ?