



# Présentation

## Directions de recherche (notamment avec Rémi)

- Quelques directions auxquelles on a pensé
- Expliquées rapidement ici (pas encore bcp boulot dessus)
- Dites-nous ce que vous en pensez :
- quelle(s) direction(s) favoriser
- d'autres idées ?
- dans quel ordre ?

Merci d'avance !

## Version poster de notre article CrownCom'17

- Déjà fait !
- Disponible en ligne → [lbo.k.vu/JdD2017](http://lbo.k.vu/JdD2017) (LaTeX et PDF) (Bitbucket non-officiel de l'équipe SCEE)
- Présenté bientôt par Rémi à la Journée des Doctorants (03 juillet 2017) @ Rennes (que j'organise)
- (?) Présenté par Lilian au Workshop on Decentralized Machine Learning, Optimization and Privacy (Sep 11-12, 2017) @ Lille

## Version longue de notre article CrownCom'17 → journal !

- Ajout explication algorithmes adversariaux, notamment Exp3 (presque OK)
- Ajout simulation comparant Exp3 et un SoftMax à UCB, Thompson etc (OK)
- Preuve détaillée/rigoureuse de l'utilisation des multiplicateurs de Lagrange pour le problème d'optimisation considéré (OK)
- Simulation si les objets dynamiques et statiques ont différents taux d'activation  $p$  (ex. plus actif que ),  $p_1, p_2$
- Simulation si chaque taux d'activation est différent  $p_k, k \in [S + D]$  (plus compliqué) application réaliste?

## Version longue de notre article CrownCom'17 → journal !

- Preuve ou justification que UCB / Thompson Sampling marche dans ce cadre non-i.i.d. : aucune idée
- Et justification face à l'activation “discrète” : l'apprentissage de chaque objet a lieu “une fois de temps en temps” (activation  $\sim \text{Bern}(p_k)$ ), à des temps différents
- $\hookrightarrow$  “sparse” learning ? est-ce un truc connu ? (@Emilie?)

## CrownCom'17 → journal !

### Essayer avec plus de stations de base ?

- But : ajouter une dimension spatiale à l'apprentissage (mais les objets n'ont pas besoin de le savoir)
- Intuition : on obtiendra des regroupements automatiques, par “clusters” autour de chaque BTS
- Cf. discussion plus bas sur le “capture effect” qu'on pourrait aussi considérer (un autre modèle de collision)
- ça fera peut-être trop pour la version journal (on ne peut pas tripler la taille de l'article !)

↪ Généraliser le regret “au cas IoT” ?

- $K$  canaux,  $M \gg K$  objets, avec des taux d'activation très faibles (activation Bernoulli, moyenne  $p$ )
- Collisions : perte de récompense si des objets utilisent le même canal en même temps (modèle classique)
- $R_T$  classique est une  $\mathbb{E}$  sur les récompenses ( $\mathbb{E}_\mu$ ), peut-être doit-on considérer un autre regret (“sparse regret” ?)  $\tilde{R}_T$  qui fasse aussi  $\mathbb{E}$  sur les activations  $\mathbb{E}_{\mu,p}$
- $\implies$  Bornes inf / sup ?!
- (je n'ai pas essayé pour l'instant)

# Idées pour améliorer des algorithmes de bandit

- ① Être plus robuste face à des environnements “lentement dynamique” (de temps en temps, les distributions des bras changent vraiment) :
  - avec une fenêtre glissante
  - autorise à se remettre en cause “après avoir convergé”
  - ex : échelle de temps d’une semaine pour des objets communicants
- ② (pas encore d’autres)



## Être plus robuste : “fenêtre glissante”

- Avec une fenêtre glissante de taille  $\tau$  “moyenne”, on garde une petite moyenne empirique  $\hat{\mu}_k(t - \tau \dots \tau)$  (ou variance empirique)
- Si la petite moyenne devient trop éloignée de la moyenne complète ( $|\hat{\mu}_k(t - \tau \dots t) - \hat{\mu}_k(0 \dots \tau)| \geq \varepsilon$ ), on remet les statistiques internes de l’algorithmes avec la petite moyenne
- Peut être malin, mais dur de savoir comment choisir  $\tau$  et  $\varepsilon$  : dépendent de la fréquence de changement (inconnue)...

## Être plus robuste : “fenêtre glissante”

Si la distribution des bras est connue, on peut lier les deux paramètres à un seul paramètre de “confiance” (@Rémi ?)

- Facile à coder pour un algorithme basé sur  $\hat{\mu}_k(t)$  et  $N_k(t)$  nb sélections du bras  $k$  (e.g., UCB, kl-UCB)
- Déjà testé :  $\rightarrow$  Policies.SlidingWindowsRestart
- Marche mal empiriquement... (pour les valeurs essayées, et impossible de savoir lesquelles peuvent marcher)
- Plus dur pour des algorithmes Bayésiens (demande l'historique de taille  $\tau$  pour réinitialiser les posteriors avec l'historique partiel : consommation de la mémoire),
- et (je pense) impossible pour des algorithmes génériques...



## Autre idée : on/off de stations de base

Apprentissage centralisé, doit couvrir tous les utilisateurs avec la plus faible consommation électrique possible...

- Suite Navik (& un peu Rémi).
- Approche combinatoire, bras = configuration des  $B$  stations de base : taille espace d'état  $\propto 2^B$
- Passe pas à l'échelle avec un réseau "dense" ( $B \geq 10$ )
- Idée pour une approche moins coûteuse ?  $\rightarrow$  "Bandit combinatoire" ? (@Emilie?)
- pas clair de savoir si on peut faire plus que ce qui a été fait (pas forcément intéressant)
- cadre applicatif réaliste ? (je pense que oui)

## Autre idée : “transfert learning” avec des algorithmes connaissant $T$

- Si l’horizon  $T$  est connue, comme c’est le cas dans les utilisations de “transfer learning” (TL), ex. par Navik
- Alors autant utiliser des algorithmes qui exploitent  $T$  pour être plus efficace (ex kl-UCB+ ou UCB+)
- ApproximatedFiniteGHIndex semble être excellent dans ce cas (par Tor Lattimore, COLT 2016)
- Exemple :  $T = 1\text{h}$  ou  $T = 1\text{ jour}$
- mais on a pas de cadre applicatif novateur ou vraiment intéressant... Idée ?!  
(@Christophe?)

## Suite projets élèves

On doit continuer et finir les projets des élèves

Rappels : 2 projets longs (Théo & Clément, Jihane & Salma), 2 projets courts... Presque rien des courts, mais les longs ont donné :

- émetteur/récepteur LoRa (modulation/démodulation) réaliste
- générateur trafic “interférant” aléatoire (configurable )
- station de base qui écoute et détecte sur  $K$  canaux
- $\implies$  produire un beau démonstrateur avec GNU Radio pour “MAB algorithms for real-world LoRa IoT networks”
- on peut en faire une démo à présenter (dans une conf’ ?)
- et un article (avec le nom des élèves mais fait par nous)

## Suite projets élèves

- Rémi a déjà beaucoup avancé pour la partie LoRa (Jihane et Salma on fait du bon boulot)
- On fera ensemble l'objet intelligent qui s'insère dans le réseau
- On va pouvoir connecter facilement ma “grosse” collection d'algorithmes ([banditslilian.gforge.inria.fr/docs/Policies.html](http://banditslilian.gforge.inria.fr/docs/Policies.html))
- On va rédiger un mini-article présentant la démo, avec Rémi, Christophe et Lilian (==et Emilie ?==), et les 4 élèves (mais on n'espère pas de l'aide de leur part)

## Suite projets élèves

- Démo : j'aimerais une interface graphique sexy (comme ce qu'a fait Quentin, lançable sans passer par GNU Radio)
- Contrairement à la démo ICC de Christophe, avec plusieurs algorithmes en parallèle, on veut un seul algorithme (un seul objet communiquant)
- Mais un large choix d'algorithme et de paramètre, qu'on puisse changer en cours d'expériences (en réinitialisant l'algo)
- Ex: pour vérifier de façon interactive que  $UCB_\alpha$  converge plus pour de petits  $\alpha$ , que TS est plus rapide que kl-UCB etc...



# Aperçu démo de Quentin : jolie interface

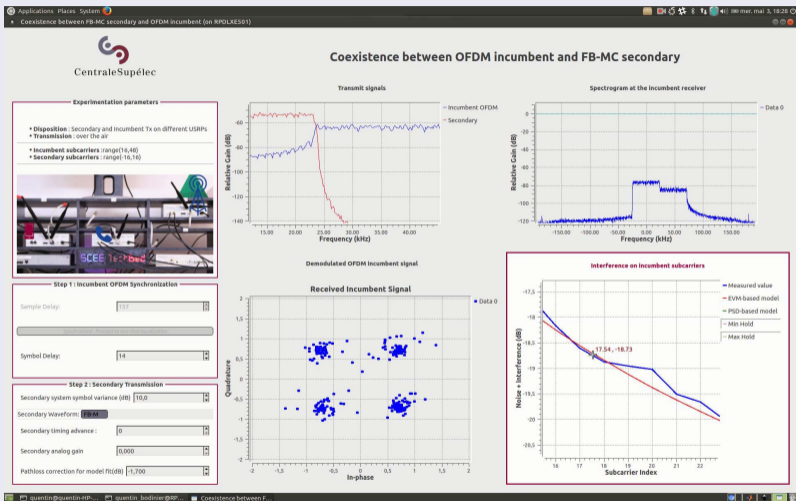


Figure 1: test

## Plusieurs stations de base (BTS ), collision selon le “capture effect”

Un autre modèle de collision, plus complexe mais plus réaliste

- Plus un objet est près de la BTS, plus elle reçoit ses messages avec un SINR fort. La BTS peut décider de répondre seulement si  $\text{SINR} \geq \theta$  (un certain seuil), ou juste à celui qui a le plus fort SINR (“near-or-far” effect)
- $\text{SNIR} = \propto P_R / (N + \sum_{\text{objet}} P_{R_i})$
- A distance  $d$ , l’interférence est souvent modélisée  $\sim d^{-\alpha}$  (pour  $\alpha \in [2, 5]$  selon le milieu ambiant)
- Modèle déjà implémenté (closerUserGetsReward) pour une distance (comme si  $\text{SNIR} \propto d$ , irréaliste mais simple)

## Plusieurs stations de base (BTS ), collision selon le “capture effect”

- A faire : implémenter un modèle de collision selon le SNIR
- Mettre  $\geq 2$  BTS sur une carte 2D, répartir des objets sur la carte, aléatoirement (comme Navik)
- Montrer que les objets convergent vers des configurations orthogonales au sein de chaque cellule, et aux frontières des cellules aussi (cf. dessin au tableau)
- (cf. répartition des fréquences pour la télé en “large cells network”)
- Intuition : ça va marcher automatiquement sans trop de soucis
- Encore le même problème : prouver des garanties théoriques sur cette réussite empirique ne sera pas facile...

Fin

==D'autres idées ?==

==Discussion==

Merci