On the relevance of bandit algorithms in digital world

Raphaël Feraud

université PARIS-SACLAY

NNT : 20XXSACLXXXX

Habilitation à diriger les recherches

orange Orange Innovation

université PARIS-SACLAY

On the relevance of bandit algorithms in digital world

Habilitation à Diriger les Recherches de l'Université Paris-Saclay
préparée à Orange Innovation

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Orsay, le 01/02/2023, par

RAPHAËL FÉRAUD

Composition du Jury :

Michèle Sebag Directrice de Recherche, LRI Paris Saclay	Présidente
Aurélien Garivier Professeur, ENS Lyon	Rapporteur
Emilie Kaufmann Chargée de recherche, CNRS INRIA	Rapporteur
Liva Ralaivola Directeur de Recherche, Critéo	Rapporteur
Florence D'Alché-Buc Professeure, Télécom Paris	Examinatrice
Vianney Perchet Professeur, ENSAE	Examinateur

Acknowledgements

Je voudrais tout d'abord remercier Michèle Sebag, qui, voilà presque quinze ans, lors d'une réunion de travail dans le cadre d'un Contrat de Recherche Externe, a prononcé les mots mystérieux *d'algorithmes de bandits*. Ces mots ont éveillé la curiosité du chef de projet, qui était à l'époque plongé dans des problèmes de tuyauterie que l'on appellera ensuite *Big Data*. Ces mots ont ensuite fait leur chemin et en quelques années ont reveillé le chercheur qui sommeillait en moi. Je voudrais ensuite remercier Emilie Kaufmann, Liva Ralaivola et Aurélien Garivier, d'avoir accepté d'être les rapporteurs de cette thèse d'Habilitation à Diriger les Recherches. Leurs conseils précieux et précis ont permis d'améliorer la qualité de ce manuscrit. Je voudrais remercier Florence D'Alché-Buc et Vianney Perchet de m'avoir fait l'honneur d'accepter de faire partie de mon jury. Je remercie Odalric-Ambryn Maillard pour sa gentillesse et pour son implication dans les thèses de Robin Allesiardo et Réda Alami. Excellent mathématicien, il a beaucoup contribué à la réussite de ces thèses. Je remercie Patrick Maillé, et Nadège Favre avec qui j'ai découvert l'Internet des Objets, un nouveau terrain de jeux pour les algorithmes de bandits multi-joueurs. J'ai eu ensuite le plaisir d'encadrer avec eux la thèse d'Hiba Dakdouk. Je remercie aussi Patricia Stolf, Jean-Marc Pierson et David Delande avec qui nous diffusons les algorithmes de bandits dans le Cloud. Je remercie aussi Tanguy Urvoy avec qui j'ai fait mes premiers papiers sur les bandits. Je remercie Djallel Bouneffouf et Romain Laroche, qui partis outre-Altantique, ont continué de travailler avec moi sur différents problèmes de bandits et d'apprentissage par reinforcement. Enfin je remercie toute cette petite communauté de bandits pas vraiment manchots que j'ai parfois le plaisir de croiser dans les conférences.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Twenty five years ago, in introduction of my PhD thesis [Féraud, 1997], I wrote two science fiction short stories. These two stories feature a vocal assistant that intelligently interacts with a person, in very different contexts. In *"2048, un matin sur la terre, la grande soeur"*, *Big Sister is watching you* in a post-democratic world dominated by big companies. In *"2048, un matin sur la terre, Doma"*, Doma is a considerate butler in a world of knowledge and democracy. In both cases, the world is totally connected through a *digital world*, which notably allows teleworking and virtual reality. In the vocal assistant, *Artificial Intelligence* is used for performing the same functionalities: image understanding, natural language processing, vocal synthesis, question answering, diary management... The point was not to perform futurology. 2048 was just a reference to Georges Orwell. The point was to say that we have in hand a class of algorithms, called neural networks, that can be used to build machines that seem to be intelligent, but which are simply things, tools that we can control to choose our future.

What has changed today?

The *Deep Learning* revolution announced in the seminal paper of Yoshua Bengio [Bengio, 2009] has occured. The use of GPU [Raina et al., 2009] allowed massive parallel computing and hence performing structural risk minimization [Vapnik, 1995] on deep architectures. The use of ReLu functions [Glorot et al., 2011] has fixed the issue of vanishing gradients through the layers. Efficient optimization algorithms [Kingma and Ba, 2015] and regularization techniques [Srivastava et al., 2014] for deep neural networks has also been proposed. All these innovations have allowed building deep LeNet [LeCun et al., 1989] for handling problems that in the end of the last century were reputed to be difficult. In classification, breakthrough results have stunned the image processing community [Krizhevsky et al., 2012, Vinyals et al., 2015]. In speech recognition impressive results have been obtained [Graves et al., 2013]. Then, the use of word embedding techniques [Tomas et al., 2013] allow breakthrough results in Natural Language Processing [Devlin et al., 2019]. But the most significant breakthroughs have been done in reinforcement learning,

and more precisely in video games [Mnih et al., 2013] and then in the game of Go [Silver et al., 2017].

Despite these exciting results in *Deep Learning*, this does not change anything in substance. Deep Neural Networks are just things, algorithms that we can control. In the seventies, my pocket calculator was better than me in mental calculation. In the nineties, Kasparov was beat at chess game by the super computer of IBM, Deep Blue. Now, a human will never win again a computer in Go game. Is that all ?

In parallel of the *deep learning* revolution, another significant change has occured: *Big Data*. This revolution is also due to the increasing of computer power, but this one is due to massively distributed CPU architectures. The map-reduce framework [Dean and Ghemawat, 2008] changes the paradigm of standard databases. The data are distributed over servers in a way that all data concerning a small subset of keys are stored together close to a CPU that can process them. This allows to store and process very large amount of data. This new technology have emerged for handling the need of processing logs. Twenty five years ago, the largest log databases were the logs of phone calls. Now, the logs of Internet's use exceeded by six orders of magnitude the logs of phone calls. In France twenty five years ago, to access the logs of phone calls, an authorization from french authorities was necessary. Today, all our acts in the digital world are logged, stored and often duplicated by a lot of states and companies.

The combination of the *Deep Learning* and *Big Data* revolutions allows to infer valuable information that can be used for different purposes. For instance, the kind of tools used in the movie minority report, where predictive models are used to avoid crimes, is no longer science fiction. There is no doubt, that with this kind of *things*, a totalitarian state, such was described in 1984, disposes of the tool for controlling the population. However, these new technologies render invaluable services. We have access to the greatest library that any human of the past centuries can just imagine. We have access to a lot of communication services that make possible teleworking, exchanging information with our community anywhere in the world. We have access to entertainment services that can recommend us movies, musics, books... The police has new tools to investigate and stops criminals. Indeed, predictive tools are used by authorities to protect us against terrorists. Unfortunately, autonomous war machines have been built and used. But again, that is called *artificial intelligence* is just an *algorithm* that can be used for the best or the worst. Things are still under control.

In our *digital world*, the true risk is simply the bug. For instance on June 2, 2021, a usual maintenance operation on a software of Orange has caused a bug, which made emergency calls impossible. On 19 November, 2021, a bug in a server prevented Tesla cars from starting. Twenty five years ago, the great fear was the year 2000 bug. Today, it is a daily risk. Moreover, with the incredibly fast deployments of algorithms by a lot of actors in all the *digital world*, a new kind of bugs could occur. Algorithms, which are more and more autonomous, which are more and more omnipresent, are going to more and more interact themself. In *smart grid*, we are working on automation and optimization of electric networks. These valuable researches aim to take into account sustainable sources of energy, which often depend on random events such as weather, and to increase the efficiency of electric networks by choosing at each time the best assignment between the production and consumption places. In *smart city*, the aim

is to optimize all the services of the city (transport, circulation, water distribution, power station, waste management, schools, hospitals...) by massively using *Internet of Things* that collect information. In *autonomous cars*, we are going to automate the driving of most vehicles. This will change our relationship with cars and will allow increasing security, decreasing energy consumption, efficiently sharing cars... In telecommunication network, we are going to equip all devices of algorithms in order to maximize successful communications and minimize energy consumption, while reducing the latency. Actually, all the fields of human life is going to be impacted by the massive usage of algorithms. Moreover, these fields are not independent. It seems reasonable that an autonomous car asks for a green light if it detects nothing around. But this induces a dependence between at least three kind of algorithms: those that help to connect to the telecommunication network, those that drive the vehicles, and those that optimize the road traffic. This dependence between algorithms, which can have been independently designed, raises two problems. The first one is obviously the sub-optimally of the decisions that are taken independently. The second one is the fragility to bugs or attacks that can occur on each device. The *side effect* of a bug, which is well known in computer sciences, could take an unknown magnitude to date.

In 2018 in Europe, the clocks of the most current electric devices were up to 6 minutes late between January and March. People were informed in the beginning of March. How many trains have been missed ? How many people were late ? The reason of this late seems incredible without more information: it was due to a political conflict between Serbia and Kosovo. Indeed, while Serbia produces electricity, to supply Europe the electricity goes through Kosovo. In January 2018, Kosovo decided to lower electric supply to Europe in order to put pressure on it for handling their different with Serbia. As a consequence the frequency of electric current decreased from 50 Hz to 49.999 Hz in whole Europe. Then, the clocks that use this frequency to synchronize themself fell further behind. An human and adversarial decision was the cause of the late, even if this consequence may not have been anticipated by the actors. The problem of late clocks warns us that we need a close control of algorithms. In a fully connected world, an event, a bug, an attack that occurs faraway, that seems totally disconnected from our everyday life, can have a significant impact on it. In the *digital world*, keeping things under control is going to be a challenging task. This necessitates to guarantee as much as possible the overall functioning of learning systems, massively decentralized, pursuing possibly different goals, and being able to interact with each other in a collaborative or a competitive manner in a changing environment. My research is in this direction.

1.2 My research: an illustrative example

In smart city, most of individuals devices and collective equipment are connected (figure 1.1). This allows optimizing most of services proposed by the city. For instance, in order to insure that autonomous cars are available where and when they are needed, the number of autonomous cars in each parking has to be optimized. For performing this optimization, one needs to decide for each parked autonomous car if it stays or if it changes of parking, while

running autonomous cars that have finished their mission have to choose a parking. Taking into account that the distances between parking are not the same, this optimization problem should be reduced to the salesman problem and hence, one can conjecture that it is NP-hard. Moreover, it depends on some random events, such as "a user would like an autonomous car for his particular and unknown purpose". Worst, the *parking of autonomous cars* interferes with another NP-hard optimization problem: *optimization of traffic light* (routing problem). Indeed, the user, who would like an autonomous car (figure 1.1), needs to cross the street. If he has a green light, he will take a car, and then the autonomous car that is looking for a parking can park. If he has not a green light, the autonomous car that is looking for a parking cannot park. Moreover, the *parking of autonomous cars* is strongly linked with another intractable problem: *smart grid optimization* consists in deciding at each time slot if each connected device (including parked vehicles) has to charge or discharge its battery. If a parked vehicle decides to leave, it cannot charge or discharge. In smart cities, *advertising optimization* notably consists in displaying information in panels. This can divert pedestrians from their initial goal to enter in a shop or to stop for looking at a panel. Hence *advertising optimization* can interfere with *optimization of traffic light*. Finally, for sending or receiving messages most of devices use radio communication. *Optimizing radio communication* is a resources allocation problem that assigns at each time slot a channel to each transmission. When too much devices would like to communicate, congestion occurs and hence all the previous optimization problems are impacted.

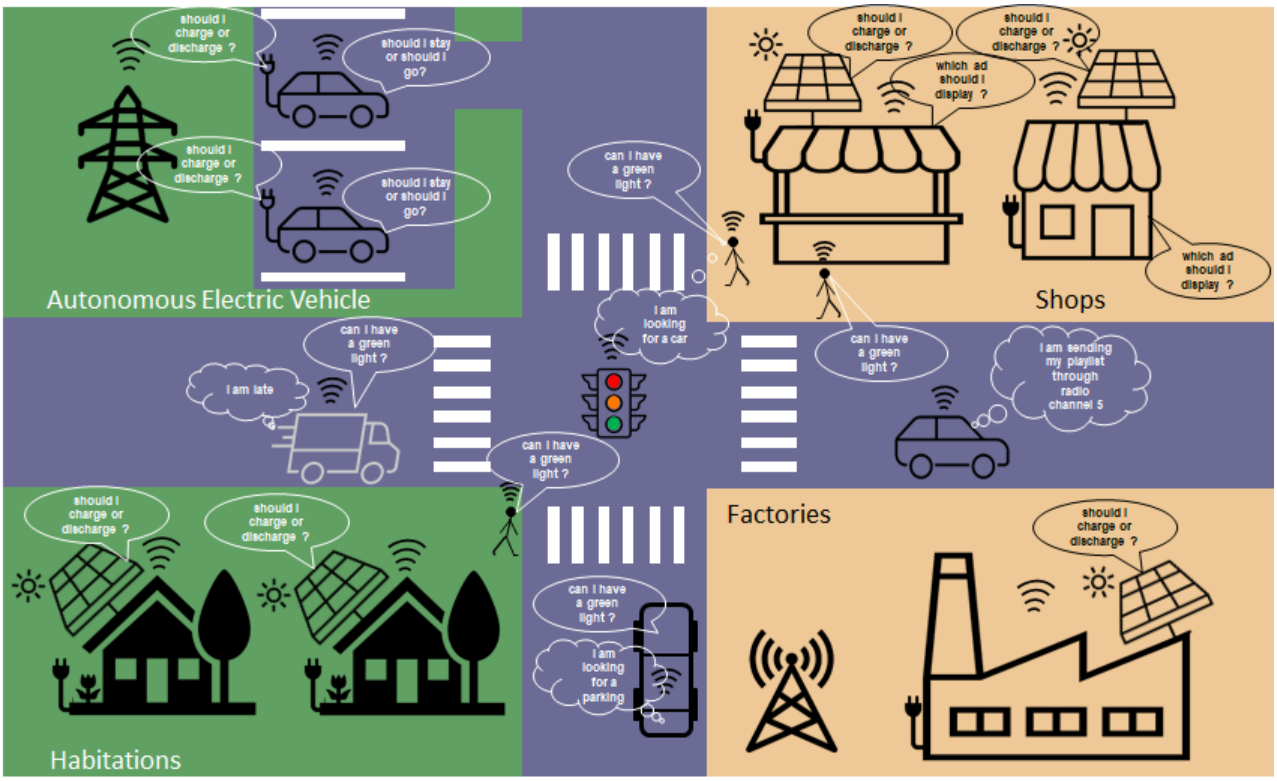


Figure 1.1: In order to optimize smart city, most of individual devices, and collective equipment are connected: autonomous cars, electric vehicle charging stations, photovoltaic panels, traffic light, advertising panels, parking, connected devices...

In smart cities, several complex optimization problems (parking of autonomous cars, smart grid, traffic light, advertising panels, radio communication...) are linked. These intertwined optimization problems raise a lot of interesting questions.

When addressing NP-hard problems, one has to work at small scale for computing an optimal policy, or to search a reasonably good and tractable policy. How defining such policies ? As the optimization problems are linked, can we find good compromises between antagonistic objectives (for instance parking cars consume energy) ? The connected devices are worn by humans or are designed to render services to humans. Hence, optimizing the behavior of connected devices impacts the everyday life of humans and raises a classical question: how can we find good compromises between individual and collective interests ? The smart city is a work in progress. One cannot expect that smart city is a stationary environment. Technically this means that one cannot consider that the random variables are sampled from a stationary distribution. Can we learn fast enough (before the environment changes) a target policy ? In smart city, a lot of connected devices monitor the actions of each citizen. Can we insure privacy in such connected smart cities ? In smart city, several intertwined optimization problems impact the everyday life of citizens. Could we insure that no bad event occurs (no car where they are needed, no electric supply for factories, traffic jam, network congestion...) ?

In digital world, we would like to guarantee as much as possible the overall functioning of learning agents, sharing the same limited resources, pursuing different objectives, interacting with other agents in a collaborative or competitive manner in the same evolving environment.

1.3 Contributions

As in the general case some of the intertwined optimization problems are NP-hard, for answering to some of the stated questions, our approach consists in using decentralized algorithms that aim to find optimal policies, when it is possible, or else reasonably good policies. For designing such algorithms, in this manuscript we study a particular case of a very general framework called *reinforcement learning*: an agent interacts with an unknown environment by choosing actions and observing their outcomes. In *reinforcement learning* framework, the *multi-armed bandit* problem corresponds to the case where a reward is obtained after each played action independently of past played actions. We chose this class of algorithms for two reasons. The bandit algorithms come with theoretical guarantees, on which we can rely to insure in the long run the functioning of digital world. On the practical side, the *multi-armed bandit* problem provides efficient algorithms useful for lot of applications in the digital world.

The second chapter (chapter 2) presents the main bandit problems that are addressed in the manuscript. The *multi-armed bandit* problem focuses on one of the fundamental problem of reinforcement learning: the *exploration-exploitation dilemma*: as the agent only observes the reward of the played action, it has to explore loosely estimated

action for finding the best action to play or to play its best estimated action for gathering rewards. We have endeavored to formally describe the problems from bandits to standard reinforcement learning in order to compare their hardness. We have used the Markov Decision Processes framework for identifying the underlying assumptions on agents, states, and actions in each problem. All these problems share the same objective: minimizing the regret between the optimal policy of the considered problem and the policy played by the agent. We recall the main results on the hardness of these problems: the lower bound of the regret.

The third chapter (chapter 3) is dedicated to our theoretical contribution for handling non-stationary bandits. This line of work strongly intersects with the thesis works of Robin Allesiardo and Réda Alami, whom I have co-supervised with their academic supervisors Michèle Sebag and Odalric Ambryn Maillard. This academic work has also been done as a part of the ANR project Badass lead by Odalric Ambryn Maillard. Handling bandit algorithms in evolving environment is significant because it corresponds to a requirement in the digital world, which evolves all the times. From the theoretical point of view, the challenging aspect is that most of the statistical tools we dispose are based on the stationary assumption. Basically, for handling non-stationary there are two opposites directions. The first one is to be specific by characterising as much as possible the kind of non-stationarity we would like to handle: piece-wise processes, known trend processes, slowly evolving processes... The second one is to be as generic as possible for the cases where it is not possible to assume some knowledge about the observed processes: unknown processes, adversarial processes...

In the first contribution (section 3.1), we chose to enhance, is the works done on selection the best learning expert [Allesiardo and Féraud, 2017]. This study clearly fits in the first direction. In contrast with the well-known selection of the best expert [Cesa-Bianchi and Lugosi, 2006], where no assumption is made on the experts, in this study we take into account that the experts learn and hence increase their performances during time. This problem could be handled by taking into account an increasing parametric trend as we did in [Bouneffouf and Féraud, 2016]. However we can be more specific by using knowledge from the statistical learning theory. The regret of learning experts with respect to the optimal expert can be decomposed in two terms: the first one, called *the estimation error* or the variance, is minimized by the learning of experts, while the second one, called *the approximation error* or the bias, is minimized by using the selection of learning experts within different set of experts. We propose, analyze and discuss several approaches for selecting contextual bandits. We have used the same kind of approaches for the selection of reinforcement learning algorithms [Laroche and Feraud, 2018].

In the second contribution (section 3.2) we chose fits in the second direction: being as generic as possible. In [Allesiardo et al., 2017] we consider a generalization of stationary stochastic, piece-wise stationary, and adversarial bandit problems: at the beginning of the game, the adversary chooses a sequence of distributions instead a sequence of rewards. The starting point of this study is that the basic tools used in bandit algorithms, the concentration inequalities, do not need that the observations are sampled from the same distribution, but simply that they

are sampled independently from distributions. We notably propose an algorithm, *Successive Elimination and Reset* (SER4), which is near optimal in a such generic problem, but also for the stochastic bandit, or the switching bandit problem. We continued this line of works by building a specific algorithm for the switching bandit problem [Alami et al., 2017]. Best of our knowledge, in practice it is the best algorithm for the switching bandit problem. Despite the fact that we have reached the main milestone to analyze this algorithm by stating that the Bayesian Online Change Point Detector is asymptotically optimal [Alami et al., 2020], the analysis of *Memory Bandit* is still an open problem.

The fourth chapter (chapter 4) considers applications of contextual bandits in the digital world. Indeed, in most of applications before an action has to be played, the agent can observe the state of the environment or a context. This is crucial in applications, since the use of the contextual information allows to dramatically increase the performances in terms of *gathered rewards* [1].

The first contribution (section 4.1) relates to the thesis of David Delande that I co-supervise with Patricia Stolf. This contribution is an application of contextual bandit to Cloud elasticity [Delande et al., 2021]. The aim is to optimize allocated resources to an application under the constraint of a maximum latency. For an application hosted in the Cloud platform, given the observed workload, the consumed memory and CPU resources, and the observed latency, the auto-scaler decides if the number of allocated resources has to be increased, decreased, or remained constant. The strong point of this contribution is that the test platform including a Kubernetes platform, a workload injector, and the agent, is provided in a GitHub repository. This allows the community comparing auto-scaling algorithms in a real Cloud platform. Our experiments notably show that the linear contextual bandit (LinUCB [Li et al., 2010a]) dominates Deep-Q Learning ([Mnih et al., 2015]) for allocating resources in the cold-start setting, i.e. when the workload is unknown at the beginning. This preliminary work can be extended in many directions. We already tested NeuralLinUCB ([Zhou et al., 2020]) that improves over LinUCB, but the main direction to improve the performances is to take into account that the application has a structure containing several components, which consume the resources. Hence the optimization should be done at the level of components, which have an unknown (or known) dependency graph. Another direction is to consider the global optimization at the level of the Cloud platform, where applications and their components are players that compete for computing resources. This direction research is strongly linked to chapter 5 (section 5.2).

In the second contribution (section 4.2), we propose to use contextual bandits for optimizing marketing campaign. Indeed for deciding what is the best marketing campaign for a given customer, in addition to the outcomes of contacts, Orange disposes of a lot of information about its customers. These information are related to the usages, the billing, the owned products and services, the commercial and technical interactions... This large amount of data is stored in databases, where they are structured according to a relational data model. Exploiting this kind of data as is a challenging task. Indeed, using standard machine learning approach, the first operation is to transform many

[1] but not in terms of regret, which is not a performance metric suitable for applications.

relational tables in a context vector of finite dimension. Thus aggregates have to be built. We propose a similar approach, which we have tested in Big Data framework [Féraud et al., 2010, Féraud et al., 2008], brute force: as many as possible aggregates are built *in parallel* to select the most relevant for the considered task. We designed a contextual bandit algorithm, called *Bandit Forest* [Féraud et al., 2016], which explores *in parallel* as many cut-off variables which can hold in computer memory. The nice analytic result is that the dependence of the number of samples needed to find with high probability the best *random forest* [Breiman, 2001] is in $O(\log M)$, where M is the number of explored cut-off variables. This work can be extended in many directions. In section 3.1, we have studied how the parameters of *Bandit Forest* can be tuned online. The decision stump, which is the basic block of *Bandit Forest*, can use *Successive Elimination and Reset* proposed in section 3.2 for handling non-stationary rewards. The regret analysis of *Bandit Forest* can be done in place of sample complexity analysis. Notice however that for avoiding the sub-optimal term in $O(1/\Delta^2)$ due to the hardness of best arm identification problem [Mannor and Tsitsiklis, 2004], an additional Lipschitz assumption on the decision stump is necessary. Finally *Bandit Forest* is by nature a parallel algorithm, and its extension to Federated Bandits [Shi and Shen, 2021] or collaborative exploration (see chapter 5) can be considered.

The third contribution (section 4.3) comes from an academic collaboration notably with Djallel Bouneffouf from IBM research. In [Bouneffouf et al., 2021a, Bouneffouf et al., 2021c], we propose a problem, where the agent can ask some questions before choosing an action. For instance, when a customer contacts the after sale service, one needs to identify the occurred problem before proposing a solution. The bot listens the claim of the customer, then knowing it, the bot can ask questions, and finally after observing the answers, the bot proposes its solution. If the problem is solved, the bot gathers a reward. We decompose the problem into two sub-problems.

1. The first one is a combinatorial bandit problem: after observing a feature vector of dimension V, the agent chooses U questions from a fixed but potentially large list of N questions.

2. The second one is a contextual bandit problem: after observing a feature vector of dimension $V+U$, the agent chooses the best action to play.

The same reward is used for both bandit problems. The potential applications of this problem seem significant and large. We notably used it in [Bouneffouf et al., 2021b] for a dialog orchestrator of a vocal assistant.

The last chapter (chapter 5) considers the multi-agent multi-armed bandit problem: a large number of agents interacts with an environment for solving a bandit problem. In comparison to the centralized bandit problem, as the actions are chosen locally, the advantages are the reduction of the communication cost, and the privacy of users.

The first contribution (section 5.1) consider an exploration problem, where the agents collaborate for handling the same bandit problem [Féraud et al., 2019]. This work was carried out within an academic collaboration with Romain Laroche from Microsoft Research. A large number of agents aim to find an approximation of the best arm with

high probability. At each time step each agent has an unknown but constant probability to interact with environment. While solving the exploration problem at the level of the agent is clearly sub-optimal due to the sampling of the agents that leads to sparse interactions, it guarantees privacy since no data transits through the network and obviously no log can be stored. The proposed algorithm, *Decentralized Exploration* consists in a compromise between the fully decentralized approach and the centralized approach. We allow each player to exchange a message per arm, corresponding to its elimination, with other players. For guarantying privacy of the player's choice, the transmitted message is wrong with high probability: an adversary, which listen the messages, cannot infer valuable information about the choices of players with high probability. We notably shows that using an optimal best arm identification subroutine, the penalty in comparison to an optimal centralized algorithm, which does not guarantee privacy, is in $O(N^2 \log 1/\delta)$, which means that the proposed algorithm is still order optimal, where N is the number of players, and δ is the failure probability. We also showed that the communication cost for solving the exploration problem is at most $\lfloor \frac{\log \delta}{\log \eta} \rfloor K - 1$ bits, where $1 - \eta$ is the privacy level, and K is the number of arms. By setting $\delta = 1/T$, this result can be favorably compared with [Hanna et al., 2022], who showed that $3T$ bits of communications is sufficient to conserve the same regret bound using no-regret bandit algorithm. As *Decentralized Exploration* is generic, we also have extended it to non-stationary distributions of reward using *Successive Elimination and Reset* (section 3.2). This work could be extended in many directions, and notably to the case where the agents handle the same contextual bandit problem. When the context does not contain private information, the decision stump (and hence *Bandit Forest*) proposed in section 4.2 can be decentralized using the same approach for guarantying privacy. However, in the case where context contain private information a local model such as a decision tree is not well suited for guarantying privacy: for updating the decision tree, one need to transmit the path in the decision tree, which contains a part of the context. In place, a global model such as a linear model can be used [Dubey and Pentland, 2020]. In the next section, we chose to use the same kind of algorithmic framework in a different problem.

The second and last contribution (section 5.2) studies the massively multi-player bandits: a large number of players interacts with the same environment, but the players compete for accessing the arms ([Dakdouk et al., 2021]). This work was carried out by Hiba Dakdouk, whom I co-supervised with Patrick Maillé from IMT Atlantique and Nadège Varsier from Orange. This problem occurs in wireless network, and notably in Internet of Things where a lot of devices send packets though the same gateway. When at least two devices send a packet using the same communication channel at the same time slot, a collision occurs. This problem seems related to Multi-Player Bandits, but actually it is very different. In contrast to multi-player bandits, where few players ($N \leq K$) send messages at each time step, here a high number of players asynchronously send messages, which radically changes the nature of the optimization problem. Indeed, in the general case the optimization problem becomes intractable. Moreover, sharing the channels with much more players than the number of channels raises the problem of fairness between players. We propose an *explore-then-exploit* approach. We first states two *good* policies. The first one aims to maximize the gathered rewards and is optimal in some cases, while the second one maximizes fairness between players.

In contrast to bandit problem, for computing the target policies, all the parameters have to be estimated in the exploration phase. This makes the proposed problem much more difficult that classic bandit problem. We showed that the proposed policies, running with the estimated parameters, are respectively near optimal in some cases and near fair in all cases with a controlled communication cost in $O(NK \log \frac{NK+N}{\delta})$. The experiments compare them favorably with the state-of-the-art. We have implemented these algorithms on a realistic IoT simulator from Orange. The experiments notably show that our algorithms outperform the standard algorithm used in the protocol in terms of successful communication and energy cost. Moreover, the experiments show that if a team of devices uses our algorithms and another team of devices uses the algorithm designed in the protocol, the first team increases its rewards at the expense of the second. On the theoretical side, the open question is about the interest of *explore and exploit* approaches when all the estimations of mean rewards of arms are needed. Is the regret with respect to an optimal or good policy of an *explore-and-exploit* algorithm can be lower than the stated lower bound for *explore-then-exploit* algorithms ? The first direction to extend this work is applying it to other use cases, which are numerous. For instance, when a fleet of electric vehicles would like to charge their batteries, each vehicle competes with other vehicles for each available time slot, or in smart city, the vehicles compete for green lights with other vehicles and pedestrians. This raises the need to extend this work to more complex decision models such as contextual bandits for taking into account the observation of the environment (chapter 4), and non-stationary rewards for taking into account evolving environment (chapter 3).

Finally, if we back to the direction of research mentioned in section 1.2, i.e. controlling a lot of learning agents acting in the digital world, in this manuscript we contribute to first steps. In chapter 3, we showed that we can learn bandit algorithms in various non-stationary environments. In chapter 4, we showed that in practice the contextual bandit algorithms can handle complex problems, while guarantying that the convergence speed to an optimal policy (or more accurately a target solution) is optimal or near optimal. In chapter 5, we showed that a lot of agents can learn and collaborate to near optimally solve an optimization problem under uncertainty. Moreover we showed that privacy could be insured, as well as fairness. Now, the most difficult raised questions are still open. In future works, for handling intertwined optimization problems, we will consider a bandit problem, where several agents interact in the same environment for handling different tasks, but where some of their actions overlap. Controlling the probability of bad events is a significant requirement for insuring the functioning of digital world and in particular smart city. For achieving this goal, the first problem to handle is to identify the bad events. This could be done by defining what we call a bad event: traffic jam, communication network congestion, no cars where they are needed, no electric supply... Then, the exploration and target policies of each optimization problem could be designed for controlling the probabilities of these bad events. The difficulty of this approach lies in the combinationatorial number of events that could lead to a bad event. Another approach could be the extensive use of simulations for learning exploration and target policies that minimize the probability of bad events.

Chapter 2

Bandits and Reinforcement Learning

In reinforcement learning an agent interacts with an environment by playing actions for gathering as much as possible rewards. For discussing the main models of reinforcement learning, it is convenient to use the Markov Decision Process framework. The environment is described by its state s that belongs from a set of states $\mathcal{S}$. The action k taken by the agent belongs from a set of actions $\mathcal{K}$. The agent is an algorithm $\pi \in \Pi : \mathcal{S} \to \mathcal{K}$ that knowing the state s of the environment chooses an action k in order to maximize the gathered rewards over time. The set of algorithms Π has to be chosen according to the problem being treated.

2.1 Stochastic multi-armed bandit

The simplest reinforcement learning problem is the stochastic multi-armed bandit.

Figure 2.1: Stochastic bandit

In stochastic multi-armed bandit (figure 2.1), the environment has a single state, the reward of played action (or arm) is sampled from a distribution with an unknown mean, and the goal of the agent is to maximize the gathered rewards over time. Despite its simplicity, the multi-armed bandit problem has been widely studied since 1933, where the Thompson Sampling algorithm has been proposed for choosing the best medical treatment [Thompson, 1933]. The first reason of this interest is theoretical. The multi-armed bandit problem is the hearth of one of the main problems of reinforcement learning, the *exploration-exploitation* dilemma: the agent has to play loosely estimated arms for finding the best arm, or play the estimated best arm for maximizing its cumulative reward. The second reason is practical. This simple problem has a lot of applications in the digital world. For instance, choosing the best banner for maximizing the clicks on a ad, is a multi-armed bandit problem, as well as choosing the best message

for a marketing campaign. Indeed at the beginning of the game, the click-through rates of banners or messages are unknown. The agent has to explore and exploit them for optimizing the gathered clicks. Twenty years ago (and sometimes today), the choice of messages for optimizing the marketing campaigns was done using A/B testing, which consists in comparing the hypotheses A and B using a statistical hypothesis test. However the drawback of this approach is that the sample sizes of populations A and B have to be set before seeing any data. While this makes sens when the samples have to be recruited before launching the hypothesis test, as a cohort of patients for testing the effects of some medical treatments, this approach is clearly sub-optimal in the digital world where the samples are collected online.

In the following, we recall the main concepts and results on the stochastic bandit problem. For more exhaustive view, the reader should report on [Bubeck and Cesa-Bianchi, 2012, Lattimore and Szepesvári, 2020].

Definition 1 (Stochastic Multi-Armed Bandit)**.** *The stochastic Multi-Armed Bandit (MAB) is a* 2*-tuple* $(\mathcal{K}, \nu_k)$*:*

- *the set of actions (or arm)* $\mathcal{K}$ *can be infinite or finite (in this case* $[K]$ *denotes the set of* K *arms),*
- *the reward* x_k *is a random variable sampled from a parametric probability distribution* ν_k*:* $x_k \sim \nu_k(\mu_k)$*, where* μ_k *is an unknown parameter.*

In stochastic multi-armed bandit, at each time step $t \in [T]$*, where* T *is the time horizon, the agent has to choose an arm* $k_t \in \mathcal{K}$*, and then the environment emits a reward* $x_{k_t}(t) \sim \nu_{k_t}(\mu_{k_t})$*.*

Remark 1. *In stochastic multi-armed bandit, the set of states is reduced to the singleton:* $\mathcal{S} = \{s_0\}$*.*

The measure of performance of bandit algorithms is called the regret.

Definition 2 (Regret)**.**

$$r(T) = \max_{k \in \mathcal{K}} \sum_{t=1}^{T} (x_k(t) - x_{k_t}(t)) \tag{2.1}$$

In stochastic bandits, the reward x_k is sampled from a distribution of unknown mean μ_k. Hence, it seems reasonable to analyze the expected regret.

Definition 3 (Expected regret)**.**

$$\mathbb{E}\left[r(T)\right] = \mathbb{E}\left[\max_{k \in \mathcal{K}} \sum_{t=1}^{T} (x_k(t) - x_{k_t}(t))\right] \tag{2.2}$$

However, the expected regret raises a conceptual issue. Indeed, using this definition of regret, the best arm depends on the particular sample that has been drawn, and hence is not unique. That is why, for analyzing purpose one generally prefers the pseudo-regret.

Definition 4 (Pseudo-regret in stochastic bandits)**.**

$$R(T) = \max_{k \in \mathcal{K}} \mathbb{E}\left[\sum_{t=1}^{T}(x_k(t) - x_{k_t}(t))\right] = \sum_{t=1}^{T}(\mu_{k^*} - \mu_{k_t}). \tag{2.3}$$

Here, the best policy that can be performed by the agent is to choose every time step the arm k^* with the greatest mean reward : $k^* = \arg\min_{k \in \mathcal{K}} \mu_k$.

Remark 2 (pseudo-regret versus expected regret)**.** *The pseudo-regret is a weaker notion of regret than the expected regret. Indeed, due to Jensen's inequality, we have:*

$$\mathbb{E}\left[\max_{k \in \mathcal{K}} \sum_{t=1}^{T}(x_k(t) - x_{k_t}(t))\right] \geq \max_{k \in \mathcal{K}} \mathbb{E}\left[\sum_{t=1}^{T}(x_k(t) - x_{k_t}(t))\right],$$
$$\Leftrightarrow \mathbb{E}\left[r(T)\right] \geq R(T).$$

Let $n_k(t) = \sum_{s=1}^{t} \mathbb{1}_{k_t = k}$ be the number of times the arm k is selected until time t, let $\Delta_k = \mu_{k^*} - \mu_k$ denotes the gap between the mean reward of arm k and the optimal arm k^*, and let K be the number of arms.

Remark 3 (useful form of the pseudo-regret)**.** *The pseudo-regret can be re-written as follow:*

$$\begin{aligned} R(T) &= \max_{k \in [K]} \mathbb{E}\left[\sum_{t=1}^{T}(x_k(t) - x_{k_t}(t))\right] \\ &= \mu_{k^*} T - \mathbb{E}\left[\sum_{t=1}^{T} x_{k_t}(t)\right] \\ &= \mu_{k^*} \sum_{k=1}^{K} \mathbb{E}[n_k(T)] - \sum_{k=1}^{K} \mu_k \mathbb{E}[n_k(T)] \\ &= \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)]. \end{aligned} \tag{2.4}$$

Equation (2.4) shows that the pseudo-regret depends on the number of times a sub-optimal arm is played. This is the basis on which the regret lower bound of the multi-armed bandit problem as well as regret upper of algorithms are built.

Definition 5 (Kullback-Leiber divergence [Cover and Thomas, 2005])**.** *The Kullback-Leiber divergence between two probability mass functions $p(x)$ and $q(x)$ is:*

$$KL(p, q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{2.5}$$

Theorem 1 (pseudo-regret lower bound for Bernoulli bandit [Lai and Robbins, 1985])**.** *For any set of Bernoulli distributions $\nu_1, ..., \nu_K$, such that for any arm k $\Delta_k > 0$, any policy π that satisfies $\mathbb{E}[N_k(T)] = o(T^a)$ for any arm k,*

any $a > 0$, we have:

$$\liminf_{T\to\infty} R(T) \geq \sum_{k=1}^{K} \frac{\Delta_k}{KL(\upsilon_k, \upsilon_{k^*})} \log T.$$

2.2 Switching stochastic Multi-Armed Bandit

In real world, the mean rewards of arms evolve during time. For instance, the reliability of a particular channel in a wireless network can evolve according to the weather, to unknown devices that try to send messages through the same channel, to the move of the device itself, the effectiveness of a marketing campaign or the click-through rate of banners can change according to external event such as marketing campaigns launched by other actors. That is why the study of non-stationary bandits is significant for automatizing bandit applications. The simplest non-stationary bandit is the switching stochastic bandit.

In switching stochastic bandit, the state of the environment changes according to a function of the previous state (figure 2.2). For choosing its actions, the agent does not observe the state of the environment. Between the changes of state of the environment, the switching stochastic bandit problem is reduced to the stochastic bandit problem: the rewards are sampled from the same distributions with unknown means.

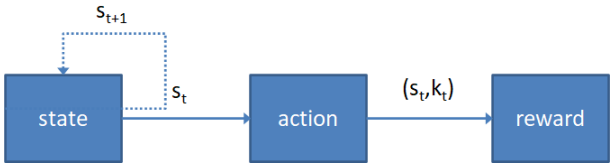


Figure 2.2: Switching stochastic bandit

Definition 6 (Switching stochastic Multi-Armed Bandit)**.** *The Switching stochastic Multi-Armed Bandit (SMAB) is a 4-tuple $([K], \nu_{k,t}, \mathcal{S}, h)$:*

- *a finite set of actions (or arm) $[K]$,*
- *the reward x_k is a random variable sampled from a parametric probability distribution $\nu_{k,t}$: $x_k \sim \nu_{k,t}(\mu_{k,t})$, where $\mu_{k,t}$ is an unknown parameter that depends on times,*
- *a discrete and infinite set of states $\mathcal{S}$, where the state $s_t \in \mathcal{S}$ is not observed and corresponds to a set of K parameters $\mu_{1,t}, ..., \mu_{K,t}$,*
- *a function $h : \mathcal{S} \to \mathcal{S}$ that can be deterministic or stochastic.*

In Switching stochastic Multi-Armed Bandit, at each time step $t \in [T]$, where T is the time horizon, the agent has to choose an arm $k_t \in [K]$, then the environment emits a reward $x_{k_t}(t) \sim \nu_{k_t,t}(\mu_{k_t,t})$, and then the environment can switch its state according to $s_{t+1} := h(s_t)$.

The pseudo-regret is defined as the difference of mean rewards between the arms chosen by the optimal policy and those chosen by the agent.

Definition 7 (Pseudo-regret in switching stochastic bandits)**.**

$$R(T) = \max_{k \in \mathcal{K}} \mathbb{E}\left[\sum_{t=1}^{T}(x_k(t) - x_{k_t}(t))\right] = \sum_{t=1}^{T}\left(\mu_{k^*,t} - \mu_{k_t,t}\right), \tag{2.6}$$

where $(k^*, t) = \arg\max_{k \in [K]} \mu_{k,t}$.

Theorem 2 (pseudo-regret lower bound for switching stochastic bandit [Garivier and Moulines, 2011])**.** *For any policy* π *and any time horizon* T*, it exists distributions of rewards and switches, and* $c > 0$ *such that:*

$$R(T) \geq \sqrt{cT}.$$

Remark 4 (Tightness)**.** *This pseudo-regret lower bound may not be tight for the switching stochastic bandit since it does not depend on the number of arms* K *and more significantly on the number of changes of best arm* γ_T*.*

2.3 Adversarial Multi-Armed Bandit

The most general non-stationary bandit is the non-stochastic bandit [Auer et al., 2002b, Cesa-Bianchi and Lugosi, 2006]. In non-stochastic bandit, no statistical assumption is made: the agent competes against an adversary. The adversary, which knows the player's policy, has generated a deterministic sequence of rewards before the game starts. As a consequence the player, which knows the time horizon T, has to randomize its policy, else the adversary could generate a sequence for which the regret is T. This problem is so general that it can handle any kind of non-stationarity. That is why, usually the algorithms that handle the adversarial bandit are more conservative than switching stochastic bandits. In practice the adversarial bandit should be used only when no information is available about the process that generates the reward.

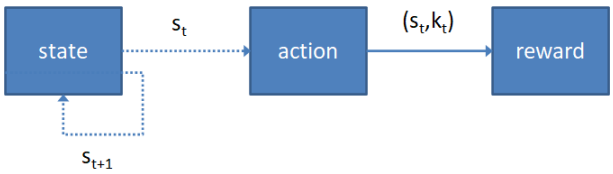


Figure 2.3: Adversarial bandit

Definition 8 (Adversarial Multi-Armed Bandit)**.** *The Adversarial Multi-Armed Bandit is a* 3*-tuple* $([K], \boldsymbol{X}_T, \mathcal{S})$*:*

- *a finite set of actions (or arm)* $[K]$*,*

- *the sequence of rewards $X_T = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$ is chosen by the adversary before the game starts, where $\mathbf{x}_t = x_1(t), ..., x_K(t)$ and $x_k(t) \in [0, 1]$,*
- *the set of states $\mathcal{S}$, where the state $s_t \in \mathcal{S}$ is not observed by the agent and corresponds to a time period in which the arm that provides the maximal reward does not change,*

In adversarial multi-armed bandit the time horizon T is known, and at each time step $t \in [T]$ the agent has to choose an arm $k_t \in [K]$, then the adversary reveals the reward $x_{k_t}(t)$.

The definitions of regret in adversarial bandit are similar to those in stochastic bandit, except that in adversarial bandit the expectation is taken over the randomization of the player.

Theorem 3 (pseudo-regret lower bound for adversarial bandit [Auer et al., 2002b])**.** *For any number of arms $K \geq 2$ and for any time horizon T, there exists a distribution over the assignment of rewards such that the pseudo regret $R(T)$ of any algorithm (where the expectation is taken with respect to both the randomization over rewards and the algorithm's internal randomization) is at least*

$$\frac{1}{20} \min\left(\sqrt{KT}, T\right).$$

Remark 5 (Generality of this pseudo-regret lower bound)**.** *Theorem 3 is given for any distribution of assignment of rewards, and hence it also applies to the switching bandit problem.*

Remark 6 (oblivious versus non-oblivious adversary)**.** *The oblivious adversary generates the sequence of rewards $\mathbf{x}_1, ..., \mathbf{x}_T$ before the game starts, whereas a non-oblivious adversary chooses a set of functions $g_1, ..., g_T$, where $g_t : [K]^{t-1} \to \mathcal{S}$ is a function of the past decisions $k_1, ..., k_{t-1}$. That is why, even if the pseudo-regret lower bound holds for a non-oblivious adversary, with such adversary the quantity of interest should be the expected regret (see Definition 3).*

2.4 Contextual Multi-Armed Bandit

In lot of applications, in addition to the reward of the chosen arm, the agent can observe the environment. For instance, before choosing an ad for a given cookie, the agent observes the profile of the cookie and the description of the ad. In practice, taking into account this contextual information dramatically increases the efficiency of the agent [Li et al., 2010a]. In contextual bandit the state of the environment evolves (figure 2.4). For instance, the profile of the cookie can be sampled from an unknown distribution, and hence the observation of the environment changes at each step [Langford and Zhang, 2007]. The environment can also change according to the chosen arm. For instance, in cloud auto-scaling the agent observes the number of resources (containers, virtual machines,...), the workload, and it chooses to increase or decrease the number of resources. While the choice of the agent changes

the number of resources and hence the state of the environment, the optimal policy for choosing to increase or to decrease the number of resources is still the same.

Definition 9 (Contextual Multi-Armed Bandit)**.** *The contextual bandit is a* 4*-tuple* $([K], \mathcal{S}, \mathcal{M}, \nu_{k,\boldsymbol{s}})$*:*

- *a finite set of actions (or arm)* $[K]$*,*
- *the set of states* $\mathcal{S}$ *(or contexts), where the state* $\boldsymbol{s}_t \in \mathcal{S}$ *is observed,*
- *a given class of model* $\mathcal{M} : \mathcal{S} \to \mathbb{R}$*, with an unknown optimal model* $\boldsymbol{\theta} \in \mathcal{M}$*,*
- *the reward* $x_{k,\boldsymbol{s}}$ *is a random variable that is sampled from a parametric probability distribution* $\nu_{k,\boldsymbol{s}}$*:* $x_{k,\boldsymbol{s}} \sim \nu_{k,\boldsymbol{s}}(\boldsymbol{\theta})$*,*

In contextual bandit, at each time step t *a state* s_t *is revealed to the agent, the agent chooses an arm* $k_t \in [K]$*, then the agent receives a reward* $x_{k_t,\boldsymbol{s}_t}(t) \sim \nu_{k_t,\boldsymbol{s}_t}(\boldsymbol{\theta})$*.*

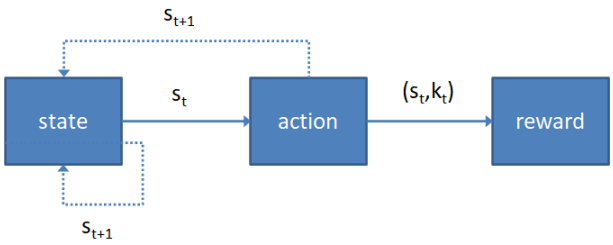


Figure 2.4: Contextual bandit

In contextual bandit, the regret is defined with respect to the optimal parameter of a given model. In other words, the bias of the model with respect to the optimal model is not taken into account. There are plenty of models coming from machine learning that have been adapted to contextual bandit: linear bandit [Li et al., 2010a, Filippi et al., 2010, Abbasi-yadkori et al., 2011], kernel bandit [Valko et al., 2013], bandit forest [Féraud et al., 2016], neural bandit [Allesiardo et al., 2014, Zhou et al., 2020]. We choose here to illustrate the contextual bandit using the linear bandit. The reason is that the linear bandit is efficient in practice, well known in theory, and lot of contextual bandits are built upon it.

Definition 10 (Linear Multi-Armed Bandit)**.** *A linear bandit is a contextual bandit, for which the reward of each arm* $k \in [K]$ *is sampled according to:*

$$x_k(t) = \boldsymbol{s}_k^\top \boldsymbol{\theta} + \eta_t, \tag{2.7}$$

where $\boldsymbol{s}_k \in \mathbb{R}^d$ *is the context of the arm* k*,* $\boldsymbol{\theta} \in \mathbb{R}^d$ *is an unknown parameter, and* η_t *is white noise that is* R*-sub-Gaussian:*

$$\forall\, \lambda \in \mathbb{R}, \quad \mathbb{E}[e^{\lambda \eta_t}] \leq \exp\left(\frac{\lambda^2 R^2}{2}\right).$$

Remark 7 (disjoint and hybrid linear bandit)**.** *In the previous definition the state s_k depends only of the arm, which is restrictive. The linear bandit definition could be done using disjoint linear models:*

$$x_k(t) = \boldsymbol{s}_t^\top \boldsymbol{\mu}_k + \eta_t, \text{ where } \boldsymbol{\mu}_k \in \mathbb{R}^d \text{ is the parameter of arm } k,$$

or with hybrid linear models:

$$x_k(t) = \boldsymbol{s}_t^\top \boldsymbol{\mu}_k + \boldsymbol{s}_k^\top \boldsymbol{\theta} + \eta_t.$$

Definition 11 (Pseudo-regret in linear bandit)**.**

$$R(T) = \sum_{t=1}^{T} \left(\boldsymbol{s}_{k^*}^\top \boldsymbol{\theta} - \boldsymbol{s}_{k_t}^\top \boldsymbol{\theta} \right), \tag{2.8}$$

where $k^* = \arg\max_{k \in [K]} \boldsymbol{s}_k^\top \boldsymbol{\theta}$, and k_t is the arm chosen by the agent.

Theorem 4 (pseudo-regret lower bound of linear bandit [Chu et al., 2011])**.** *For the linear bandit for any T and K such that $T \geq K \geq 2$, for any algorithm choosing action at time t, there exists is a constant $c > 0$ and a set of states $\mathcal{S}$ such that for $d^2 \leq T$:*

$$R(T) \geq \sqrt{cTd}$$

Remark 8. *Interestingly the pseudo-regret lower bound of linear bandit does not depend on the number of arm K (Theorem 4). As OFUL algorithm [Abbasi-yadkori et al., 2011] reaches this lower bound at some logarithm factors of T, it is tight. This means that the linear bandit can be used with large set of arms without regret penalty. However, the linear bandit suffers from the dimension of contexts d, in terms of regret and computational time cost, which is quadratic in d.*

2.5 Markov Decision Process

As Multi-Armed Bandit, Markov Decision Process [Puterman, 1994, Sutton and Barto, 2018] are also a particular case of reinforcement learning, that satisfies the Markov property: the next state depends only on the current state. As in contextual bandit, the agent observes a state, chooses an action, receives a reward, and then the environment changes its state (figure 2.5). The difference is that in Markov Decision Process (MDP), when the state changes, the optimal policy to choose the arm in the next state also changes. Hence in MDP, given an initial state, the optimal policy for gathering rewards consists in playing sequences of arms. This chain rule raises an optimization problem: as the reward depends on the trajectory through the state space, it could be optimal for the agent to choose

an action that does not provide an immediate reward, but that leads to states that provide great rewards. This optimization problem, also known as *credit assignment problem* is addressed by dynamic programming [Bellman, 1957]. Two kinds of Markov Decision Process can be distinguished. In episodic MDP there are terminal states, which corresponds for instance to the end of a game. In non-episodic MDP, there is no terminal state, for instance controlling a domestic robot is a continuous task. In such tasks, a discount factor is needed to forget the past.

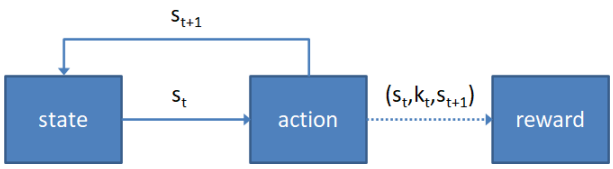


Figure 2.5: Markov Decision Process

Definition 12 (Markov Decision Process)**.** *A Markov Decision Process is a* 4*-tuple* $([K], [S], \tau(s'|s,k), \nu_{k,s,s'})$*:*

- *a finite set of actions (or arm)* $[K]$*,*
- *a finite set of states* $[S]$*, where the state* $s_t \in [S]$ *is observed, and the state* $s_0 \in [S]$ *is the initial state,*
- $\tau(s'|s,k)$ *an unknown probability of transition from state* $s \in [S]$ *to state* $s' \in [S]$ *when arm* k *is chosen,*
- *the reward when arm* k *is chosen in state* s *with a transition to state* s'*,* $x_{k,s,s'}$*, is a random variable that is sampled from an unknown parametric probability distribution* $\nu_{k,s,s'}$*:* $x_{k,s,s'} \sim \nu_{k,s,s'}$*.*

In Markov Decision Process, at each time step t *a state* s_t *is revealed to the agent, the agent chooses an arm* $k_t \in [K]$*, then the agent receives a reward* $x_{k_t,s_t,s_{t+1}}(t)$*, and the environment changes its state to* s_{t+1}*.*

Definition 13 (Optimal policy)**.** *Let* M *be an MDP,* $\pi \in \Pi : [S] \to [K]$ *be a policy, and* T *be the time horizon. The optimal policy* $\pi^* \in \Pi$ *is:*

$$\pi^* = \arg\max_{\pi \in \Pi} \mu(M, \pi, s_0),$$
$$\text{where } \mu(M, \pi, s_0) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[x_{k_t,s_t,s_{t+1}}].$$

Definition 14 (Pseudo-regret in MDP)**.**

$$R(T) = \sum_{t=1}^{T} \mu(M, \pi^*, s_0) - \mathbb{E}[x_{k_t,s_t,s_{t+1}}].$$

Now, to provide a lower bound of the pseudo-regret, an additional notion related to the hardness of the state transition structure is needed.

Definition 15 (Diameter of MDP [Jaksch et al., 2010]). *Let $\pi \in \Pi : [S] \to [K]$ be a policy, M be an MDP, and $s_0 \in [S]$ be the initial state, and $T(s'|M, \pi, s_0)$ be the random variable denoting the event $s' \in [S]$ is reached for the first time. The diameter of M is:*

$$D(M) = \max_{s_0 \in [S], s_0 \neq s'} \min_{\pi \in \Pi} E[T(s'|M, \pi, s_0)].$$

Theorem 5 (pseudo-regret lower bound of MDP [Jaksch et al., 2010]). *For any policy π, any natural number $S, K \geq 10$, $D(M) \geq 20 \log_K S$, and $T \geq DSK$, there exists an MDP M with a diameter D, such that for any initial state $s_0 \in [S]$ we have:*

$$R(T) \geq 0.015\sqrt{DSKT}.$$

Remark 9. *The pseudo-regret lower bound of MDP and contextual bandit scale in $\Omega(\sqrt{T})$, while the optimal policy to be learnt is much more complex in case of MDP. Firstly in case of contextual bandit the number of states can be infinite, while the pseudo-regret lower bound of MDP states that obtaining a finite pseudo-regret is hopeless in such case. Secondly, in the case of contextual bandit, the state could be partially observed, while finding the optimal Partially Observed Markov Decision Process is PSPACE-hard [Papadimitriou and Tsitsiklis, 1987].*

2.6 Multi-Player Multi-Armed Bandit

The multi-player bandit [Liu and Zhao, 2010] has been motivated by opportunistic spectrum access [Santivanez et al., 2006] in cognitive radio: primary users have a strict priority over secondary users, which are allowed *sensing* a channel before sending a packet in order to check that it is free. The objective is to avoid collisions between concurrent secondary users, that share the same channels, while choosing the best channels, i.e. with the highest probabilities to be free of primary users. In multi-player bandit, there are an unknown number of agents (or players), which is less than the number of channels, that simultaneously choose an arm at each time step. When two or more players choose the same are, a collision occurs and the rewards of those players is zero, else the player receives the reward of the chosen arm, which is sampled from a parametric distribution with unknown mean. As a consequence, for each player, the other players are a part of the environment. More precisely for each player, the state of the environment corresponds to the chosen arms of other players, it is not observed, and it can change at each time step (figure 2.6).

Definition 16 (Multi-Player Multi-Armed Bandit). *The multi-player bandit is a 3-tuple $([N], [K], \nu_k)$:*

- *the set of player $[N]$, where $N \leq K$ is unknown to players,*
- *the set of arms $[K]$,*

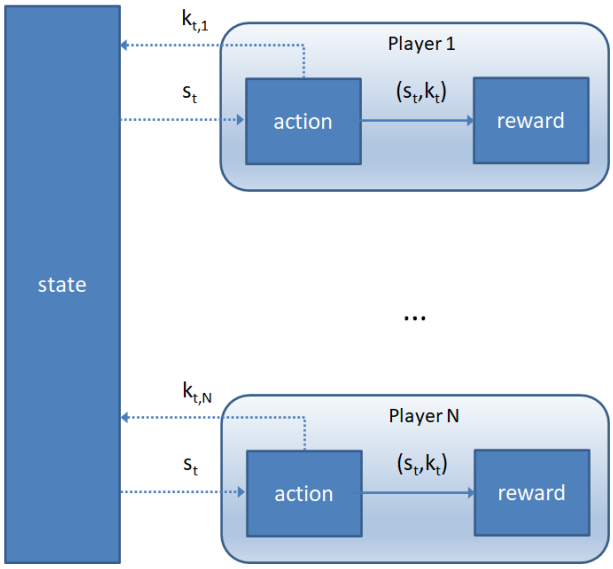


Figure 2.6: Multi-player bandit

- *the reward of player n when choosing arm k_n is a random variable such that: $x_{k_n} = y_{k_n}(1 - \eta_{k_n})$, where $\eta_k = \mathbb{1}_{|n\in[N],k_n=k|>1}$ is the collision indicator and y_k is sampled from a parametric probability distribution ν_k: $y_k \sim \nu_k(\mu_k)$, where μ_k is an unknown parameter.*

In multi-player bandit, at each time step $t \in [T]$, where T is the time horizon, each agent $n \in [N]$ simultaneously choose an arm $k_{n,t} \in [K]$. Then for each player n the environment emits a reward $k_{n,t}$ corresponding to its state s_t, which is an assignment of players over arms $s_t = \{k_{1,t}, ..., k_{N,t}\}$. The collision indicator η_k can be observed or not by the players.

Definition 17 (Pseudo-regret in multi-player bandit)**.** *Let assume that the index of mean reward of arms is such that: $\mu_1 \geq \mu_2 ... \geq \mu_K$. Then, the pseudo regret with respect to an optimal policy is:*

$$R(T) = \sum_{k=1}^{N} \mu_k T - \mathbb{E}\left[\sum_{t=1}^{T}\sum_{n=1}^{N} x_{k_{n,t}}(t)\right]$$

Remark 10 (The optimal policies)**.** *Formally, a policy π is a vector of probability distributions over the set of arms: $\pi = (\pi_1, ..., \pi_N)$, with $\pi_n = (\pi_n^1, ..., \pi_n^K)$, where π_n^k denotes the probability that player n chooses arm k. In multi-player bandit, an optimal policy is an optimal assignment over arms:*

$$\forall n \in [N] \quad \pi_n^* \in \{(\pi_n^{*,1}, ..., \pi_n^{*,k_n}, ..., \pi_n^{*,K}) : k_n \leq N, \forall k \in [K] \setminus \{k_n\} \quad \pi_n^{*,k} = 0, \forall n' \in [N] \setminus \{n\} \quad k_{n'} \neq k_n\}.$$

Theorem 6 (pseudo-regret lower bound for multi-player Bernoulli bandit [Anantharam et al., 1987])**.** *For any set of Bernoulli distributions $\nu_1, ..., \nu_K$ such that $\mu_1 \geq \mu_2 ... \geq \mu_K$, for any $N \leq K$, for any policy π that satisfies*

$\mathbb{E}[N_k(T) = o(T^a)]$, *any* $a > 0$, *we have:*

$$\liminf_{T\to\infty} R(T) \geq \sum_{k>N} \frac{\mu_N - \mu_k}{KL(\upsilon_k, \upsilon_N)} \log T.$$

Remark 11 (Decentralized versus centralized algorithms)**.** *This pseudo-regret lower bound is stated in the case when the player can communicate without limitation with other players, in other words in the centralized case. In the decentralized case, two attempts of lower bound [Liu and Zhao, 2010, Besson and Kaufmann, 2018], suggesting that in decentralized case the pseudo-regret lower bound suffers from a factor* N *in comparison to the centralized case, have been proposed. But these lower bounds did not take into account that a communication protocol can be built using collisions [Boursier and Perchet, 2019a]. Using the same communication protocol, a decentralized algorithm, for which the upper bound of the pseudo-regret reaches the lower bound, has been provided in [Wang et al., 2020].*

Chapter 3

Bandits in evolving environments

3.1 Learning the best learning expert

Abstract. The contextual bandits can be viewed as a generalization of online classification models, where only the chosen class is observed. The selection of learning experts allows to find the best parametrization of an expert during its learning, within a set of predefined parameters, and reduces the bias of the hypothesis space, and hence improves the performances. As the contextual bandits learn, their performances tend to increase during time, and hence the choice of the best one implies to solve a non-stationary problem. We provide a theoretical framework to solve this difficult problem. A first approach to handle the selection of learning experts problem is to reduce it to stochastic problems: the experts are learned in parallel during a first phase, then they are explored and exploited in a second phase. A second approach models the selection of learning experts as an adversarial problem with a finite budget of *contaminated rewards*. We call this setting *adversarial bandits with budget*. Here, experts are learned, selected and exploited at the same time. When the budget of *contaminated rewards* is known, we propose and analyze a randomized variant of the algorithm SUCCESSIVE ELIMINATION. When this budget is unknown, we analyze the algorithm EXP3 for the proposed setting. We illustrate both approaches in the case where the learning experts are based on BANDIT FORESTS initialized with different sets of parameters.

3.1.1 Introduction

The *contextual bandit problem* [Langford and Zhang, 2007] is a repeating game where a player must select actions after observing a context. At each step, it receives a reward which depends on the played action and the associated context. The rewards of not chosen actions are never revealed. The goal is to minimize the *regret* expressed as the difference between rewards that could be acquired by an *optimal policy* knowing hidden parameters of the problem.

There are two popular approaches to deal with the contextual bandit problem. The first is based on policy

selection algorithms ([Cesa-Bianchi and Lugosi, 2006], [Dudík et al., 2011], [Agarwal et al., 2014]). At each step, a player chooses a policy within a known set of policies. Then, the selected policy chooses the action to play. The modeling of the dependence between actions, rewards and contexts is delegated to the policies. The policy selection algorithms do not have to manage the environment. The second approach to handle the contextual bandit problem is based on the learning of a model from the feedback obtained through the game. Here, the algorithms interact directly with the environment by choosing the player action. At each step, the available information (context, played action, reward) is used to improve the model. Such approach can be performed with linear models (e.g., [Kakade et al., 2008a], [Li et al., 2010b]), neural networks [Allesiardo et al., 2014] or random forest [Féraud et al., 2016]. A methodology for using any batch learning algorithm in a contextual bandit setting has also been proposed by [Langford and Zhang, 2007] with the epoch-greedy algorithm.

However, both approaches suffer from weaknesses. The policy selection algorithms have strong theoretical guarantees but in practice their performances depend of the availability of a good policy within the set of policies. Moreover, their computational time cost in $O(\text{poly(T)})$ could be an issue when the time horizon T is large. Learning a policy directly from the data could be very effective in practice and in theory but still has an approximation error due to the bias of the hypothesis space considered by the algorithm: the best linear model, the best random forest of depth D and size L, a reasonably good multi-layer perceptron with h hidden neurons...

In this paper, we consider an hybrid approach, the selection of learning experts. We call a learning expert the tuple characterizing an instance of a contextual bandit algorithm (parameters, random seed, ...). At each round, a player chooses an expert and this expert selects an action after observing a context vector generated by the environment. Both receive a feedback and the expert can modify its policy with this new observation. The player has to estimate the performances of experts and to optimize their selection. Each expert learns in order to minimize the estimation error within its hypothesis space. The player minimizes the approximation error among the experts. This hybrid approach tackles the main weakness of the two previous approaches for the contextual bandit problem. Firstly, the exploration of the hypothesis space (i.e. the set of policies) is done by efficient algorithms with strong theoretical guarantees such as LINUCB [Li et al., 2010b], or BANDIT FOREST [Féraud et al., 2016]. Secondly, the bias of these algorithms is reduced by selecting the expert with the highest performances. Selection of learning experts is a non-trivial problem. Indeed, as experts learn and can modify their policies over time, the optimal algorithms for stochastic multi-armed bandit problem, such as UCB [Auer et al., 2002a] or Thompson Sampling ([Agrawal and Goyal, 2012], [Kaufmann et al., 2012]), cannot be used directly. Moreover, the adversarial multi-armed bandit algorithms such as EXP3 or EXP4 [Auer et al., 2002b, Allesiardo and Féraud, 2015] do not take advantage of the fact that the rewards of learning experts tend to increase over time.

Our contribution Contrarily to offline learning, where the learner has beforehand a set of samples from the joint distribution of contexts and rewards $D_{x,y}$, in online learning, $D_{x,y}$ is unknown at the beginning of the optimization.

As we cannot try several learning algorithms and parameters before the deployment, to control the risk of deploying models, we need worst case theoretical guarantees. We provide a theoretical framework and two selection algorithms that can be applied to the cases of bandit information (the rewards are partially observed), and full information (the rewards of not chosen actions are not observed). In the full information setting the dependence in K of the lower and upper bounds are removed, and the vector of rewards y is fully observed by experts.

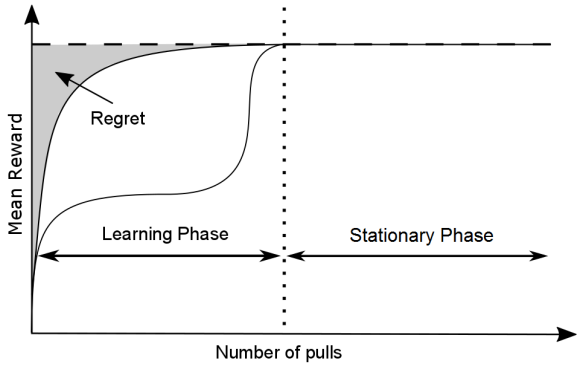


Figure 3.1: Two examples of mean reward obtained by a learning algorithm over time.

A promising approach for learning experts selection is to weaken the adversarial hypotheses and to include information about the non-stationarity of the learning phase. Based on the shape of the regret curves of contextual bandit algorithms (see figure 3.1), we can make the following observations:

- during the learning phase, the mean reward of experts tend to increase over time,
- after the learning phase, the mean reward can be considered to be constant over time.

The shape of the expert learning curves depends on the algorithm and the values of parameters. The expert can quickly converge to a good solution and then take a long time to reach the optimum, or the learning curve can have a plateau for a long time and then quickly converge to the optimum (see figure 3.1). Two quantities characterize the learning curves:

- The sample-complexity: the number of pulls needed to obtain the best policy within the set of experts with high probability,
- the regret accumulated over time versus the best policy within the set of experts: the area above the learning curve.

Two natural settings appear from these observations. The first is based on the PAC-setting [Valiant, 1984]. The experts are learned, and after enough time, mean rewards of experts become stationary. Thus, after the learning phase, a stochastic MAB algorithm can be used to select the best expert. This setting is presented in Section 3.1.3. Using SUCCESSIVE ELIMINATION algorithm [Even-Dar et al., 2006] to select the best expert, we provide an analysis of the selection experts after then learning: we showed that the proposed approach is optimal up to logarithmic

factors when near optimal experts such as BANDIT FOREST [Féraud et al., 2016] are used. This approach takes advantages of the exploration phase, where actions are sequentially played, to share observations between experts, and thus to learn them in parallel.

The second setting, presented in Section 3.1.4 uses a similar approach than *contaminated rewards* introduced by [Seldin and Slivkins, 2014], where the rewards are mainly drawn from stationary distributions except for a minority of rewards of means chosen in advance by an adversary. We adapted this setting to the selection of learning experts. Here, as the performances over time of each learning expert are bounded by those of the best expert of each hypothesis space, the maximal mean of *contaminated rewards* is bounded by the mean of the best one. Moreover, to take into account that the performances of learning experts tend to increase and their regret is bounded, we bound the amount of contamination available to the adversary. The time steps of contaminated rewards and the means of their distributions are arbitrary chosen by an oblivious adversary before the beginning of the run. The other rewards are drawn from the distribution with highest mean reward within the hypothesis space of the expert. We call this setting *adversarial bandits with budget*. In Section 3.1.3, we propose a randomized version of the algorithm SUCCESSIVE ELIMINATION for the case where the budget B is known. We take advantage of this randomized version to share the observations of played experts with the others: the reward used to update each expert is divided by the probability to select the action. In comparison to the first setting, the strength of this approach is to play more and more the best experts over time. The weakness of this approach is that it necessitates to know the budget of contaminated rewards. When the budget is unknown, we analyzed the adversarial bandit algorithm EXP3 for *adversarial bandits with budget*. In Section 5.2.7, we illustrate experimentally our results using the BANDIT FOREST algorithm to learn the experts.

3.1.2 Selection of Learning Expert Problem

Let $A = \{1, ..., K\}$ be a set of actions. Let x_t be a context vector describing the environment at time t. Let y_t be a vector of rewards at time t and $y_k(t) \in [0, 1]$ the reward of the action k at time t. Let $D_{x,y}$ be a joint distribution on (x, y). Let $\pi : X \to A$ be a policy. Let $S = \{1, ..., M\}$ be the set of indexes of experts. Let Π_m be the set of policies reachable during the learning of the expert $m \in S$ and $\Pi = \bigcup_{m=0}^{M} \Pi_m$ be the set of policies. The policy used by the expert m at time t is denoted $\pi_{m,t}$. Its mean reward is defined by:

$$\mu_m(t) = \mathbb{E}_{D_{x,y}}[y_{\pi_{m,t}(x)}].$$

The optimal policy of the set Π_m is defined by:

$$\pi_m^* = \arg\max_{\pi \in \Pi_m} \mathbb{E}_{D_{x,y}}[y_{\pi(x)}].$$

The optimal policy is defined by:

$$\pi^* = \arg\max_m \pi_m^* \,.$$

The expected regret of the learning experts selection task is defined by:

$$\mathbb{E}_{D_{x,y}}[R(T)] = \mathbb{E}_{D_{x,y}}\left[\sum_{t=1}^{T}\left(y_{\pi^*(x_t)}(t) - y_{\pi_{m_t,t}(x_t)}(t)\right)\right],$$

where m_t is the expert played at time t, and $\pi_{m_t,t}$ is the policy selected at time t in the set Π_{m_t}.

Let m^* be the index of the subset containing π^*, let μ_m be the best policy in the set Π_m, and $\hat{\mu}_m(t)$ be the empirical mean reward at time t of the learning expert m.

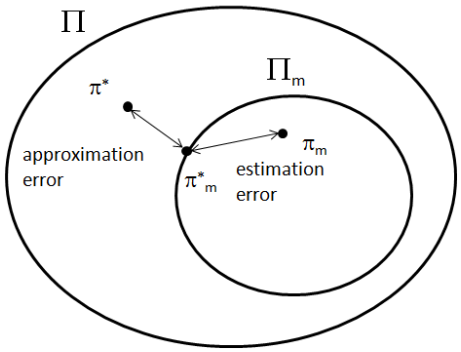


Figure 3.2: The learning algorithm task minimizes the estimation error in the set Π_m. The learning experts selection task reduces the approximation error.

The decomposition of the expected regret of the expert m in terms of approximation and estimation (see figure 3.1.2) is given below:

$$\begin{aligned}\mathbb{E}_{D_{x,y}}[R_m(T)] &= \sum_{t=0}^{T}\left(\mu_{m^*} - \hat{\mu}_m(t)\right) \\ &= \sum_{t=0}^{T}\left(\mu_{m^*} - \mu_m\right) + \sum_{t=0}^{T}\left(\mu_m - \hat{\mu}_m(t)\right).\end{aligned}$$

Notice that the expected regret of the best expert m^* has no approximation error. Now, the expected regret of an algorithm of selection of learning experts can be defined by:

$$\mathbb{E}_{D_{x,y}}[R(T)] = \sum_{t=0}^{T}\left(\mu_{m^*} - \mu_{m_t}\right) + \sum_{t=0}^{T}\left(\mu_{m_t} - \hat{\mu}_{m_t}(t)\right) \tag{3.1}$$

where m_t denotes the selected expert at time t.

The estimation error is the right term of equation 3.1. It is minimized using contextual bandit algorithms that learn the dependency between the contexts, the rewards, and the actions. By performing a selection within a set of several contextual bandit algorithms, the learning experts selection task reduces the approximation error (left term

of equation 3.1).

3.1.3 Selection of Learning Experts as a Learn Then Explore and Exploit approach

A first approach to handle selection of learning algorithms is to use a Learn Then Explore and Exploit approach (LTEE):

- M contextual bandit problems are allocated and solved during the learning phase,
- one multi-armed bandit problem is allocated and optimizes the choice of experts during the exploration and exploitation phases.

Learn Then Explore and Exploit

Learning Phase:

In order to obtain an unbiased learning of each expert, during the learning phase the actions are played sequentially. To ensure that each expert has ended its learning, we set the size of the learning phase as the maximum sample complexity of experts. We consider here bandit algorithms, where an upper bound of the sample complexity is known such as BANDIT FOREST [Féraud et al., 2016] or linear bandits [Soare et al., 2014].

Definition 18. *The sample-complexity n_m of the expert m is the number of samples in $D_{x,y}$ needed to find π^*_m with a probability $1-\delta$.*

Proposition 1. *Let $[M]$ be a set of M experts able to learn from the uniform sampling of $D_{x,y}$. After sampling n times in $D_{x,y}$, the mean reward of each expert $m \in [M]$ is μ^*_m with a probability $1 - M\delta$, with a sample complexity $n = \max_m n_m$.*

Proof. Using the sample-complexity definition, the proof is a direct application of the union bound. □

Proposition 1 is now used to set the size of the learning phase.

Exploration and Exploitation Phase:

At the end of the Exploration phase, experts have learned and their performances can be considered as stationary. Then, any bandit algorithm such as UCB [Auer et al., 2002a] or SUCCESSIVE ELIMINATION [Even-Dar et al., 2006] can handle the usual trade-off between exploration and exploitation. We choose SUCCESSIVE ELIMINATION algorithm [Even-Dar et al., 2006] to provide a consistent end-to-end analysis of the selection of learning expert problem.

SUCCESSIVE ELIMINATION plays each remaining expert sequentially. The empirical mean of each expert $\hat{\mu}_m(t)$ after the time-step n is maintained through the run using past observations:

$$\hat{\mu}_m(t) = \frac{1}{t-n} \sum_{i=n+1}^{t} y_{\pi_{m_i}(x_i)}(i) \mathbb{1}_{m_i=m}.$$

When the gap between the empirical mean rewards of an expert m and the estimated best expert is high enough, then the expert is considered as suboptimal and removed from the set.

Algorithm 1 LTEE

input: $\delta \in (0, 1]$, $\epsilon \in [0, 1)$

output: an ϵ-approximation of the best expert

For $t := 0, ..., n$

Play sequentially an action $k_t \in A$

Update each expert $\pi_{m,t}$ in S with the tuple $(x_t, k_t, y_{k_t}(t))$

End for

$S_t := S$, $r := 0$

While $t \leq T$

For each $m \in S_t$

Play the action $k_t := \pi_{m,t}(x_t)$

Update $\hat{\mu}_m(t+1)$

$t := t + 1$ [1]

End for

$r := r + 1$

Remove from S_t all m such as:

$$\max_{m \in S_t} \hat{\mu}_m(t) - \hat{\mu}_m(t) + \epsilon \geq 2\sqrt{\log(4r^2 M/\delta)/2r}$$

End while

Theorem 7. *For $M > 0$, and $\delta > 0$, and $\epsilon = 0$, the sample complexity of* LTEE *is upper bounded by:*

$$O\left(\sum_{m=1}^{M} \frac{1}{\Delta_m^2} \log \frac{M}{\delta \Delta_m} + n\right),$$

where $\Delta_m = \mu_{m^*} - \mu_m$, and n the sample complexity needed to learn the set of experts.

Corollary 1. *The pseudo regret of* LTEE *is upper bounded by:*

$$O\left(\frac{M}{\Delta} \log \frac{MT}{\Delta} + n\Delta\right),$$

[1] For the sake of clarity, the condition $t \leq T$ is tested only before the round-robin phase. To ensure the end of the algorithm at T, another test is necessary at this mark.

The proofs of Theorem 7 and Corollary 1 are provided in section 3.1.7.

Theorem 8. *There exists a distribution $D_{x,y}$ such that any algorithm that learns M experts and then finds the best has a sample-complexity of at least:*

$$\Omega\left(\frac{M}{\Delta^2}\log\frac{1}{\delta}+\mathcal{N}\right),$$

where δ the probability of finding the optimal expert within the set of experts and $\mathcal{N}$ the lower-bound of the sample complexity needed to learn the experts.

The proof of Theorem 8 is provided in section 3.1.7. The LTEE algorithm is optimal up to logarithmic factors when the algorithm used to learn the experts has an optimal sample complexity, i.e. when $n = O(\mathcal{N})$.

3.1.4 Selection of Learning Experts as an Adversarial Problem with Budget

In previous approach, the experts are learned at the beginning of the run and then selected when their performance becomes stationary. LTEE algorithm obtains strong theoretical results and can be very efficient in practice when the sample complexities needed to learn the experts are close to each other. When one algorithm needs a large sample complexity and the others do not, LTEE algorithm spends a lot of steps playing sequentially the actions to learn the last expert. In this section, we present a new algorithm Learn, Explore and Exploit (LEE), which handles the parallel learning of experts, and which explores and exploits at the same time the experts. The exploration and exploitation of learning experts is modeled by a new problem setting, which we called *adversarial bandit with budget*. We describe this setting in the next section. We will analyze EXP3 for this setting. Then, we will explain the methodology used to learn experts in parallel and we propose a new algorithm, RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET, based on a randomized version of SUCCESSIVE ELIMINATION. Finally, we will present LEE, which uses RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET and an unbiased estimation of rewards to simultaneously learn, explore, and exploit experts.

Setting of Adversarial Bandit with Budget

Let $S = 1, ..., M$ be a set of experts. The reward $y_m(t) \in [0, 1]$, obtained by the player after selecting the expert m, is drawn from a distribution $D_m(t)$ of mean $\mu_m(t) \in [0, \mu_m]$, with $\mu_m \in [0, 1]$. The values of $\mu_m(t)$ are arbitrary chosen by an oblivious adversary before the beginning of the run. The adversary is constrained by a budget B such as $\forall m$, $\sum_{t=1}^{T} \mu_m - \mu_m(t) \leq B$. We abstract the learning experts selection problem by considering steps where $\mu_m(t) \neq \mu_m$ as chosen by an adversary with a fixed budget. This budget constraint is the main innovation compared to *contaminated rewards* proposed in [Seldin and Slivkins, 2014]. Notice that unlike *contaminated rewards* this

setting models different behaviors of learning curves: an high amount of the contaminated budget can be spent in a short period to model quick learner, or the budget can be spent equally over iterations to model slow learner.

EXP3 for Adversarial Bandit with Budget

When the knowledge of the budget B is not available and the time horizon T is known, EXP3 can be successfully applied to *adversarial bandit with budget* (see Theorem 9).

Theorem 9. *For any $M > 0$, the pseudo-regret of* EXP3 *for learning experts selection run with input parameter $\gamma = \sqrt{\frac{M \log(M)}{(e-1)T}}$ is:*

$$R(T) \leq 2\sqrt{(e-1)MT\log(M)} + B. \tag{3.2}$$

The proof of Theorem 9 is provided in section 3.1.7.

Unbiased Learning of Experts

In order to update each expert m in parallel with any tuple $(x_t, k_t, y_{k_t}(t))$, where k_t is chosen by another randomized policy, the observed reward $y_{k_t}(t)$ is replaced by an unbiased estimator of y_{k_t}:

$$r_{k_t}(t) = y_{k_t}(t)/P(k_t)(t).$$

Indeed, at any time t we have:

$$\mathbb{E}[r_{k_t}] = P(k_t).y_{k_t}/P(k_t) = y_{k_t}.$$

This approach, called *Inverse Propensity Scoring* [Horvitz and Thompson, 1952], can be used to evaluate the gain of policies only when no action has a zero probability (see Theorem 1 in [Langford and Strehl, 2008]). Lemma 1 shows that using LEE (see Algorithm 3), the probability of any remaining actions for any experts cannot be equal to zero.

Lemma 1. *When* LEE *algorithm uses randomized experts, which draw actions from their set of remaining actions $A_{m,t}$ according to an uniform distribution, the probability to draw any remaining action k is $P(k) \geq \frac{1}{MK}$.*

The proof of Lemma 1 is provided in section 3.1.7.

Randomized Successive Elimination with Budget

In this section, we present an algorithm for the best expert identification problem which is able to perform on distributions of large support and reward contaminated by an adversary with a budget B (see Algorithm 2). This algorithm

serves two purposes, firstly as an algorithm for selecting learning experts and secondly as an elementary block to build RANDOMIZED BANDIT FORESTS.

We introduce three significant differences: choices of experts are randomized, rewards are unbiased by the probability of playing the expert and the contaminated budget is taken into account when eliminating experts. Notice that this formulation is equivalent to a full information problem where the played expert has a reward of $|S_t|.y_m(t)$ and other experts a reward of zero.

Algorithm 2 RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET

input: $\delta \in (0,1]$, $\epsilon \in [0,1)$, $B \geq 0$
output: an ϵ-approximation of the best expert
$S_1 := S$, $\forall m$, $\hat{\mu}_m(0) := 0$, $Z(0) := |S|^2$, $t := 1$
While $|S_t| > 1$
$Z(t) := Z(t-1) + |S_t|^2$
Sample an expert $m_t \sim 1/|S_t|$
For each $m \in S_t$ **do**
$\hat{\mu}_m(t) := \frac{t-1}{t}\hat{\mu}_m(t-1) + \frac{|S_t|.y_m(t)}{t}\mathbb{1}_{\{m=m_t\}}$
$t := t+1$
End for
Remove from S_t all m such as:

$$\max_{m \in S_t} \hat{\mu}_m(t) - \hat{\mu}_m(t) + \epsilon \geq B/t + 2\sqrt{\frac{Z(t)}{2t^2}\log\left(\frac{4Mt^2}{\delta}\right)}$$

End while

When $B = 0$, RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET can be applied to best action identification problem.

Theorem 10. *For $M > 0$, and $\delta > 0$, and $\epsilon = 0$, the sample-complexity of* RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET *needed to find the best expert is upper bounded by:*

$$O\left(\sum_{m=1}^{M} \frac{M^2}{\Delta_m^2}\left(\log(\frac{M}{\delta\Delta_m}) + B\right)\right),$$

where $\Delta_m = \mu_{m^*} - \mu_m$.

This upper-bound, including a factor M^3, may seem high but is unavoidable when dealing with rewards debiased by the probability of playing the experts. We provide a lower-bound on the best expert identification problem with full information and reward distributions with a support of $[0, M]$, instead of $[0, 1]$ for the case where $B = 0$.

Theorem 11. *For $B = 0$ and $\epsilon = 0$, there exists a distribution D_Y such that any algorithm that observes at each step all experts' rewards $0 \leq Y_m(t) \leq M$ and then finds the best one has a sample-complexity of at least:*

$$\Omega\left(\frac{M^2}{\Delta^2}\log\frac{1}{\delta}\right),$$

where Δ is the difference of mean rewards between the two best experts.

This lower-bound shows the optimality of RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET up to a logarithmic factor when the support of the reward distribution is $[0, M]$ for $B = 0$.

Corollary 2. *For $M > 0$, $\delta > 0$, and $\epsilon = 0$, the expected regret of* RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET *is upper bounded by:*

$$O\left(\frac{M^3}{\Delta}\left(\log\frac{MT}{\Delta} + B\right)\right),$$

where $\Delta = \min_{m\in[M]} \Delta_m$

The proofs of Theorem 10, Theorem 11, and Corollary 2 are provided in section 3.1.7.

Algorithm 3 is a direct application of RANDOMIZED SUCCESSIVE ELIMINATION WITH BUDGET for Learning, Exploring, and Exploiting M experts.

Algorithm 3 LEARN, EXPLORE AND EXPLOIT (LEE)

input: $\delta \in (0, 1]$, $\epsilon \in [0, 1)$, $B \geq 0$
output: an ϵ-approximation of the best expert
$S_1 := S$, $\forall m$, $\hat{\mu}_m(0) := 0$, $Z(0) := |S|^2$, $t := 1$
While $t \leq T$
$Z(t) := Z(t-1) + |S_t|^2$
Sample an expert $m_t \sim 1/|S_t|$
Sample an action $k_t \sim A_{m,t}$
$P_{k_t} := 0$
For each $m \in S_t$ **do**
$P_{k_t} := P_{k_t} + \frac{1}{|A_{m,t}|\cdot|S_t|}$
End for
For each $m \in S_t$ **do**
$\hat{\mu}_m(t) := \frac{t-1}{t}\hat{\mu}_m(t-1) + \frac{|S_t|.y_{k_t}(t)}{t}\mathbb{1}_{\{m=m_t\}}$
Update the expert m with the tuple $\left(x_t, k_t, \frac{y_{k_t}(t)}{P_{k_t}}\right)$
$t := t + 1$
End for
Remove from S_t all m such as:

$$\max_{m\in S_t} \hat{\mu}_m(t) - \hat{\mu}_m(t) + \epsilon \geq B/t + 2\sqrt{\frac{Z(t)}{2t^2}\log\left(\frac{4Mt^2}{\delta}\right)}.$$

End while

3.1.5 Application of the Selection of Learning Experts Methodology to Bandit Forests

In this section, we illustrate the application of the LTEE and the LEE approaches, using BANDIT FORESTS [Féraud et al., 2016] as experts and illustrate their efficiency with numerical simulations.

Parametrization of the experts' learning algorithms have an high incidence on the final experts' performances. For

instance, on the *Forest Cover Type* dataset from the *UCI Machine Learning Repository*, a poorly parametrized expert may achieve a final classification rate of 45% whereas the expert with the most effective parametrization within the pool achieves a classification rate of 65.4%. As figure 3.3 shows, if the parameters n or B are underestimated, LEE and LTEE do not find the best expert in the set. Indeed, with $n = 0$ or $B = 0$, LTEE and LEE are equivalent to a stationary multi-armed bandit algorithm.

On a wide range of parameter values, LTEE and LEE find the best parametrization and achieve the classification rate of 65.4%, the same than the optimal forest, but at the cost of a slightly higher cumulative regret. In contrast, EXP3.S also finds the best parametrization but suffers from the constant exploration of the others experts inherent to this algorithm. While an underestimation of n or B can prevent the convergence of the algorithms to the best expert, an overestimation has a low impact on the cumulative regret. The practical use of LTEE and LEE, for which the optimal values of n and B are unknown, makes appear a trade-off between the better performances of LTEE at its point of best parametrization and the low sensitivity of LEE to an overestimation of B. Indeed, LTEE outperforms LEE only on a small range of values and shows an high degradation of performances when the parameter n is overestimated (LTEE is outperformed by EXP3.S when $n = 2 \times 10^6$) .

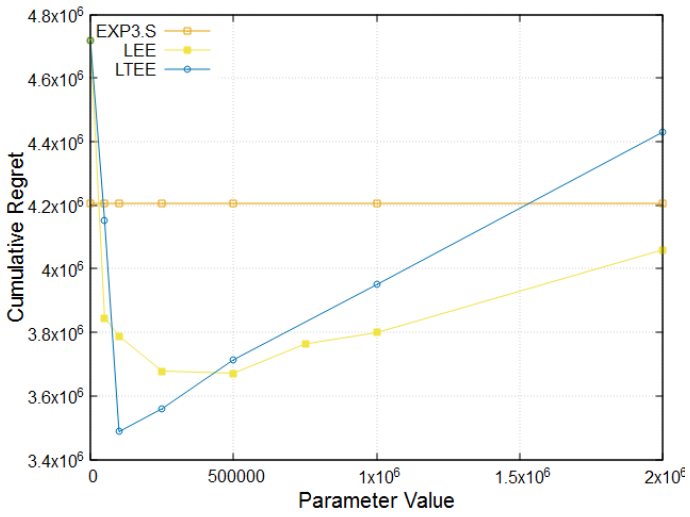


Figure 3.3: The cumulative regrets of EXP3.S, LTEE and LEE initialized with different parameters (respectively n and B) on Forest Cover Type.

Table 3.1: Forest Cover Type dataset where each continuous variable is recoded using *equal frequencies* into 5 binary variables, and each categorical variable is recoded into disjunctive binary variables. The 7 targeted classes are used as the set of actions. Cumulative regrets are given for $n = 100000$ and $B = 500000$. Classification rates are computed on the 100000 last contexts.

Algorithm	Regret	Classification Rate
Forest Cover Type, Target: Cover Type (7 classes)		
LTEE	$3.56\ 10^6 \pm 10^6$	65.4%
LEE	$3.68\ 10^6 \pm 2.10^6$	65.4%
EXP3.S	$4.21\ 10^6 \pm 5.10^5$	62.9%
OPTIMAL FOREST	$3.55\ 10^6 \pm 5.10^4$	65.4%

3.1.6 Conclusion

In this paper, we reduced the selection of learning experts to stochastic MAB problems. Taking advantage of this reduction, we proposed the LTEE algorithm. We showed that the proposed algorithm is optimal up to logarithmic factors. Despite this strong theoretical result, when the sample complexities of experts are not close to each other, in practice LTEE can spend a lot of steps playing sequentially the actions. We proposed a second algorithm LEE based on an unbiased estimation of rewards to share observations between experts and on a new problem setting, which we called *adversarial with budget*. We proposed a randomized variant of SUCCESSIVE ELIMINATION taking advantage of the proposed setting. Even if the obtained upper bound of the sample complexity is not as tight as the one obtained by LTEE, we experimentally demonstrated the efficiency of LEE. Further works may involve the analysis of the lower-bound of the *adversary bandit problem with budget*. An adaptation of these algorithms to handle switches in the distribution generating contexts and rewards also could be considered.

3.1.7 Proofs

Proof of Theorem 7

Proof. The sample-complexity of LTEE is upper bounded by the sum of the sample complexity of SUCCESSIVE ELIMINATION (see Theorem 8 in [Even-Dar et al., 2006]) and n (see Proposition 1). □

Proof of Corollary 1

Proof. For each time step of the exploration phase (i.e. $t \leq t^*$), the instantaneous pseudo-regret is $\Delta_{m_t} = \mu_{m^*} - \mu_{m_t}$. Applying Theorem 7, it exists $C_1 > 0$ such that:

$$R(T) \leq C_1 \sum_{m=1}^{M} \frac{1}{\Delta_m^2} \left(\log(\frac{M}{\delta \Delta_m}) \right) \Delta_m + (T - t^*) \left[1.\mathbb{P}(m_t \neq m^*) + 0.\mathbb{P}(m_t = m^*) \right] .$$
$$\leq C_1 \sum_{m=1}^{M} \frac{1}{\Delta_m} \left(\log(\frac{M}{\delta \Delta_m}) \right) + T\delta.$$

If we choose $\delta = \frac{1}{T}$, we obtain:

$$R(T) \leq O\left(\frac{M}{\Delta} \left(\log(\frac{MT}{\Delta}) \right) \right).$$

□

Proof of Theorem 8

Proof. The lower bound of the sample-complexity is the sum of the lower bound of best arm identification problem (see Theorem 1 in [Mannor and Tsitsiklis, 2004]) and the lower bound $\mathcal{N}$ of learning experts. Indeed, in worst case all the experts end their learning phase at the same time, and no observation coming from the learning phase can be used to evaluate the performances of experts. □

Proof of Theorem 9

Proof. The proof is trivial by adding B to both sides of the upper-bound on the cumulative regret of EXP3 provided in [Auer et al., 2002b]:

$$\mathbb{E}\left[R(T) - B\right] + B \leq 2\sqrt{(e-1)MT\log(M)} + B\,.$$

□

Proof of Lemma 1

Proof. Let S_t be the set of remaining experts at time t and $A_{m,t}$ be the set of remaining actions of the expert m. Let A_t bet the set of remaining actions, $A_t = \bigcup_m A_{m,t}$. In worst case, there exists an action $k \in A_t$ which is played by only one expert. Each expert has a probability $1/|S_t|$ to be drawn, which does not depend on the action. The probability that the action k be drawn is:

$$P(k) = \sum_m P(k|\pi_m).P(\pi_m) = \frac{1}{MK}.\sum_m \frac{K}{|A_{m,t}|} \geq \frac{1}{MK}$$

□

Proof of Theorem 10

Proof. From Hoeffding's inequality, at time t for any m we have:

$$P\left(|\hat{\mu}_m - \mathbb{E}[\hat{\mu}_m]| \geq \epsilon_t\right) \leq 2\exp\left(-\frac{2\epsilon_t^2 t^2}{\sum_{i=1}^{t}|S_t|^2}\right),$$

where $\mathbb{E}$ denotes the expectation with respect to the joint distribution $D_{y,m}$, where $p_m(t) = 1/|S_t|$.

We upper-bound $\sum_{i=1}^{t}|S_t|^2$ by tM^2.

By setting $\epsilon_t = \sqrt{\frac{M^2}{2t}\log\left(\frac{4Mt^2}{\delta}\right)}$, we have:

$$P\left(|\hat{\mu}_m - \mathbb{E}[\hat{\mu}_m]| \geq \epsilon_t\right) \leq 2\exp\left(\frac{-2\sqrt{\frac{M^2}{2t}\log\left(\frac{4Mt^2}{\delta}\right)}^2 t}{M^2}\right) = \frac{\delta}{2Mt^2}.$$

Using Hoeffding's inequality on each time-step t, applying the union bound and then $\sum 1/t^2 = \pi^2/6$, the following inequality holds for any time t with a probability at least $1 - \frac{\delta\pi^2}{12M}$:

$$\hat{\mu}_m - \epsilon_t \leq \mathbb{E}[\hat{\mu}_m] \leq \hat{\mu}_m + \epsilon_t. \tag{3.3}$$

An expert m' remains in the set S as long as for each $m \in S - \{m'\}$:

$$\hat{\mu}_m - \epsilon_t < \hat{\mu}_{m'} + \frac{B}{t} + \epsilon_t. \tag{3.4}$$

Using (3.3) in (3.4):

$$\mathbb{E}[\hat{\mu}_m] - 2\epsilon_t < \mathbb{E}[\hat{\mu}_{m'}] + \frac{B}{t} + 2\epsilon_t. \tag{3.5}$$

For each m we have:

$$\hat{\mu}_m = \frac{1}{t}\sum_{i=0}^{t}\frac{y_m(i)}{p_m(i)}\{m = m_i\}, \text{ where } p_m(i) = \frac{1}{|S_t|}.$$

Taking the expectation with respect to the reward distribution D_y we have:

$$\mathbb{E}_{D_y}[\hat{\mu}_m] = \frac{1}{t}\sum_{i=0}^{t}\frac{\mu_m - \mu_m + \mu_m(i)}{p_m(i)}\{m = m_i\},$$

$$\mathbb{E}_{D_y}[\hat{\mu}_m] = \frac{1}{t}\sum_{i=0}^{t}\left(\frac{\mu_m}{p_m(i)} - \frac{\mu_m - \mu_m(i)}{p_m(i)}\right)\{m = m_i\}. \tag{3.6}$$

Taking the expectation of both sides of equation (3.6) with respect to the distribution of experts $\langle m_1, ..., m_T\rangle$, with

$p_m(t) = 1/|S_t|$ and using $B \geq 0$ we have:

$$\mu_m - \frac{B}{t} \leq \mathbb{E}[\hat{\mu}_m] \leq \mu_m \,. \tag{3.7}$$

If (3.5) holds for m and m' by using (3.7) :

$$\mu_m - 2\epsilon_t - \frac{B}{t} \leq \mu_{m'} + 2\epsilon_t + \frac{B}{t} \,.$$

$$\Delta_{m,m'} < 4\epsilon_t + \frac{2B}{t} \,. \tag{3.8}$$

Replacing the value of ϵ_t in (3.8) and taking the square:

$$\Delta^2_{m,m'} < \frac{8M^2}{t} \log(\frac{4Mt^2}{\delta}) + \frac{4B^2}{t^2} \,. \tag{3.9}$$

For $m' = m^*$ and $m \neq m^*$ (3.9) is always true, involving that the optimal expert will always remain in the set with high probability for any t. Let $\Delta_{m'} = \mu_{m^*} - \mu_{m'}$. We have when $t \geq B$:

$$\Delta^2_{m'} < \frac{8M^2}{t} \log(\frac{4Mt^2}{\delta}) + \frac{4B}{t} \,. \tag{3.10}$$

The expert m' is or has been eliminated if inequality (3.10) is false.

We denote

$$C_1(t) = \frac{8M^2}{t} \log(\frac{4Mt^2}{\delta}) \text{ and } C_2(t) = \frac{4B}{t}.$$

Let

$$t_1^* = \frac{64^2}{\Delta^2_{m'}} M^2 \log\left(\frac{16M}{\delta\Delta_{m'}}\right) \,.$$

We denote For $t = t_1^*$,

$$C_1(t_1^*) = \frac{8\Delta^2_{m'}}{64^2 \log\frac{16M}{\delta\Delta_{m'}}} \left(\log\frac{4M}{\delta} + 4\log\frac{64M}{\Delta_{m'}} + 2\log\log\frac{16M}{\delta\Delta_{m'}}\right) \,,$$

$$C_1(t_1^*) \leq \frac{8\Delta^2_{m'}}{64^2 \log\frac{16M}{\delta\Delta_{m'}}} \left(8\log\frac{16M}{\delta\Delta_{m'}} + 24\log 2 + 2\log\log\frac{16M}{\delta\Delta_{m'}}\right) \,.$$

For $X > 8$ we have

$$24\log 2 + 2\log\log X < 8\log X \,.$$

Hence, we have:

$$C_1(t_1^*) \leq \frac{8\Delta_{m'}^2}{64^2 \log \frac{16M}{\delta\Delta_{m'}}} \left(16 \log \frac{16M}{\delta\Delta_{m'}}\right) ,$$

$$C_1(t_1^*) \leq \frac{\Delta_{m'}^2}{512} . \tag{3.11}$$

As $C_1(t_1^*)$ is strictly decreasing with regard to t, (3.11) is true for all $t > t_1^*$. When $t > t_1^*$, it exists $C_2(t)$ such as:

$$\Delta^2 = C_1(t) + C_2(t) .$$

For invalidating (3.10) we need to find a value of $t > t_1^*$ for which:

$$t \geq \frac{4B}{C_2(t)} .$$

As $C_2(t) = \Delta^2 - C_1(t)$ we have $C_2(t) \geq \Delta_{m'}^2 - \frac{\Delta_{m'}^2}{512}$,

$$t \geq \frac{2048B}{511\Delta_{m'}^2} . \tag{3.12}$$

For $t = \frac{64^2}{\Delta_{m'}^2} M^2 \log\left(\frac{16M}{\delta\Delta_{m'}}\right) + \frac{5B}{\Delta_{m'}^2}$, (3.12) is true, invalidating (3.10) and involving the elimination of sub-optimal expert m' with a probability at least $1 - \delta/M$, concluding the proof. □

Proof of Corollary 2

Proof. For each time step of the exploration phase (i.e. $t \leq t^*$), the instantaneous pseudo-regret is $\Delta_{m_t} = \mu_{m^*} - \mu_{m_t}$. Applying Theorem 13, it exists $C_1 > 0$ such that:

$$\begin{aligned} R(T) &\leq C_1 \sum_{m=1}^{M} \frac{M^2}{\Delta_m^2} \left(\log(\frac{M}{\delta\Delta_m}) + B\right) \Delta_m + (T - t^*) \left[1.\mathbb{P}(m_t \neq m^*) + 0.\mathbb{P}(m_t = m^*)\right] . \\ &\leq C_1 \sum_{m=1}^{M} \frac{M^2}{\Delta_m} \left(\log(\frac{M}{\delta\Delta_m}) + B\right) + T\delta. \end{aligned}$$

If we choose $\delta = \frac{1}{T}$, we obtain:

$$R(T) \leq O\left(\frac{M^3}{\Delta} \left(\log(\frac{MT}{\Delta}) + B\right)\right) .$$

□

Proof of Theorem 11

Proof. Let $0 \leq Y_i(t) \leq M$ be a bounded random variable corresponding to the rewards of the expert i. By dividing $Y_i(t)$ by M we have:

$$0 \leq \frac{Y_i(t)}{M} \leq 1 \tag{3.13}$$

and

$$\mathbb{E}\left[\sum_{t=1}^{t^*} \frac{Y_i(t)}{M}\right] = \frac{\mu_i}{M}\,. \tag{3.14}$$

Let Θ be the sum of the binary random variables $\theta_1, ..., \theta_t, ...\theta_{t^*}$ where

$$\theta_t = \frac{Y_i(t)}{M} \geq \frac{Y_j(t)}{M}\,. \tag{3.15}$$

Let $p_{i,j}$ be the probability that the use of expert i leads to more rewards than the use of expert j. We have

$$p_{i,j} = \frac{1}{2} - \Delta_{i,j}, \text{where } \Delta_{i,j} = \frac{\mu_i - \mu_j}{M}\,. \tag{3.16}$$

Slud's inequality [Slud, 1977] states that when $p \leq 1/2$ and $t_k^* \leq x \leq t_k^*.(1-p)$, we have:

$$P(\Theta \geq x) \geq P\left(Z \geq \frac{x - t_k^*.p}{\sqrt{t_k^*.p(1-p)}}\right)\,, \tag{3.17}$$

where Z is a normal $\mathcal{N}(0,1)$ random variable. To choose the best expert between i and j, one needs to find the time t^* where $P(\Theta \geq t^*/2) \geq \delta$. To state the number of trials t^* needed to estimate Δ_{ij}, we recall and adapt the arguments developed in [Mousavi, 2010]. Using Slud's inequality (see equation 4.20), we have:

$$P(\Theta \geq t_k^*/2) \geq P\left(Z \geq \frac{t^*.\Delta_{ij}}{\sqrt{t^*.p_{ij}(1-p_{ij})}}\right)\,, \tag{3.18}$$

Then, we use the lower bound of the error function [Chu, 1955]:

$$P(Z \geq z) \geq 1 - \sqrt{1 - \exp\left(-\frac{z^2}{2}\right)}$$

Therefore, we have:

$$\begin{aligned} P(\Theta \geq t^*/2) &\geq 1 - \sqrt{1 - \exp\left(-\frac{t^*.\Delta_{ij}^2}{2p_{ij}(1-p_{ij})}\right)} \\ &\geq \frac{1}{2}\exp\left(-\frac{t^*.\Delta_{ij}^2}{p_{ij}}\right) \end{aligned}$$

As $p_{ij} = 1/2 - \Delta_{ij}$, we have:

$$\log \delta = \log \frac{1}{2} - \frac{t^*.\Delta_{ij}^2}{1/2 - \Delta_{ij}} \geq \log \frac{1}{2} - 2t^*.\Delta_{ij}^2$$

Hence, we have:

$$t^* = \Omega\left(\frac{1}{\Delta_{ij}^2} \log \frac{1}{\delta}\right)$$

In the worst case, $\min_j \Delta_{i,j} = \Delta_i = \mu_{i^*} - \mu_i$. Now $\Delta_i = \frac{\mu_{i^*} - \mu_i}{M}$. Then for eliminating an expert, the sample complexity is at least:

$$\Omega\left(\frac{M^2}{(\mu_{i^*} - \mu_i)^2} \log \frac{1}{\delta}\right) . \tag{3.19}$$

□

3.2 Non stationary stochastic Bandits

Abstract. We consider a variant of the stochastic multi-armed bandit with K arms where the rewards are not assumed to be identically distributed, but are generated by a non-stationary stochastic process. We first study the *unique best arm* setting when there exists one unique best arm. Second, we study the general *switching best arm* setting when a best arm switches at some unknown steps. For both settings, we target problem-dependent bounds, instead of the more conservative problem free bounds. We consider two classical problems: 1) Identify a best arm with high probability (best arm identification), for which the performance measure by the sample complexity (number of samples before finding a near optimal arm). To this end, we naturally extend the definition of sample complexity so that it makes sense in the switching best arm setting, which may be of independent interest. 2) Achieve the smallest cumulative regret (regret minimization) where the regret is measured with respect to the strategy pulling an arm with the best instantaneous mean at each step.

3.2.1 Introduction

The theoretical framework of the multi-armed bandit problem formalizes the fundamental exploration/exploitation dilemma that appears in decision making problems facing partial information. At a high level, a set of K arms is available to a player. At each turn, she has to choose one arm and receives a reward corresponding to the played arm, without knowing what would have been the received reward had she played another arm. The player faces the dilemma of *exploring*, that is playing an arm whose mean reward is loosely estimated in order to build a better estimate or *exploiting*, that is playing a seemingly best arm based on current mean estimates in order to maximize her cumulative reward. The accuracy of the player policy at time horizon T is typically measured in terms of *sample complexity* or of *regret*. The sample complexity is the number of plays required to find an approximation of the

best arm with high probability. In that case, the player can stop playing after identifying this arm. The regret is the difference between the cumulative rewards of the player and the one that could be acquired by a policy assumed to be optimal.

The **stochastic** multi-armed bandit problem assumes the rewards to be generated independently from stochastic distribution associated with each arm. Stochastic algorithms usually assume distributions to be constant over time like with the Thompson Sampling (TS) [Thompson, 1933], UCB [Auer et al., 2002a] or Successive Elimination (SE) [Even-Dar et al., 2006]. Under this assumption of *stationarity*, TS and UCB achieve optimal upper-bounds on the cumulative regret with logarithmic dependencies on T . SE algorithm achieves a near optimal sample complexity.

In the **adversarial** multi-armed bandit problem, rewards are chosen by an adversary. This formulation can model any form of non-stationarity. The EXP3 algorithm [Auer et al., 2002b, Neu, 2015] achieves an optimal regret of $O(\sqrt{T})$ against an oblivious opponent that chooses rewards before the beginning of the game, with respect to the best policy that pulls the same arm over the totality of the game. This weakness is partially overcame by EXP3.S [Auer et al., 2002b], a variant of EXP3, that forgets the past adding at each time step a proportion of the mean gain and achieves controlled regret with respect to policies that allow arm switches during the run.

The **switching bandit** problem introduces non-stationarity within the *stochastic* bandit problem by allowing means to change at some time-steps. As mean rewards stay stationary between those changes, this setting is also qualified as *piecewise-stationary*. *Discounted* UCB [Kocsis and Szepesvári, 2006b] and *sliding-window* UCB [Garivier and Moulines, 2011] are adaptations of UCB to the switching bandit problem and achieve a regret bound of $O(\sqrt{MT \log T})$, where $M - 1$ is the number of distribution changes. It is also worth citing META-EVE [Hartland and M., 2006] that associates UCB with a mean change detector, resetting the algorithm when a change is detected. While no analysis is provided, it has demonstrated strong empirical performances.

Stochastic and Adversarial. Several variants combining stochastic and adversarial rewards have been proposed by Seldin & Slivkins [Seldin and Slivkins, 2014] or Bubeck & Slivkins [Bubeck and Slivkins, 2012]. For instance, in the setting with *contaminated rewards*, rewards are mainly drawn from stationary distributions except for a minority of mean rewards chosen in advance by an adversary. In order to guarantee their proposed algorithm EXP3++ [Seldin and Slivkins, 2014] achieves logarithmic guarantees, the adversary is constrained in the sense it cannot lowered the gap between arms more than a factor $1/2$. They also proposed another variant called *adversarial with gap* [Seldin and Slivkins, 2014] which assumes the existence of a round after which an arm persists to be the best. These works are motivated by the desire to create generic algorithms able to perform bandit tasks with various reward types, stationary, adversary or mainly stationary. However, despite achieving good performances on a wide range

of problems, each one needs a specific parametrization (i.e. an instance of EXP3++ parametrized for stationary rewards may not perform well if rewards are chosen by an adversary).

Our contribution. We consider a generalization of the stationary stochastic, piecewise-stationary and adversarial bandit problems. In this formulation, rewards are drawn from stochastic distributions of arbitrary means defined before the beginning of the game. Our first contribution is for the unique best arm setting. We introduce a deceptively simple variant of the SUCCESSIVE ELIMINATION (SE) algorithm, called SUCCESSIVE ELIMINATION WITH RANDOMIZED ROUND-ROBIN (SER3) and we show that the seemingly minor modification – a randomized round-robin procedure – leads to a dramatic improvement of the performance over the original SE algorithm. We identify a notion of gap that generalizes the gap from stochastic bandits to the non-stationary case, and derive *gap-dependent* (also known as problem-dependent) sample complexity and regret bounds, instead of the more classical and less informative *problem-free* bounds. We show for instance in Theorem 12 and Corollary 3 that SER3 achieves a non-trivial problem dependent sample complexity scaling with Δ^{-2} and a cumulative regret in $O(K\log(TK/\Delta)/\Delta)$ after T steps, in situations where SE may even suffers from a linear regret, as supported by numerical experiments (see Section 3.2.5). This result positions, under some assumptions, SER3 as an alternative to EXP3 when the rewards are non-stationary.

Our second contribution is to manage best arm switches during the game. First, we extend the definition of the sample complexity in order to analyze the best arm identification algorithms when best arm switches during the game. SER4 takes advantages of the low regret of SER3 by resetting the reward estimators randomly during the game and then starting a new phase of optimization. Against an optimal policy with $N-1$ switches of the optimal arm (but arbitrarily many distribution switches), this new algorithm achieves an expected sample complexity of $O(\Delta^{-2}\sqrt{NK\delta^{-1}\log(K\delta^{-1})})$, with probability $1-\delta$, and an expected cumulative regret of $O(\Delta^{-1}\sqrt{NTK\log(TK)})$ after T time steps. A second algorithm for the non stationary stochastic multi-armed bandit with switches is an alternative to the passive approach used in SER4 (the random resets). Second, the algorithm EXP3.R takes advantage of the exploration factor of EXP3 to evaluate unbiased estimations of the mean rewards. Combined with a drift detector, this active approach resets the weights of EXP3 when a change of best arm is detected. We finally show that EXP3.R also obtains competitive problem-dependent regret minimization guarantees in $O\left(3NCK\sqrt{TK\log T}\right)$, where C depends on Δ.

We provide in Table 3.2 and 3.3 a brief summary of the existing results regarding the performance of a few algorithms, together with the contributions of this article, that are indicated in bold. In both tables, T is the time horizon, assumed to be know, K the number of arms, Δ is the gap, and δ is the probability of success of the algorithm. C is quantity similar to the gap, described in subsection 3.2.4. Finally, M is the number of breakpoints (the mean reward of an arm changes) and N the number of best arm switches.

Table 3.2: Overview of the different bandit algorithms for policies with unique best arm

Algorithms	Regret	Sample Complexity	Non Stationarity
State of the art			
UCB	$O\left(\Delta^{-1}K\log(T)\right)$	X	No
SE	$O\left(\Delta^{-1}K\log(TK/\Delta)\right)$	$O\left(\Delta^{-2}K\log(TK/\Delta)\right)$	No
EXP3	$O\left(\sqrt{KT\log K}\right)$	X	Yes
EXP3++	$O(\Delta^{-1}K\log^3 T)+\tilde{O}(\Delta^{-3})$	X	Yes
Our contribution			
SER3	$O\left(\Delta^{-1}K\log(TK/\Delta)\right)$	$O\left(\Delta^{-2}K\log(TK/\Delta)\right)$	Yes

Table 3.3: Overview of the different bandit algorithms for policies with switching best arm

Algorithms	Regret	Sample Complexity	Non Stationarity between breakpoints
State of the art			
SW-UCB	$O\left(\Delta^{-1}\sqrt{MT\log T}\right)$	X	No
EXP3.S	$O\left(\sqrt{NKT\log(KT)}\right)$	X	Yes
Our contributions			
SER4	$O\left(\Delta^{-1}\sqrt{NKT\log(KT)}\right)$	$O\left(\Delta^{-2}\sqrt{NK\delta^{-1}\log(K\delta^{-1})}\right)$	Yes
EXP3.R	$O\left(3NCK\sqrt{TK\log T}\right)$	X	Yes

3.2.2 Problem Setting

We consider a generalization of the stationary stochastic, piecewise-stationary and adversarial bandit problems where the adversary chooses before the beginning of the game a sequence of *distributions* instead of directly choosing a sequence of rewards. This formulation generalizes the adversarial setting since choosing arbitrarily a reward $y_k(t)$ is equivalent to drawing this reward from a distribution of mean $y_k(t)$ and a variance of zero. The stationary stochastic formulation of the bandit problem is a particular case, where the distributions do not change.

The problem

Let $[K] = 1, ..., K$ be a set of K arms. The reward $y_{k_t}(t) \in [0, 1]$ obtained by the player after playing the arm k_t is drawn from a distribution of mean $\mu_{k_t}(t) \in [0, 1]$. The instantaneous gap between arms k and k' at time t is:

$$\Delta_{k,k'}(t)\mu_k(t) - \mu_{k'}(t)\,. \tag{3.20}$$

The player competes against an optimal policy, assumed as optimal (per example, always playing the arm with the highest mean reward). Let $k^*(t)$ be the arm played by the optimal policy at time t.

The notion of sample complexity

In the literature [Kaufmann et al., 2016], the sample-complexity of an algorithm is the number of samples needed by this algorithm to find a policy achieving a specific level of performance with high probability. We denote $\delta \in (0, 1]$ the probability of failure. For instance, for the best arm identification in the stochastic stationary bandit (that is when $\forall k \forall t\,, \mu_k(t) = \mu_k(t + 1)$ and $k^*(t) = k^*(t + 1)$), the sample complexity is the number of sample needed to find, with a probability at least $1 - \delta$, the arm k^* with the maximum mean reward. Analysis in sample complexity are useful for situations where the knowledge of the optimal arm is needed to make one impact-full decision, for example to choose which one of several possible products to manufacture or for building hierarchical models of contextual bandits in a greedy way [Féraud et al., 2016], reducing the exploration space.

Definition 19 (Sample complexity)**.** *Let A be an algorithm. An arm k is epsilon optimal if $\mu_k \geq \mu^* - \epsilon$, with $\epsilon \in [0, 1]$. The sample-complexity of A performing a best arm identification task is the number of observations needed to find an ϵ-optimal arm with a probability of at least $1 - \delta$.*

The usual notion of sample complexity - the minimal number of observations required to find a near optimal arm with high probability - is well adapted to the case when there exists a unique best arm during all the game, but makes little sense in the general scenario when the best arm can change. Indeed, after a best arm change, a learning algorithm requires some time steps before recovering. Thus, we provide in section 3.2.4 a meaningful extension of the sample complexity definition to the *switching best arm* scenario. This extended notion of sample complexity now takes into account not only the number of time-steps required by the algorithm to identify a near optimal arm, but more generally the number of time steps required before recovering a near optimal arm after each change.

The notion of regret

When the decision process does not lead to one final decision, minimizing the sample complexity may not be an appropriate goal. Instead, we may want to maximize the cumulative gain obtained through the game which is

equivalent to minimize the difference between the choices of an optimal policy and those of the player. We call this difference, the regret. We define the *pseudo cumulative regret* as the difference of mean rewards between the arms chosen by the optimal policy and those chosen by the player.

Definition 20 (Pseudo cumulative regret)**.**

$$R(T) = \sum_{t=1}^{T} \mu_{k^*(t)}(t) - \mu_{k_t}(t)\,. \tag{3.21}$$

Usually, in the stochastic bandit setting, the distributions of rewards are stationary and the instantaneous gap $\Delta_{k,k'}(t) = \mu_k(t) - \mu_{k'}(t)$ is the same for all the time-steps.

There exists a non reciprocate relation between the minimization of the sample-complexity and the minimization of the pseudo cumulative regret. For instance, the algorithm UCB has an order optimal regret, but it does not minimize the sample-complexity. UCB will continue to play sub-optimal arms, but with a decreasing frequency as the number of plays increases. However, an algorithm with an optimal sample complexity, like MEDIAN ELIMINATION [Even-Dar et al., 2006], will also have an optimal pseudo cumulative regret (up to some constant factors). More details on the relation between both lower bounds can be found in [Bubeck and Cesa-Bianchi, 2012, Garivier et al., 2016].

Therefore, the algorithms presented in this paper slightly differ according to the quantity to minimize, the regret or the sample complexity. For instance, when the target is the regret minimization, after identifying the best arm, the algorithms continue to sample it whereas in the case of sample complexity minimization, the algorithms stop the sampling process when the best arm is identified. When best arm switches are considered, algorithms minimizing the sample complexity enter a waiting state after identifying the current best arm and do not sample the sequence for exploitation purposes (sampling the optimal arm still increases the sample complexity). However, they still have to parsimoniously collect samples for each actions in order to detect best arm changes and face a new trade-off between the rate of sampling and the time needed to find the new best arm after a switch.

3.2.3 Non-stationary Stochastic Multi-armed Bandit with Unique Best Arm.

In this section, we present the algorithm SUCCESSIVE ELIMINATION WITH RANDOMIZED ROUND-ROBIN (SER3, see algorithm 4), a randomized version of SUCCESSIVE ELIMINATION which tackles the best arm identification problem when rewards are non-stationary.

A modified Successive Elimination algorithm

We elaborate on several notions required to understand the behavior of the algorithm and to relax the constraint of stationarity.

The elimination mechanism.

The elimination mechanism was introduced by SUCCESSIVE ELIMINATION [Even-Dar et al., 2006]. Estimators of the rewards are built by sequentially sampling the arms. After $\tau_{\min}$ turns of round-robin, the elimination mechanism starts to occur. A lower-bound of the reward of the best empirical arm is computed and compared with an upper-bound of the reward of all other arms. If the lower-bound is higher than one of the upper-bounds, then the associated arm is eliminated and stop being considered by the algorithm. Processes of sampling and elimination are repeated until the elimination of all arms except one.

Algorithm 4 SUCCESSIVE ELIMINATION WITH RANDOMIZED ROUND-ROBIN (SER3)

input: $\delta \in (0, 0.5]$, $\epsilon \in [0, 1)$, $\tau_{\min} := \log \frac{K}{\delta}$
output: an ϵ-approximation of the best arm
$S_1 := [K]$, $\forall k$, $\hat{\mu}_k(0) := 0$, $t := 1$, $\tau := 1$
While $|S_\tau| > 1$
  Shuffle S_τ
  For each $k \in S_\tau$ **do**
    Play k
    $\hat{\mu}_k(\tau) := \frac{\tau-1}{\tau}\hat{\mu}_k(\tau-1) + \frac{y_k(t)}{\tau}$
    $t := t + 1$
  End for
  If $\tau \geq \tau_{\min}$
    Remove from $S_{\tau+1}$ all k such as:

$$\max_{k \in S} \hat{\mu}_k(\tau) - \hat{\mu}_k(\tau) + \epsilon \geq \sqrt{\frac{2}{\tau} \log\left(\frac{4K\tau^2}{\delta}\right)} \tag{3.22}$$

  End if
  If $|S_\tau| = 1$ **and** the algorithm performs a sample complexity minimization task
    Return the last element of S_τ
  End if
  $\tau := \tau + 1$
End while

Hoeffding inequality.

SUCCESSIVE ELIMINATION assumes that the rewards are drawn from stochastic distributions that are identical over time (rewards are identically distributed). However, the Hoeffding inequality used by this algorithm does not require stationarity and only requires independence. We remember the Hoeffding inequality:

Lemma 2 (Hoeffding inequality [Hoeffding, 1994])**.** *If $X_1, X_2, ..., X_\tau$ are τ independent random variables and $0 \leq X_i \leq 1$ for all $(i = 1, 2, ..., \tau)$, then for $\epsilon_\tau > 0$*

$$P\left(\left|\sum_{i=1}^{t} \frac{X_i}{\tau} - \mathbb{E}\left[\sum_{i=1}^{t} \frac{X_i}{\tau}\right]\right| \geq \epsilon_\tau\right) \leq 2\exp\left(-2\epsilon_\tau^2 \tau\right) .$$

Thus, we can use this inequality to calculate confidence bounds of empirical means computed with rewards drawn from non identical distributions.

Randomization of the Round-Robin.

We illustrate the need of randomization with an example tricking a deterministic algorithm (see table 3.4).

$\mu_k(t)$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$k=1$	0.6	1	0.6	1	0.6	1
$k=2$	0.4	0.8	0.4	0.8	0.4	0.8

Table 3.4: A sequence of mean rewards tricking a deterministic bandit algorithm.

The best arm seems to be $k = 1$ as $\mu_1(t)$ is greater than $\mu_2(t)$ at every time-step t. However, by sampling the arms with a deterministic policy playing sequentially $k = 1$ and then $k = 2$, after $t = 6$ the algorithm has only sampled rewards from a distribution of mean 0.6 for $k = 1$ and of mean 0.8 for $k = 2$. After enough time following this pattern, an elimination algorithm will eliminate the first arm. Our algorithm SER3 adds a shuffling of the arm set after each round-robin cycle to SUCCESSIVE ELIMINATION and avoids this behavior.

Uniqueness of the best arm.

The best arm identification task assumes a criteria identifying the best arm without ambiguity. We define the **optimal arm** as:

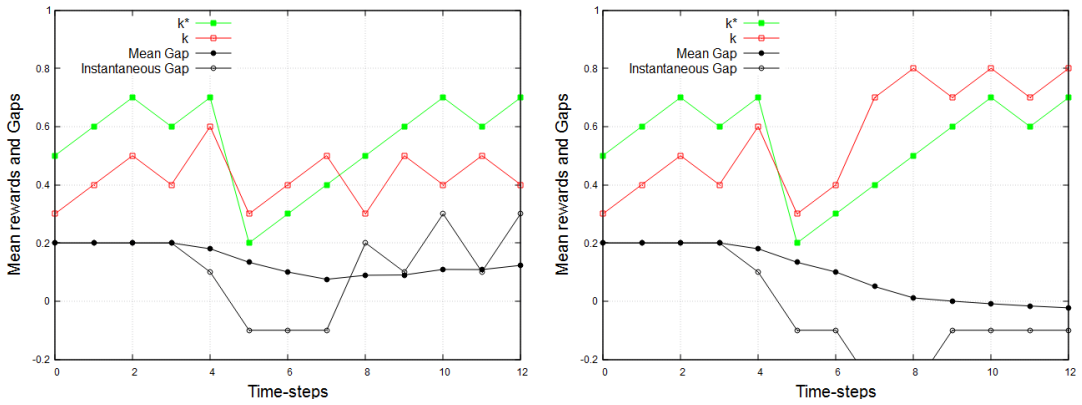
$$k^* = \arg\max_{k\in[K]} \sum_{t=1}^{T} \mu_k(t)\,. \tag{3.23}$$

As an efficient algorithm will find the best arm before the end of the run, we use assumption 1 to ensure its uniqueness at every time-step. First, we define some notations. A run of SER3 is a succession of round-robin. The set $[\tau] = \{(t_1, |S_1|), ..., (t_\tau, |S_\tau|)\}$ is a realization of SER3 and t_i is the time step when the round-robin i^{th} of size $|S_i|$ starts ($t_i = 1 + \sum_{j=1}^{i-1} |S_j|$). As arms are only eliminated, $|S_i| \geq |S_{i+1}|$. We denote $\mathbb{T}(\tau)$ the set containing all possible realizations of τ round-robin steps. Now, we can introduce assumption 1 that ensures the best arm is the same at any time-step.

Assumption 1 (Positive mean-gap)**.** *For any $k \in [K] - \{k^*\}$ and any $[\tau] \in \mathbb{T}(\tau)$ with $\tau \geq \tau_{\min}$, we have:*

$$\Delta_k^*\left([\tau]\right) = \frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_i|-1}\frac{\Delta_{k^*,k}(j)}{|S_i|} > 0\,. \tag{3.24}$$

Assumption 1 is trivially satisfied when distributions are stationary, is quite weak (see e.g. figure 3.4(b)) and can tolerate a large noise when τ is high. As the optimal arm must distinguish itself from others, instantaneous gaps are more constrained at the beginning of the game. It is quite similar to the assumption used by Seldin & Slivkins [**?**] to be able to achieve logarithmic expected regret on *moderately contaminated rewards*, i.e., the adversary does not lower the averaged gap too much. Another analogy can be done with the *adversarial with gap* setting [**?**], $\tau_{\min}$ representing the time needed for the optimal arm to accumulate enough rewards and to distinguish itself from the



(a) Assumption 1 is satisfied as the mean gap remains positive.

(b) Assumption 1 is not satisfied. This sequence involves a best arm switch as the mean gap become non positive.

Figure 3.4: Two examples of sequence of mean rewards.

suboptimal arms.

Figure 3.4(a) illustrates assumption 1. In this example the mean of the optimal arm k^* is lower than the second one on time-steps $t \in \{5, 6, 7\}$. Thus, even if the instantaneous gap is negative during these time-steps, the mean gap $\Delta_k^*([\tau])$ stays positive. The parameter $\tau_{\min}$ protects the algorithm from local noise at the initialization of the algorithm. In order to ease the reading of the results in the next sections, we here assume $\tau_{\min} = \log \frac{K}{\delta}$.

Assumption 1 can be seen as a sanity-check assumption ensuring that the best-arm identification problem indeed makes sense. In section 3.2.4, we consider the more general switching bandit problem. In this case, assumption 1 may not be verified (see figure 3.4(b)), and is naturally extended by dividing the game in segments wherein assumption 1 is satisfied.

Analysis

All theoretical results are provided for $\epsilon = 0$ and therefore accept only k^* as the optimal arm.

Theorem 12 (Sample-complexity of SER3)**.** *For $K \geq 2$, $\delta \in (0, 0.5]$, and $\tau_{\min} = \log \frac{K}{\delta}$, the sample-complexity of* SER3 *is upper bounded by:*

$$O\left(\frac{K}{\Delta^2}\log(\frac{K}{\delta\Delta})\right)$$

where $\Delta = \min_{[\tau],k} \frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{t=t_i}^{t_i+|S_i|-1}\frac{\Delta_{k^*,k}(t)}{|S_i|}$.

The proof is given in section 3.2.7.

Guarantee on the sample complexity can be transposed in guarantee on the pseudo-regret. In that case, when only one arm remains in the set, the player continues to play this last arm until the end of the game.

Corollary 3 (Expected pseudo-regret of SER3)**.** *For $K \geq 2$, and $\delta = 1/T$, and $\tau_{\min} = \log(KT)$, the pseudo-regret of* SER3 *is upper bounded by:*

$$\min\left(O\left(\frac{K-1}{\Delta}\log(\frac{KT}{\Delta})\right), O\left(\sqrt{TK\log\frac{T}{K}}\right)\right)$$

The proof is given in section 3.2.7.

These guarantees are the same as the original SUCCESSIVE ELIMINATION performed with a deterministic round-robin on arms with stationary rewards. Indeed, when reward distributions are stationary, we have for all t and all $[\tau]$:

$$\frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{t=t_i}^{t_i+|S_i|-1}\frac{\Delta_{k^*,k}(t)}{|S_i|} = \Delta_{k^*,k}(t) = \Delta_{k^*,k}(t+1)\,. \tag{3.25}$$

However, in a non-stationary environment satisfying assumption 1 SUCCESSIVE ELIMINATION will eliminate the optimal arm if the adversary knows the order of its round-robin before the beginning of the run and exploits this knowledge against the learner, thus resulting in a linear cumulative regret. Our modification of the SE algorithm allows SER3 to perform on *near adversarial* sequence of reward while achieving a gap dependent logarithmic pseudo cumulative regret.

Remark 12. *These logarithmic guarantees result from assumption 1 that allows to stop the exploration of eliminated arms. They do not contradict the lower bound for non-stationary bandit whose scaling is in $\Omega(\sqrt{T})$ [Garivier and Moulines, 2011] as it is due to the cost of the constant exploration for the case where the best arm changes.*

Non-stationary Stochastic Multi-armed Bandit with Budget

We study the case when the sequence from which the rewards are drawn does not satisfy assumption 1.

The sequence of mean rewards is build by the adversary in two steps. First, the adversary choose the mean rewards $\mu_k(1), ..., \mu_k(T)$ associated with each arm in such a way that assumption 1 is satisfied. The adversary can then apply a malus $b_k(t) \in [0, \mu_k(t)]$ to each mean reward to obtain the final sequence. The mean reward of the arm k at time t is $\mu_k(t) - b_k(t)$. The budget spent by the adversary for the arm k is $B_k = \sum_{t=1}^{T} b_k(t)$. We denote $B \geq \arg\max_k B_k$ the upper-bound on the budget of the adversary.

The algorithm SER3 can be modified to perform a best arm identification task when assumption 1 is not satisfied but B is known. To achieve that, we replace the condition of elimination (Inequality (3.22) in Algorithm 4) is replaced

by the following:

$$\hat{\mu}_{\max}(\tau) - \hat{\mu}_k(\tau) + \epsilon \geq \frac{B}{\tau} + 2\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}$$

This new algorithm is called SUCCESSIVE ELIMINATION WITH ROUND ROBIN RANDOMIZED AND BUDGET (SER3.B).

Theorem 13. *For $K \geq 2$, $\delta \in (0, 0.5]$, and $\tau_{\min} = \log \frac{K}{\delta}$, the sample complexity of* SER3.B *is upper-bounded by:*

$$O\left(\frac{K}{\Delta^2}\left(\log\frac{K}{\delta\Delta} + B\right)\right)$$

where $\Delta = \min_{[\tau],k} \frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{t=t_i}^{t_i+|S_i|-1}\frac{\Delta_{k^*,k}(t)}{|S_i|}$.

The proof is given in section 3.2.7.

3.2.4 Non-stationary Stochastic Multi-armed Bandit with Best Arm Switches

The switching bandit problem has been proposed in [Garivier and Moulines, 2011] and assumes means to be stationary between switches. In particular, the algorithm SW-UCB is built on this assumption and is a modification of UCB using only the rewards obtained inside a sliding window. In our setting, we allow mean rewards to change at every time-steps and consider that a best arm switch occurs when the arm with the highest mean change. This setting provides an alternative to the adversarial bandit with budget, when B is very high or unknown.

The **optimal policy** is the sequence of couples (optimal arm, time when the switch occurred):

$$\{(k_1^*, 1), ..., (k_N^*, T_N)\}, \tag{3.26}$$

with $k_n^* \neq k_{n+1}^*$ and $\Delta_{k_n^*,k}(t) > 0$ for any $k \in [K] - \{k_n^*\}$ and any $t \in [T_n, T_{n+1})$. The optimal policy starts playing the arm k_n^* at the time-step T_n. Time-steps T_n when switches occur are unknown to the player.

Successive Elimination with Randomized Round-Robin and Resets (SER4)

The Definition 19 of the sample complexity is not adapted to the switching bandit problem. Indeed this definition is used to measure the number of observations needed by an algorithm to find one unique best arm. When the best arm changes during the game, this definition is too limiting. In subsection 3.2.4 we introduce a generalization of the sample complexity for the case of switching policies.

The sample complexity of the best arm identification problem with switches.

A cost associated is added to the usual sample complexity. This cost is equal to the number of iterations after a switch during which the player does not know the optimal arm and does not sample.

Definition 21 (Sample complexity with switches)**.** *Let A be an algorithm. The sample-complexity of A performing a best arms identification task for a segmentation* $\{T_n\}_{n=1..N}$ *of* $[1:T]$*, with* $T_1 = 1 < T_2 < \cdots < T_N < T$*, is:*

$$\sum_{n=1}^{N} \sum_{t=T_n}^{T_{n+1}-1} \max(s(t), 1_{k_t \neq k_n^*}) \,, \tag{3.27}$$

where $s(t)$ *is a binary variable equal to* 1 *if and only if the time-step* t *is used by the sampling process of A,* k_t *is the arm identified as optimal by A at time t,* k_n^* *is the optimal arm over the segment* n *and* $T_{N+1} = T + 1$*.*

In order to clarify definition 21, we detail the different states achievable by an algorithm of best arms identification and their impact on the sample complexity. Two states are achievable during a task of minimization of the sample complexity:

- $s(t) = 1$ if the algorithm is sampling an arm during the time-step t. In the case of SER4, $s(t) = 1$ when $|S_\tau| \neq 1$ and the sample complexity increases by one.
- $s(t) = 0$ if the algorithm submits an arm as the optimal one during the time-step t. In the case of SER4, $s(t) = 0$ when $|S_\tau| = 1$. The sample complexity increases by one if $k_t \neq k^*(t)$.

In the context of SER4, the sample complexity is the number of time-steps during which the arm set does not only contain the optimal arm.

Algorithm.

In order to allow the algorithm to choose another arm when a switch occurs, at each turn, estimators of SER3 are reseted with a probability $\varphi \in [0, 1]$ and a new task of best arm identification is started. We name this algorithm SUCCESSIVE ELIMINATION WITH RANDOMIZED ROUND-ROBIN AND RESETS (SER4).

Analysis.

We now provide the performance guarantees of the SER4 algorithm, both in terms of sample complexity and of pseudo cumulative regret.

The following results are given in expectation and in high probability. The expectations are taken with regard to the randomization of the resets. The sample complexity or the pseudo cumulative regret achieved by the algorithm between each resets (given by the analysis of SER3) are still results in high probability.

Theorem 14 (**Expected sample complexity of** SER4**).** *For* $K \geq 2$*,* $\tau_{\min} = \log \frac{K}{\delta}$ *and* $\varphi \in (0, 1]$*, the expected sample complexity of* SER4 *w.r.t. the randomization of resets is upper bounded by:*

$$O\left(\frac{\varphi K}{\delta \Delta^2} \log(\frac{K}{\delta \Delta}) + \frac{N}{\varphi}\right)$$

Algorithm 5 SUCCESSIVE ELIMINATION WITH RANDOMIZED ROUND-ROBIN AND RESETS (SER4)

input: $\delta \in (0,1]$, $\epsilon \in [0,1)$, $\varphi \in [0,1)$
$S_1 := [K]$, $\forall k$, $\hat{\mu}_k(0) := 0$, $t := 1$, $\tau := 1$
While $t \leq T$
Shuffle S_τ
For each $k \in S_\tau$ **do**
If $|S_\tau| \neq 1$ **or If** the algorithm performs a regret minimization task
Play k
$\hat{\mu}_k(\tau) := \frac{\tau-1}{\tau}\hat{\mu}_k(\tau-1) + \frac{y_{k_t}(t)}{\tau}$
End if
$t := t+1$
End for
Remove from $S_{\tau+1}$ all k such as:

$$\max_{k \in S} \hat{\mu}_k(\tau) - \hat{\mu}_k(\tau) + \epsilon \geq 2\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}$$

$\tau = \tau + 1$
$t := t+1$
With a probability φ
$S_t := [K]$
$\forall k$, $\hat{\mu}_k(t) := 0$
$\tau := 1$
End with a probability
End while

with a probability of at least $1-\delta$*.*

The proof is given in section 3.2.7.

We tune φ in order to minimize the sample complexity.

For $K \geq 2$, $\tau_{\min} = \log\frac{K}{\delta}$, $\Delta \geq \frac{\delta}{K}$ and $\varphi = \sqrt{\frac{N\delta}{K\log(\frac{K}{\delta})}}$, the expected sample complexity of SER4 w.r.t. the randomization of resets is upper bounded by:

$$O\left(\frac{1}{\Delta^2}\sqrt{\frac{NK\log(\frac{K}{\delta})}{\delta}}\right).$$

Remark 13. *Transposing Theorem 14 for the case where* $\epsilon \in [\frac{1}{KT}, 1]$ *is straightforward. This allows to tune the bound by setting* $\varphi = \epsilon\sqrt{(N\delta)/(K\log(TK))}$*.*

This result can also be transposed in bound on the expected cumulative regret. We consider that the algorithm continues to play the last arm of the set until a reset occurs.

Corollary 4 (**expected pseudo-regret of** SER4**).** *For* $K \geq 2$*, and* $\delta = 1/T$*,* $\tau_{\min} = \log(KT)$*,* $\Delta \geq \frac{1}{KT}$ *and*

$\varphi = \sqrt{\frac{N}{TK\log(KT)}}$, *the expected pseudo-regret of* SER4 *w.r.t. the randomization of resets is upper bounded by:*

$$\min\left(O\left(\frac{\sqrt{NTK\log(KT)}}{\Delta}\right), O\left(T^{2/3}\sqrt{NK\log\frac{T}{K}}\right)\right). \tag{3.28}$$

The proof is given in section 3.2.7.

Remark 14. *A similar dependency in* $\sqrt{T}\Delta^{-1}$ *appears also in* SW-UCB *(see Theorem 1 in [Garivier and Moulines, 2011]), and is standard in this type of results.*

EXP3 with Resets

SER4 and other algorithms from the state of the art [Auer et al., 2002b, Kocsis and Szepesvári, 2006a, Garivier and Moulines, 2011] use a passive approach through forgetting the past. In this subsection, we propose an active strategy which consists in resetting the reward estimations when a change of the best arm is detected. A supposed advantage of this approach is to let the algorithm converge on a longer time period, as it is reset only when a switch is detected, and thus build a more accurate estimate of the reward distributions. First, we describe the adversarial bandit algorithm EXP3 [Auer et al., 2002b], which will be used by the proposed algorithm EXP3.R between detections. We then describe the drift detector used to detect changes of the best arm. Finally, we combine the both to obtain the EXP3.R algorithm.

Algorithm 6 EXP3

The parameter $\gamma \in [0,1]$ controls the exploration and the probability to choose an action k at round t is:

$$p_k(t) := (1-\gamma)\frac{w_k(t)}{\sum_{i=1}^{k} w_i(t)} + \frac{\gamma}{K}, \tag{3.29}$$

where the weight $w_k(t)$ of each action k is computed from the unbiased cumulative reward estimator $\hat{X}_k(t)$:

$$w_k(t) := \exp(\frac{\gamma}{K}\hat{X}_k(t)), \tag{3.30}$$

with

$$\hat{X}_k(t) := \sum_{j=t_r}^{t} \frac{x_k(j)}{p_k(j)} k = k(j), \tag{3.31}$$

where t_r is the time steps when the algorithm is initialized.

The EXP3 algorithm.

The EXP3 algorithm (see Algorithm 6) minimizes the regret against the best arm using an unbiased estimation of

the cumulative reward at time t for computing the choice probabilities of each action. While this policy can be viewed as optimal in an actual adversarial setting, in many practical cases the non-stationarity within a time period exists but is weak and is only noticeable between different periods. If an arm performs well in a long time period but is extremely bad on the next period, the EXP3 algorithm can need a number of trial equal to the first period's length to switch its most played arm.

The detection test.

The detection test (see Algorithm 7) uses confidence intervals to estimate the expected reward in the previous time period. The action distribution in EXP3 is a mixture of uniform and Gibbs distributions. We call γ-observation an observation selected though the uniform distribution. Parameters γ, H and δ define the minimal number of γ-observations by arm needed to call a test of accuracy ϵ with a probability $1-\delta$. They will be fixed in the analysis (see Corollary 5) and the correctness of the test is proven in Lemma 3. We denote $\bar{\mu}^k(I)$ the empirical mean of the rewards acquired from the arm k on the interval I using only γ-observations and Γ_{min}(I) the smallest number of γ-observations among each action on the interval I. The detector is called only when $\Gamma_{\mathsf{min}}(I) \geq \frac{\gamma H}{K}$. The detector raises an alert when the action $k_{\max}$ with the highest empirical mean $\hat{\mu}^k(I-1)$ on the interval $I-1$ is eliminated by an other on the current interval.

Algorithm 7 DriftDetection(I)

Parameters: Current interval I
$k_{\max} := \arg\max_k \hat{\mu}^k(I-1)$
$\epsilon := \sqrt{\frac{K \log(\frac{1}{\delta})}{2\gamma H}}$
return $\exists k, \quad \hat{\mu}^k(I) - \hat{\mu}^{k_{\max}}(I) \geq 2\epsilon$

The EXP3.R algorithm.

Coupled with a detection test, the EXP3 algorithm has several advantages. First in a non-stationary environment, we need a constant exploration to detect changes where a sub-optimal arm becomes optimal and this exploration is naturally given by the algorithm. Second, the number of breakpoints is higher than the number of best arm changes ($M \geq N$). This means that the number of resets needed by EXP3 is lower than the one needed by a stochastic bandit algorithm such as UCB. Third, EXP3 is robust against test failures (non detection) or local non-stationarity. We call EXP3.R the algorithm obtained by combining EXP3 and the drift detector. First, one instance of EXP3 is initialized and used to select actions. When the count of $\frac{\gamma H}{K}$ γ-observations per arm is fulfilled, the detection test is called. If in the corresponding interval, the empirical mean of an arm exceeds by 2ϵ the empirical mean of the current best arm then a drift detection is raised. In this case, weights of EXP3 are reset. Then a new interval of collect begins, preparing the next test. These steps are repeated until the run ends (see Algorithm 8).

Analysis.

In this section we analyze the drift detector and then we bound the expected regret of the EXP3.R algorithm.

Algorithm 8 EXP3 with Resets

Parameters: Reals δ, γ and Integer H
$I := 1$
for each $t = 1, ..., T$ **do**
  Run EXP3 on time step t
  if $\Gamma_{\min}(I) \geq \frac{\gamma H}{K}$ **then**
    if $DriftDetection(I)$ **then**
      Reset EXP3
    end if
    $I := I + 1$
  end if
end for

Assumption 2 (**Accuracy of the drift detector).** *During each of the segments S where k_S^* is the optimal arm, the gap between k_S^* and any other arm is of at least 4ϵ with*

$$\epsilon = \sqrt{\frac{K \log(\frac{1}{\delta})}{4\gamma H}}\,. \tag{3.32}$$

Lemma 3 guarantees that when Assumption 1 holds and the interval I is included into the interval S then, with high probability, the test will raise a detection if and only if the optimal action k_S^* eliminates a sub-optimal action.

Lemma 3 (Arm switches are detected)**.** *When Assumption 2 holds and $I \subseteq S$, then, with a probability $1 - 2\delta$, for any $k \neq k_S^*$:*

$$\hat{\mu}^{k_S^*}(I) - \hat{\mu}^k(I) \geq 2\sqrt{\frac{K \log(\frac{1}{\delta})}{2\gamma H}} \Leftrightarrow \mu^{k_S^*}(I) \geq \mu^k(I)\,. \tag{3.33}$$

The proof is given in section 3.2.7.

Theorem 15 bounds the pseudo-regret of EXP3.R.

Theorem 15 (**pseudo-regret of** EXP3.R**).** *For any $K > 0$, $0 < \gamma \leq 1$, $0 \leq \delta < \frac{1}{2}$, $H \geq K$ and $N \geq 1$ when Assumption 2 holds, the pseudo-regret of* EXP3.R *is*

$$G^* - \boldsymbol{E}[G_{EXP3.R}] \leq (e-1)\gamma T + \frac{\left(N - 1 + \frac{K\delta T}{H} + K\delta\right) K \log(K)}{\gamma} + (N-1)HK\left(\frac{1}{1-2\delta} + 1\right). \tag{3.34}$$

The proof is given in section 3.2.7.

In Corollary 5 we optimize parameters of the bound obtained in Theorem 15.

Corollary 5 (**pseudo-regret of** EXP3.R**).** *For any $K \geq 1$, $T \geq 10$, $N \geq 1$ and $C \geq 1$ when Assumption 1 holds, the pseudo-regret of* EXP3.R *run with input parameters*

$$\delta = \sqrt{\frac{\log T}{KT}}, \gamma = \sqrt{\frac{K \log K \log T}{T}} \text{ and } H = C\sqrt{T \log T} \tag{3.35}$$

is

$$G^* - \mathbf{E}[G_{EXP3.R}]) \leq (e-1)\sqrt{TK \log K \log T} + N\sqrt{TK \log K} + (C+1)K\sqrt{T \log K} + 3NCK\sqrt{T \log T}. \tag{3.36}$$

The proof is given in section 3.2.7.

Accordingly to C, the precision ϵ is:

$$\epsilon = \sqrt{\frac{1}{2C}} \sqrt{\frac{\log \sqrt{\frac{KT}{\log T}}}{\log T} \sqrt{\frac{K}{\log K}}}. \tag{3.37}$$

Notice that, when T increases, $\sqrt{\frac{\log \sqrt{\frac{KT}{\log T}}}{\log T} \sqrt{\frac{K}{\log K}}}$ tends towards a constant.

3.2.5 Numerical Experiments

We compare our algorithm with the state-of-the-art. For each problem, $K = 20$ and $T = 10^7$. The instantaneous gap between the optimal arm and the others is constant, $\Delta = 0.05$, i.e. the mean of the optimal arm is $\mu^*(t) = \mu(t) + \Delta$. During all experiments, probabilities of failure of SUCCESSIVE ELIMINATION (SE), SER3 and SER4 are set to $\delta = 0.05$. Constant explorations of all algorithms of the EXP3 family are set to $\gamma = 0.05$. Results are averaged over 50 runs. On problems 1 and 2, variances are low (in the order of 10^3) and thus not showed. On problem 3, variances are plotted as the gray areas under the curves.

Problem 1: Sinusoidal means

The index of the optimal arm k^* is drawn before the game and does not change. The mean of all suboptimal arm is $\mu(t) = \cos(2\pi t/K)/5 + 0.5$.

This problem challenges SER3 against SE, UCB and EXP3. SER3 achieves a low cumulative regret, successfully eliminating sub-optimal arms at the beginning of the run. Contrarily, SE is tricked by the periodicity of

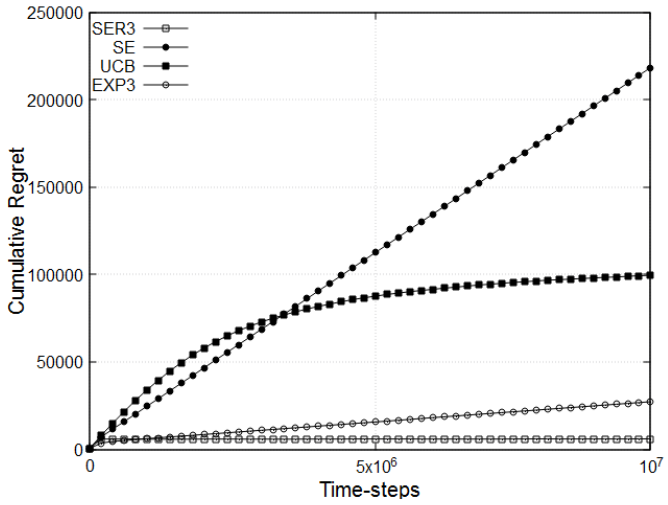


Figure 3.5: Cumulative regret of SER3, SE, UCB and EXP3 on the Problem 1.

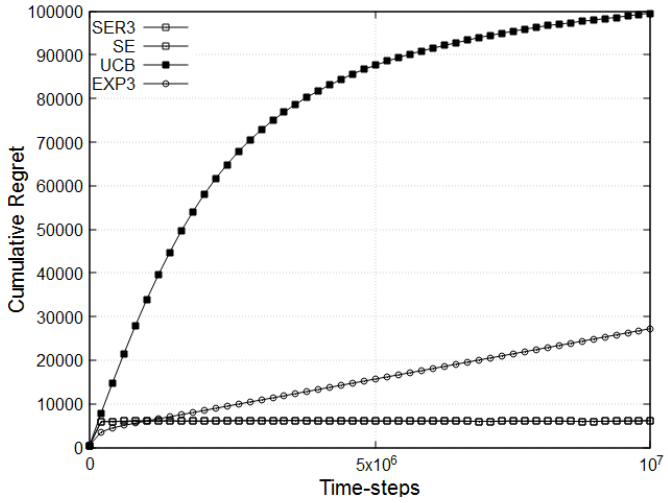


Figure 3.6: Cumulative regret of SER3, UCB and EXP3 on the Problem 2.

the sinusoidal means and eliminates the optimal arm. The deterministic policy of UCB is not adapted to the non-stationarity of rewards and thus the algorithm suffers from a high regret. The unbiased estimators of EXP3 enable the algorithm to quickly converge on the best arm. However, EXP3 suffers from a linear regret due to its constant exploration until the end of the game.

Problem 2: Decreasing means with positive gap

The optimal arm k^* does not change during the game. The mean of all suboptimal arms is $\mu(t) = 0.95 - \min(0.45, 10^{-7}t)$.

On this problem, SER3 is challenged against SE, UCB and EXP3. SER3 achieves a low cumulative regret, successfully eliminating sub-optimal arms at the beginning of the run. Contrarily to problem 1, mean rewards

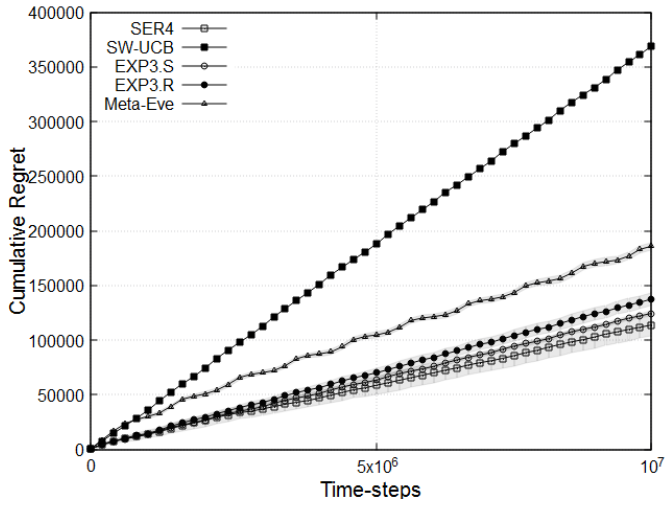


Figure 3.7: Cumulative regret of SER4, SW-UCB, EXP3, EXP3.R and META-EVE on the Problem 3.

evolve slowly and SUCCESSIVE ELIMINATION (SE) achieves the same level of performance than SER3. Similarly to problem 1, UCB achieves a high cumulative regret. The cumulative regret of EXP3 is low at the end of the game but would still increase linearly with time.

Problem 3: Decreasing means with arm switches

At every turn, the optimal arm $k^*(t)$ changes with a probability of 10^{-6}. In expectation, there are 10 switches by run. The mean of all suboptimal arms is $\mu(t) = 0.95 - \min(0.45, 10^{-7}(t[\text{mod } 10^6])$.

On problem 3, SER4 is challenged against SW-UCB, EXP3.S, EXP3.R and META-EVE. The probability of reset of SER4 is $\varphi = 5^{-5}$. The size of the window of SW-UCB is 10^5. The historic considered by EXP3.R is $H = 4 \cdot 10^5$ and the regularization parameter of EXP3.S is $\alpha = 10^{-5}$.

SER4 obtains the lowest cumulative regret, confirming the random resets approach to overcome switches of best arm. SW-UCB suffers from the same issues as UCB in previous problems and obtains a very high regret. Constant changes of mean cause META-EVE to reset very frequently and to obtain a lower regret than SW-UCB. EXP3.S and EXP3.R achieves both low regrets but EXP3.R suffers from the large size of historic needed to detect switches with a gap of Δ. We can notice that the randomization of resets in SER4, while allowing to achieve the best performances on this problem, involve a highest variance. Indeed, on some runs, a reset may occur lately after a best arm switch whereas the use of windows or regularization parameters will be more consistent.

3.2.6 Conclusion

We proposed a new formulation of the multi-armed bandit problem that generalize the stationary stochastic, piecewise-stationary and adversarial bandit problems. This formulation allows to manage difficult cases, where the means rewards and/or the best arm may change at each turn of the game. We studied the benefit of *random shuffling* in the design of sequential elimination bandit algorithms. We showed that the use of *random shuffling* extends their range of application to a new class of best arm identification problems involving non-stationary distributions, while achieving the same level of guarantees than SE with stationary distributions. We introduced SER3 and extended it to the switching bandit problem with SER4 by adding a probability of restarting the best arm identification task. We extended the definition of the sample complexity to include switching policies. Up to our knowledge, we proved the first sample complexity based upper-bound for the best arm identification problem with arm switches. The upper-bound over the cumulative regret of SER4 depends only of the number $N-1$ of arm switches, as opposed to the number of distribution changes $M-1$ in SW-UCB ($M \geq N$ can be of order T in our setting). The algorithm EXP3.R also achieves a competitive regret bound. The adversarial nature of EXP3 makes it robust to non-stationarity and the detection test accelerates the switch when the optimal arm changes while allowing convergence of the bandit algorithm during periods where the best arm does not change.

3.2.7 Proofs

Proof of Theorem 12 and Theorem 13

Proof. Theorem 12 is a special case of Theorem 13. For Theorem 12, for every k and every t, $B = 0$, $B_k = 0$ and $b_k(t) = 0$.

The proof consists of three main steps. The first step makes explicit the conditions leading to the elimination of an arm from the set. The second step shows that the optimal arm will not be eliminated with high probability. Finally, the third step shows that a sub-optimal arm will be eliminated after at most a critical number of steps τ^*, which then allows to derive an upper-bound on the sample complexity.

Step 1. Conditions for the elimination of an arm.

From Hoeffding's inequality, for any deterministic round-robin length τ and arm k we have:

$$P\left(|\hat{\mu}_k - \mathbb{E}[\hat{\mu}_k]| \geq \epsilon_\tau\right) \leq 2\exp\left(-2\epsilon_\tau^2 \tau\right) .$$

where $\mathbb{E}$ denotes the expectation with respect to the distribution D_y. By setting

$$\epsilon_t = \sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}, \text{ we have:}$$

$$P\left(|\hat{\mu}_k - \mathbb{E}[\hat{\mu}_k(\tau)]| \geq \epsilon_t\right) \leq 2\exp\left(-2\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}^2 \tau^2\right) = \frac{\delta}{2K\tau^2}\,.$$

Applying Hoeffding's inequality for each round-robin size $\tau \in \mathbb{N}^\star$, applying a standard union bound and using that $\sum_{\tau=1}^{\infty} 1/\tau^2 = \pi^2/6$, the following inequality holds simultaneously for any τ with a probability at least $1 - \frac{\delta\pi^2}{12K}$:

$$\hat{\mu}_k(\tau) - \epsilon_\tau \leq \mathbb{E}[\hat{\mu}_k] \leq \hat{\mu}_k(\tau) + \epsilon_\tau\,. \tag{3.38}$$

Let $S_i \subset \{1,\ldots,K\}$ be the set containing all the arms that are not eliminated by the algorithm at the start of the i^{th} round-robin. By construction of the algorithm, an arm k' remains in the set of selected arms as long as for each arm $k \in S_\tau - \{k'\}$:

$$\hat{\mu}_k(\tau) - \epsilon_\tau < \hat{\mu}_{k'}(\tau) + \epsilon_\tau \text{ and } \tau \geq \tau_{\min} \tag{3.39}$$

Combining (3.38) and the left inequality of (3.39), it holds on an event Ω of high probability

$$\mathbb{E}[\hat{\mu}_k(\tau)] - 2\epsilon_\tau < \mathbb{E}[\hat{\mu}_{k'}(\tau)] + 2\epsilon_\tau\,. \tag{3.40}$$

We denote t_τ, the time-step where the τ^{th} round-robin starts ($t_\tau = 1 + \sum_{i=1}^{\tau-1}|S_i|$). Let us remind that $\mathbb{T}(\tau)$ is the set containing all possible realizations of τ sequences of round-robin. Each arm k is played one time during each round-robin phase and thus τ observations per arm are available after τ^{th} round-robin phases. The empirical mean reward $\hat{\mu}_k(\tau)$ of each arm k after the τ^{th} round-robin is:

$$\hat{\mu}_k(\tau) = \sum_{r\in\mathbb{T}(\tau)} \frac{1_{r=[\tau]}}{\tau} \sum_{j=1}^{t_\tau+|S_\tau|-1} y_k(j) 1_{k=k_j}\,. \tag{3.41}$$

Decomposing the second sum in round-robin phases and taking the expectation with respect to the reward distribution D_y we have:

$$\mathbb{E}_{D_y}[\hat{\mu}_k(\tau)] = \sum_{r\in\mathbb{T}(\tau)} \frac{r=[\tau]}{\tau} \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} (\mu_k(j) - b_k(t))k = k_j\,. \tag{3.42}$$

Taking the expectation of equation (3.42) with respect to the randomization of the round-robin we have:

$$\mathbb{E}[\hat{\mu}_k(\tau)] = \left(\sum_{r\in\mathbb{T}(\tau)} \frac{r=[\tau]}{\tau} \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|}\right) - \frac{B_k}{\tau}\,. \tag{3.43}$$

Now, under the event Ω for which (3.40) holds for k and k', we deduce by using (3.43) that

$$\sum_{r\in\mathbb{T}(\tau)} \frac{r=[\tau]}{\tau}\left(\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_\tau|-1}\frac{\mu_k(j)}{|S_i|}-\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_\tau|-1}\frac{\mu_{k'}(j)}{|S_i|}\right) < 4\epsilon_\tau + \frac{B_k}{\tau}-\frac{B_{k'}}{\tau}+\frac{B}{\tau}\,. \tag{3.44}$$

Let us introduce the following mean-gap quantity

$$\Delta_{k,k'}([\tau]) = \sum_{r\in\mathbb{T}(\tau)} \frac{1_{r=[\tau]}}{\tau}\left(\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_\tau|-1}\frac{\mu_k(j)}{|S_i|}-\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_\tau|-1}\frac{\mu_{k'}(j)}{|S_i|}\right)\,.$$

Replacing the value of ϵ_t in (3.44), it comes

$$\Delta_{k,k'}([\tau]) < 4\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}+\frac{B_k}{\tau}-\frac{B_{k'}}{\tau}+\frac{B}{\tau}\,,$$

$$\Delta_{k,k'}([\tau])^2 < \frac{8}{\tau}\log\left(\frac{4K\tau^2}{\delta}\right)+\frac{B_k}{\tau}-\frac{B_{k'}}{\tau}+\frac{B}{\tau}\,. \tag{3.45}$$

An arm will be eliminated if (3.45) becomes false and if $\tau \geq \tau_{\min}$.

Step 2. The optimal arm is not eliminated.

For $k' = k^*$ et $k \neq k^*$, in the worst case $B_k = 0$ and $B_{k'} = B$. After injecting those quantities in (3.45), we have :

$$\Delta_{k,k'}([\tau])^2 < \frac{8}{\tau}\log\left(\frac{4K\tau^2}{\delta}\right)\,. \tag{3.46}$$

By assumption ($\Delta_{k,k^*}([\tau])$ is negative after $\tau_{\min}$), (3.46) is always true when $\tau \geq \tau_{\min}$,implying that the optimal arm will always remain in the set with a probability of at least $1-\frac{\delta}{K}$ for all τ.

Step 3. The elimination of sub-optimal arms.

If the arm k' still remain in the set, it will be eliminated if inequality (3.45) is not satisfied and if $\tau^* \geq \tau_{\min}$.

Let us consider $k = k^*$, $k' \neq k^*$, and define the quantity

$$\Delta_k([\tau]) = \sum_{r\in\mathbb{T}(\tau)} \frac{1_{r=[\tau]}}{\tau}\left(\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_\tau|-1}\frac{\mu_k(j)}{|S_i|}-\sum_{i=1}^{\tau}\sum_{j=t_i}^{t_i+|S_\tau|-1}\frac{\mu_{k'}(j)}{|S_i|}\right)\,.$$

In the worst case, $B_{k^*} = B$ et $B_k = 0$. Using equation (3.45) we obtain the condition to invalidate to eliminate the arm of index k:

$$\Delta_{k,k'}([\tau])^2 < \frac{8}{\tau}\log\left(\frac{4K\tau^2}{\delta}\right)+\frac{2B}{\tau}\,. \tag{3.47}$$

Let us also introduce for convenience the critical value

$$\tau_1^* = \frac{64^2}{\Delta_k([\tau])^2}\log\left(\frac{16K}{\delta\Delta_k([\tau])}\right)\,.$$

Notice that $\tau_1^* \geq \tau_{\min}$, satisfying one of the two conditions needed to eliminate an arm.

We introduce the following quantity

$$C_1(t) = \frac{8}{\tau} \log\left(\frac{4K\tau^2}{\delta}\right).$$

For $\tau = \tau_1^*$, we derive the following bound $\mathsf{C}_1(\tau_1^*) = \frac{8\Delta_k([\tau])^2}{64^2 \log \frac{16K}{\delta\Delta_k([\tau])}} \left(\log \frac{4K}{\delta} + 2\log \frac{64}{\Delta_k([\tau])^2} + 2\log\log \frac{16K}{\delta\Delta_k([\tau])}\right)$,
$= \frac{8\Delta_k([\tau])^2}{64^2 \log \frac{16K}{\delta\Delta_k([\tau])}} \left(\log \frac{4K}{\delta} - 4\log \Delta_k([\tau]) + 24\log 2 + 2\log\log \frac{16K}{\delta\Delta_k([\tau])}\right)$,
$\leq \frac{8\Delta_k([\tau])^2}{64^2 \log \frac{16K}{\delta\Delta_k([\tau])}} \left(4\log \frac{16K}{\delta\Delta_k([\tau])} + 24\log 2 + 2\log\log \frac{16K}{\delta\Delta_k([\tau])}\right)$.

We remark that for $X > 8$ we have

$$24\log 2 + 2\log\log X < 8\log X\,.$$

Hence, provided that for $K \geq 2$, $\delta \in (0, 0.5]$ and $\Delta_k([\tau]) > 0$, we have $\frac{4K}{\delta\Delta_k([\tau])} > 8$ and

$\mathsf{C}_1(\tau_1^*) \leq \frac{8\Delta_k([\tau])^2}{64^2 \log \frac{16K}{\delta\Delta_k([\tau])}} \left(16\log \frac{16K}{\delta\Delta_k([\tau])}\right)$
$\leq \frac{\Delta_k([\tau])^2}{512}$.

As $C_1(\tau_1^*)$ is strictly decreasing with regard to t, (3.2.7) is true for all $\tau > \tau_1^*$.

When $t > \tau_1^*$, it exists $C_2(t)$ such as:

$$\Delta_k([\tau])^2 = C_1(t) + C_2(t)\,.$$

For invalidating 3.47, we must find a value $\tau_2^* > \tau_1^*$ such as:

$$\tau_2^* \geq \frac{4B}{C_2(t_2^*)} \tag{3.48}$$

As $C_2(\tau) = \Delta_k([\tau])^2 - C_1(\tau)$, we have $C_2(\tau) \geq \Delta_k([\tau])^2 - \frac{\Delta_k([\tau])^2}{512}$ and:

$$\tau \geq \frac{2048B}{511\Delta_k([\tau])^2}$$

For $\tau = \tau_2^*$ with:

$$\tau_2^* = \frac{64^2}{\Delta_k([\tau])^2} \log\left(\frac{16K}{\delta\Delta_k([\tau])}\right) + \frac{5B}{\Delta_k([\tau]^2)}\,. \tag{3.49}$$

(3.48) is true, invalidating (3.47) and invalidating (3.45) and involving the elimination of the suboptimal arms k with a probability at least $1 - \delta/K$.

We conclude the proof by summing over all the arms, taking the union bound and lower-bounding all $\Delta_k([\tau])$ by

$$\Delta = \min_{[\tau]\in\mathbb{T}(\tau),k} \sum_{r\in\mathbb{T}(\tau)} \frac{r = [\tau]}{\tau} \left(\sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|} - \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_{k'}(j)}{|S_i|}\right). \tag{3.50}$$

□

Proof of Corollary 3

Proof. We first provide the proof of the distribution dependent upper-bound.

The pseudo cumulative regret of the algorithm is:

$$R(T) = \sum_{k \neq k^*} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \Delta_{k^*,k}(t) 1_{k=k_t} \,. \tag{3.51}$$

Taking in each round-robin the expectation of the corresponding random variable k_t with respect to the randomization of the round-robin (denoted by $\mathbb{E}_{k_t}$), it comes:

$$\begin{aligned} \mathbb{E}[R(T)] &= \mathbb{E}\Big[\sum_{k \neq k^*} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \mathbb{E}_{k_t}[\Delta_{k^*,k}(t) 1_{k=k_t}] \Big] \\ &= \mathbb{E}\Big[\sum_{k \neq k^*} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(t)}{|S_i|} \Big] . \end{aligned}$$

$$\mathbb{E}[R(T)] = \mathbb{E}\Big[\sum_{k \neq k^*} \tau \underbrace{\frac{1}{\tau} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(t)}{|S_i|}}_{\Delta_k^*} \Big] = \mathbb{E}\Big[\sum_{k \neq k^*} \tau \Delta_k^* \Big] . \tag{3.52}$$

The penultimate step of the proof of Theorem 12 allows us to upper-bound τ with the previously introduced critical value τ^* on an event of high probability $1-\delta$, while the cumulative regret is controlled by the trivial upper bound T on the complementary event of probability not higher than δ, leading to:

$$\mathbb{E}[R(T)] \leq \sum_{k \neq k^*} \frac{64}{\Delta_k^2} \log\left(\frac{4K}{\delta \Delta_k}\right) \Delta_k + \delta T \,. \tag{3.53}$$

We conclude the proof of the distribution dependent upper-bound by setting $\delta = 1/T$ and :

$$\mathbb{E}[R(T)] = O\left(\frac{K-1}{\Delta} \log(\frac{KT}{\Delta}) \right) , \tag{3.54}$$

with $\Delta = \min_{[\tau],k} \frac{1}{\tau} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(t)}{|S_i|}$.

We now upper-bound the regret in the worst case in order to derive a distribution independent bound. To this end, we consider a sequence that ensures that, with high probability, no suboptimal arm is eliminated by the algorithm at the end of the T rounds, while maximizing the instantaneous regret. According to (3.40) an arm is not eliminated as long as

$$\mathbb{E}[\hat{\mu}_k(\tau)] - \mathbb{E}[\hat{\mu}_{k'}(\tau)] < 4\epsilon_\tau \,. \tag{3.55}$$

By injecting (3.55) in (3.52) and replacing ϵ_τ by its value $\sqrt{\frac{2}{\tau} \log\left(\frac{4K\tau^2}{\delta}\right)}$ we obtain:

$$\mathbb{E}[R(T)] < \sum_{k \neq k^*} \tau 4 \sqrt{\frac{2}{\tau} \log\left(\frac{4K\tau^2}{\delta}\right)} + \delta T \,. \tag{3.56}$$

The non-elimination of sub-optimal arms involves $\tau = \frac{T}{K}$ and by setting $\delta = \frac{1}{T}$ we obtain the distribution independent upper-bound:

$$\mathbb{E}[R(T)] < (K-1)\frac{T}{K} 4 \sqrt{\frac{K}{T} \log(\frac{4T^3}{K})} + 1 \,, \tag{3.57}$$

$$\mathbb{E}[R(T)] = O\left(\sqrt{TK \log \frac{T}{K}}\right) . \tag{3.58}$$

□

Proof of Theorem 14

Proof. In order to prove Theorem 14, we consider the following quantities:

- The expected number of times when the estimators are reseted: $N_{\text{reset}} = \varphi T$.
- The sample complexity needed to find the best arm between each reset is $S_{\text{SER3}} = O\left(\frac{K}{\Delta^2} \log(\frac{K}{\delta\Delta})\right)$.
- The time before a reset, that follows a negative binomial distribution of parameters $r = 1$ and $p = 1 - \varphi$. Its expectation is upper-bounded by $1/\varphi$.
- The number of arm switches: $N - 1$.

The sample complexity of SER4 is the total number of time-steps spent sampling an arm added to the time between each switch and reset.

Taking the expectation with respect to the randomization of resets, we obtain an upper-bound on the expected number of suboptimal plays given by

$$O\left(\frac{\varphi TK}{\Delta^2} \log\left(\frac{K}{\delta\Delta}\right) + \frac{N}{\varphi}\right) . \tag{3.59}$$

The first term is the expectation of the total number of time-steps required by the algorithm in order to find the best arms at its initialization and then after each reset of the algorithm. The second term is the expected total number of steps lost by the algorithm when not resetting the algorithm after the $N - 1$ arm switches.

We obtain the final statement of the Theorem by setting $T = \frac{1}{\delta}$. □

Proof of Corollary 4

Proof. Converting Corollary 3.2.4 into a distribution dependent upper-bound on the cumulative regret is straightforward by setting $\delta = \frac{1}{T}$, replacing the sample complexity in the proof of Theorem 14 by the cumulative regret and

using the upper-bound of Corollary 3.

$$\mathbb{E}[R(T)] = O\left(\frac{\varphi TK}{\Delta}\log\left(\frac{KT}{\Delta}\right) + \frac{N}{\varphi}\right). \tag{3.60}$$

Setting $\varphi = \sqrt{\frac{N}{TK\log(KT)}}$ and assuming $\Delta \geq \frac{1}{KT}$ we obtain the final statement of the theorem:

$$\mathbb{E}[R(T)] = O\left(\frac{\sqrt{NTK\log(KT)}}{\Delta}\right). \tag{3.61}$$

We also derive below a distribution independent upper-bound. We introduce some notations, N_{reset} is the number of resets, τ_i^{reset} is the number of round-robin phases between the i^{th} and the $(i+1)^{\text{th}}$ resets and L_n is the number of timesteps before a reset after the n^{th} arm switch.

When the resets are fixed, the expected cumulative regret is:

$$\mathbb{E}[R(T)] < \mathbb{E}\left[\sum_{i=1}^{N_{\text{reset}}+1}(K-1)\tau_i^{\text{reset}}4\sqrt{\frac{2}{\tau_i^{\text{reset}}}\log(\frac{4(\tau_i^{\text{reset}})^2}{\delta})} + \sum_{n=1}^{N}L_n + \delta T\right], \tag{3.62}$$

$$\mathbb{E}[R(T)] < \mathbb{E}\Bigg[\sum_{i=1}^{N_{\text{reset}}+1}\underbrace{(K-1)4\sqrt{2\tau_i^{\text{reset}}\log(\frac{4(\tau_i^{\text{reset}})^2}{\delta})}}_{f(\tau_i^{\text{reset}})}\Bigg] + \mathbb{E}\left[\sum_{n=1}^{N}L_n\right] + \delta T\,. \tag{3.63}$$

At this point, we note that $\{\tau_i^{\text{reset}}\}_i$ is an i.i.d sequence of random variables and that N_{reset} is a random stopping time with respect to this sequence. Moreover, f is a concave function. We can thus apply Wald's equation followed by Jensen's inequality and deduce that

$$\begin{aligned}\mathbb{E}[\sum_{i=1}^{N_{\text{reset}}+1}f(\tau_i^{\text{reset}})] &\leq \mathbb{E}[N_{\text{reset}}+1]\mathbb{E}[f(\tau_1^{\text{reset}})]\\ &\leq \mathbb{E}[N_{\text{reset}}+1]f(\mathbb{E}[\tau_1^{\text{reset}}])\,.\end{aligned}$$

We upper-bound $\log(\frac{4(\tau_i^{\text{reset}})^2}{\delta})$ by $\log(\frac{4T^2}{\delta K^2})$ and set $\delta = \frac{1}{T}$. As $\mathbb{E}[N_{\text{reset}}] = \varphi T$, $\mathbb{E}[\tau_1^{\text{reset}}] = \frac{1}{\varphi K}$ and $\mathbb{E}[L_n] \leq \frac{1}{\varphi}$, we have

$$\mathbb{E}[R(T)] < 4(\varphi T+1)\sqrt{\frac{2}{\varphi}K\log\left(\frac{4T^3}{K^2}\right)} + \frac{N}{\varphi} + 1\,. \tag{3.64}$$

The previous equation makes appear a trade-off in φ, and we set $\varphi = \frac{\sqrt{N}}{T^{2/3}}$.

Finally we have shown that

$$\mathbb{E}[R(T)] = O\left(T^{2/3}\sqrt{NK\log\frac{T}{K}}\right). \tag{3.65}$$

□

Proof of Lemma 3

Proof. We justify our detection test by considering an observation of a reward through γ-exploration as a drawing in an urn without replacement. More specifically, when all the necessary observations are collected, the detection test procedure is called. During the interval, rewards were draw from m different distributions of mean $\mu_0^k(I), ..., \mu_m^k(I)$. We denote t_i the steps where the mean reward starts being $\mu_i^k(I)$ and t_{m+1} the time step of the call. When the test is called, all $x_k(t)$ have a probability $(t_{i+1} - t_i)/(t_{m+1} - t_0)$ to be drawn from the distribution of mean $\mu_i^k(I)$. The mean $\mu^k(I)$ of the urn corresponding to the action k is:

$$\mu^k(I) = \sum_{i=1}^{m} \frac{t_{i+1} - t_i}{t_{m+1} - t_0} \mu_i^k(I)\,. \tag{3.66}$$

At each time step, by assumption , the mean reward of the best arm is away by 4ϵ from any suboptimal arms. Consequently, the difference between the mean reward of the urn of the optimal arm k^* and that of an another arm k is at least 4ϵ if the best arm doesn't change during the interval.

$$\mu^k(I) \leq \sum_{i=1}^{m} \frac{t_{i+1} - t_i}{t_{m+1} - t_0} (\mu_i^{k_S^*} - 4\epsilon) \leq \mu^{k_S^*}(I) - 4\epsilon\,. \tag{3.67}$$

The following arguments prove the equivalence between the detection and the optimality of k_S^* with high probability.

Applying the Serfling inequality [Serfling, 1974], we have:

$$P(\hat{\mu}^{k_S^*}(I) + \epsilon \geq \mu^{k_S^*}(I)) \leq e^{\frac{-2n\epsilon^2}{1 - \frac{n-1}{U}}} \leq e^{-2n\epsilon^2} = \delta \tag{3.68}$$

and

$$P(\hat{\mu}^k(I) - \epsilon \leq \mu^k(I)) \leq \delta\,, \tag{3.69}$$

where $n = \frac{\gamma H}{K}$is the number of observation and U the size of the urn.

$$\mu^{k_S^*}(I) - \mu^k(I) \geq 4\epsilon \tag{3.70}$$

and with probability at least $1 - 2\delta$,

$$\hat{\mu}^{k_S^*}(I) + \epsilon \geq \mu^{k_S^*}(I) \tag{3.71}$$

and

$$\hat{\mu}^k(I) - \epsilon \leq \mu^k(I) \tag{3.72}$$

Summing (3.73), (3.71) and (3.72) we obtain:

$$\hat{\mu}^{k_S^*}(I) - \hat{\mu}^{k}(I) \geq 2\epsilon \tag{3.73}$$

This ensures, with high probability, a positive test if $\hat{\mu}^{k_{\max}}$ is not the optimal arm.

Reciprocally, we also have

$$\hat{\mu}^{k}(I) - \hat{\mu}^{k_S^*}(I) \leq -2\epsilon\,. \tag{3.74}$$

ensuring, with high probability, a negative test if $\hat{\mu}^{k_{\max}}$ is the optimal arm.

□

Proof of Theorem 15

Proof. First we obtain the main structure of the bound. In the following, $L(T)$ denotes the expected number of intervals after a best action change occurs before detection and $F(T)$ denotes the expected number of false detections up to time T. Using the same arguments as [Yu and Mannor, 2009] we deduce the form of the bound with drift detector from the classical EXP3 bound. If there are $N-1$ changes of best arm. Therefore the expectation of the number of resets over an horizon T is upper bounded by $N-1+F(T)$. The regret of EXP3 on these periods is $(e-1)\gamma T + \frac{K\log K}{\gamma}$ [Auer et al., 2002b]. While our optimal policy plays the arm with the highest mean, the optimal policy of EXP3 plays the arm associated with the actual highest cumulative reward. As

$$\sum_{t=T_S}^{T_{S+1}-1} x_{k_S^*}(t) \leq \max_k \sum_{t=T_S}^{T_{S+1}-1} x_k(t)\,, \tag{3.75}$$

the gain of our optimal policy is upper bounded by the gain the EXP3 optimal policy. Summing over each periods we obtain $(e-1)\gamma T + \frac{(N-1+F(T))K\log K}{\gamma}$.

The regret also include the delay between a best arm change and its detection. To evaluate the expected size of the intervals between each call of the detection test, we consider an hypothetical algorithm that collects only the observations of one arm and then proceeds on the next arm until collecting all the observations. The γ-observation are drawn with a probability $\frac{\gamma}{K}$ and $\frac{\gamma H}{K}$ observations are needed per action. The expectation of the number of failures before collecting $\frac{\gamma H}{K}$ γ-observations follows a negative binomial distribution of expectation

$$\frac{\gamma H}{K}(1-\frac{\gamma}{K})\frac{K}{\gamma} = H - \frac{\gamma H}{K}\,. \tag{3.76}$$

The expectation of the number of steps at the end of the collect is the number of success plus the expected number of failures:

$$\frac{\gamma H}{K} + H - \frac{\gamma H}{K} = H\,. \tag{3.77}$$

Summing over all arms gives a total expectation of HK. Because our algorithm collects γ-observations from any arm at any step, on a same sequence of drawings, our algorithm will collect the required observations before the hypothetical algorithm. By consequence, the expectation of the time between each query of the detection test is upper bounded by HK and lower bounded by H, the expected time of collect for one arm. There are $N-1$ best action changes and the detections occur at most $\lceil L(T) \rceil HK$ time steps after the drifts. Finally, there are also at most $N-1$ intervals where the optimal arm switches. In these intervals we don't have any guarantee on the test behavior due to this change. In the worst case, the test doesn't detect the drift and we set the instantaneous regret to 1.

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\gamma T + \frac{(N-1+F(T))K\log K}{\gamma} + (N-1)HK(\lceil L(T) \rceil + 1). \quad (3.78)$$

We now bound $F(T)$ and $L(T)$. Confidence intervals hold with probability $1-\delta$ and they are used K times at each detection test. The maximal number of calls of the test up to time horizon T is $\frac{T}{H}+1$. Using the union bound we deduce $F(T) \leq K\delta(\frac{T}{H}+1)$. $L(T)$ is the first occurrence of the event DETECTION after a drift. When a drift occurs, Lemma 3 ensures the detection happens with a probability $1-2\delta$. We have $L(T) \leq \frac{1}{1-2\delta}$.

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\gamma T + \frac{\left(N-1+\frac{K\delta T}{H}+K\delta\right)K\log K}{\gamma} + (N-1)HK\left(\frac{1}{1-2\delta}+1\right). \quad (3.79)$$

□

Proof of Corollary 5

Proof. We set $\delta = \sqrt{\frac{\log T}{KT}}$ and $H = C\sqrt{T\log T}$ in Theorem 15

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\gamma T + \frac{(N-1+(C+1)\sqrt{K})K\log K}{\gamma} + 3(N-1)CK\sqrt{T\log T}. \quad (3.80)$$

Finally, setting $\gamma = \sqrt{\frac{K\log K\log T}{T}}$ we obtain:

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\sqrt{TK\log K\log T} + N\sqrt{TK\log K} + (C+1)K\sqrt{T\log K} + 3NCK\sqrt{T\log T}. \quad (3.81)$$

□

Chapter 4

Bandits in environments with context information

4.1 HSLinUCB for Cloud auto-scaling

Abstract. One characteristic of the Cloud is elasticity: it provides the ability to adapt resources allocated to applications as needed at runtime. This capacity relies on scaling and scheduling. In this article online horizontal scaling is studied. The aim is to determine dynamically applications deployment parameters and to adjust them in order to respect a Quality of Service level without any human parameters tuning. This work focuses on CaaS (container-based) environments and proposes an algorithm based on contextual bandits (HSLinUCB). Our proposal has been evaluated on a simulated platform and on a real Kubernetes's platform. The comparison has been done with several baselines: threshold based auto-scaler, Q-Learning, and Deep Q-Learning. The results show that HSLinUCB gives very good results compared to other baselines, even when used without any training period.

4.1.1 Targeted use case: Cloud auto-scaling

Cloud hosted services need agile, automated and dynamic elasticity techniques. The Cloud elasticity can be decomposed in two sub-problems [Gari et al., 2020]: scaling and scheduling. The former consists in determining and adjusting the appropriate number of resources and the second consists in assigning workflow jobs on resources for execution. Automatic scaling techniques help to guarantee SLA (Service Level Agreement) to applications while optimizing the resources allocation of the Cloud provider [Singh et al., 2019]. An auto-scaler is defined as a system that autonomously takes decisions to optimize the latency, i.e., the response time, under the constraint of minimizing the number of allocated resources. Depending on the workflow, scheduling has to be taken into account or not. For parallel and distributed computing, the scheduling has to find the most efficient allocation between jobs and

resources, while in Web application the scheduling is implicitly done by the load balancer. This paper focuses on the auto-scaling problem for Web application in the Cloud, and hence the scheduling problem is not considered.

Two different elasticity techniques can be achieved depending on the used Cloud technology [Coutinho et al., 2015]. Horizontal elasticity is a technique allowing to create or delete a complete Cloud resource (container for a CaaS technology and virtual machine for the IaaS technology). These resources have predefined size (number of CPU, ram size, storage size...) at creation time and a dedicated pricing model. Any desired change in the Cloud resource type implies the complete deletion/recreation of a Cloud resource. The second technology is the vertical elasticity technique. The vertical elasticity can be used in two different modes: resizing or replacement [Hwang et al., 6 01]. The resizing mode allows to modify an existing Cloud resource size without the constraint to completely delete/recreate the resource. The replacement mode consists in creating a new more powerful resource and deleting the old one. This paper focuses on the horizontal scaling as it is the most widely used method to provide elasticity.

Resource creation time, service stabilization time and the adaptation capacity to dynamically create a new resource in face of a workload change is an important constraint in horizontal resource elasticity [Pascual et al., 2018]. Indeed in a IaaS environment a virtual machine creation time is in the range of the minute [Abdullah et al., 2019]. Container in a CaaS Cloud computing environment is said to not suffer from these constraints as the creation time is considered to be in the range of the second under the assumptions that the container image does not need to be downloaded and that the container manager does not suffer from resource constraints. Hence, CaaS environment has been chosen in this work.

When a new application is hosted in the Cloud, the auto-scaler has no knowledge about its hosting requirements nor its specific workload. Furthermore, nowadays the DevOps process has been adopted by many companies. This process implements some automatic delivery procedures which dramatically reduce the delivery time and increase the delivery frequency. This introduces some minor to major changes in both the hosting requirements and the workload. The hot-start or cold-start mode define respectively an auto-scaler, starting to operate, with or without environment knowledge. An auto-scaler, which can quickly perform from a cold-start, can be used for a new hosted application or for a major change in an existing application. An auto-scaler which can only work from a hot-start can only handle minor changes.

In this paper, the cold-start processing capability is sought, and thus, the minimization of the interactions number needed to adapt the resources to the current workload. That is why the use of contextual bandits is considered among the relevant learning techniques towards the best horizontal scaling policy in CaaS environments. The contributions in this article are the following: i) a mathematical formulation of the horizontal scaling problem in Cloud is provided, ii) a new scaling algorithm called Horizontal Scaling LinUCB (HSLinUCB) is proposed, iii) a complete experimental environment is made publicly available through a GitHub repository, iv) extensive simulations and real experiments have been conducted to exhibit the benefits of HSLinUCB over approaches from the literature. In the next section the state of the art on auto-scaling is presented. Then, the problem of Cloud elasticity is described and

formalized as a contextual bandit problem. Then, the proposed algorithm HSLinUCB is detailed. The experimental section presents the Cloud platform used for comparing the approach with different baselines. The real platform is used to generate and store datasets which are then used to simulate, with different workload, the behaviour of our proposal. Then, real online experiments are presented in a real CaaS environment. The last section is devoted to the conclusion and future works.

4.1.2 Related Work

The surveys in [Gari et al., 2020, Lorido-Botran et al., 2014, Qu et al., 2018, Al-Dhuraibi et al., 2018] show the active research topics on automatic scaling for Cloud infrastructure and applications. Threshold-based auto-scaling is the most popular technique. Basically, most Cloud computing environments[1][2] offer to application providers a threshold based auto-scaler. In their turn, application providers have to configure thresholds on Cloud resources e.g., CPU, memory, network throughput, consumed by their application deployment units to be used, e.g., virtual machines and containers. This configuration task is complex: setting the thresholds is dependent on each application and requires a deep understanding of workload trends.

Time-series analysis is essentially used for resource requirement forecasting [Nikravesh et al., 2015, Nguyen et al., 2013, Shariffdeen et al., 2016, Khatua et al., 2010]. Various prediction techniques are firstly used to extrapolate future values, e.g., moving average, auto-regression, exponential smoothing and various machine learning techniques like neural networks. Prediction adds pieces of information to the auto-scaler to take decisions. It is worth noting that Amazon combined the Threshold techniques with Time-series analysis to bring pro-activity to their auto-scaler.[3] This paper does not focus on time series analysis, but this approach could be used in combination of reinforcement learning for predicting the future states of the environment.

Another approach consists in modeling the system with queuing theory [Tadakamalla and Menasce, 2018], and control theory in order to shape specific controllers [Lorido-Botran et al., 2014, Al-Dhuraibi et al., 2018]. These approaches are efficient but necessitate a deep understanding of the system (Cloud, workload and applications). These model-based approaches are effective for specific applications, for which the model has been well tuned. However, to be resilient in case of change of the environment, they require to be tuned over time. Authors in [Jin et al., 2018] showed that model-free reinforcement learning is better suited to adapt in case of varying workload than model-based systems.

In reinforcement learning, an agent interacts with an environment. The agent observes the state of the environment, and then plays an action. The played action changes the state of the environment, and depending on the reached state the environment can generate a reward. The goal of the agent is to maximize its cumulative rewards. When the agent has a model of the environment, i.e., model-based reinforcement learning, this problem is reduced

[1] https://aws.amazon.com/ec2/autoscaling/
[2] https://docs.openstack.org/senlin/latest/tutorial/autoscaling.html
[3] https://aws.amazon.com/blogs/aws/new-predictive-scaling-for-ec2-powered-by-machine-learning/

to the planning problem. When the agent does not know the environment, i.e., model-free reinforcement learning, it has to interact with the environment to learn an efficient policy. Reinforcement Learning has been widely studied in the Cloud Elasticity research field for both scaling and scheduling aspects [Gari et al., 2020]. Q-learning has been used for horizontal scaling of IaaS Cloud environment in [Dutreilh et al., 2011, Barrett et al., 2013, Ayimba et al., 2019, Wei et al., 2019] and for horizontal scaling of CaaS Cloud environment in [Schuler et al., 2020]. Despite the fact that Q-learning based auto-scalers have proved their ability to learn the intrinsic dynamic of the environment, as it will be shown in our experiments, Q-Learning auto-scaler needs a lot of interactions (trials/errors) with the environment before starting to provide an efficient decision. If the learning is done online, the high number of interactions needed to explore the space of policies would increase hosting cost by over-provisioning, and would degrade the latency by under-provisioning. That is why Q-Learning auto-scalers are limited to hot start with an offline learning in a simulated environment. A particular case of reinforcement learning is used in this paper: Multi-Armed Bandits. The main difference with standard reinforcement learning is that the received reward only depends on the played action and not on the past sequence of actions. This simplification allows significant gain in terms of exploration costs, and hence allows to learn a model from cold start.

The use of Multi-Armed Bandits for Cloud elasticity is not as widespread as the use of standard reinforcement learning. However, Combinatorial Bandits has been used for task scheduling in [Xu et al., 2020]. [Toslali et al., 2020] uses a service mesh combined with a Multi-armed Bandits based on a Thompson sampling algorithm. The Multi-armed Bandits is not used for the scaling nor the scheduling parts of the elasticity but for identifying the best communication path between CaaS components. In [Cano et al., 2018], the authors propose a resource controller dedicated to IaaS environment for adjusting CPU and memory on a fixed number of virtual machines. As a contextual bandit is pre-trained on a simulated environment and then transferred on a real system, authors tackle only the hot-start stage and not the cold-start stage. In this paper, authors demonstrate the good behavior of a contextual bandit for vertical scaling.

In contrast, in the present work the horizontal scaling problem in CaaS environment is formalized as a contextual bandit problem in cold-start that could be also applied to IaaS environment.

4.1.3 Optimization of Container Allocation

Optimization of Container Allocation as a Reinforcement Learning Problem.

Our objective is to find and set the smallest number of containers required to satisfy SLA (Service Level Agreement) of a single client application under varying workloads. Here the SLA is expressed in term of response latency. Let $l_t \in \mathbb{R}_+$ be the observed latency at time t, and $l^* \in \mathbb{R}_+$ be the maximum latency corresponding to SLA. Let $w_t \in \mathbb{R}_+$ be the observed workload of the considered application at time t. Let $c_t \in \mathbb{N}_+$ be the number of containers of the application at time t. Let f be the function that given the number of containers allocated at time t c_t, and the

workload at the next time step w_{t+1} returns the latency at the next time step l_{t+1}:

$$f : \mathbb{R}_+ \times \mathbb{N}_+ \to \mathbb{R}_+, \quad l_{t+1} \leftarrow f(w_{t+1}, c_t). \tag{4.1}$$

The optimal allocation of containers of the application is defined as:

$$c_t^* = \arg \min_{c_t \in \mathbb{N}_+} \{c_t \text{ such that } f(w_{t+1}, c_t) \leq l^*\}. \tag{4.2}$$

In the following, this optimal allocation of containers of the application is denoted Oracle.

f is a complex function that depends on the Cloud environment, and that in the general case is unknown to the application. Moreover, the number of containers of the application is allocated at time t, while the latency at time $t+1$ depends on the workload at time $t+1$. As a consequence, the latency at time $t+1$ cannot be evaluated with the only knowledge of f at time t. Let w_{t+1} be the observed value at time $t+1$ of the random variable w sampled from an unknown distribution υ_w. Hence, the latency l_{t+1} is also the observed value at time $t+1$ of a random variable l sampled from an unknown distribution υ_l.

Algorithm 9 Optimization of Container Allocation.

1: **for** $t \leftarrow 1, ..., T$
2: The agent observes the workload $w_t \sim \upsilon_w$
3: The agent chooses a number of containers c_t
4: The latency $l_{t+1} \sim \upsilon_l$ is revealed
5: **End For**

Algorithm 9 provides a summary of the problem of optimization of container allocation in Cloud environments. Observing the latency at time $t+1$, a reward can be evaluated. Hence, this problem can be formalized as a reinforcement learning problem, where an agent interacts with an unknown environment.

4.1.4 Reward Function.

A set of three actions available to the agent is considered: $\mathcal{A} = \{u, s, d\}$, where:

- u is the choice of increasing the number of containers by one,
- s is the choice of staying with the same number of containers,
- d is the choice of decreasing the number of containers by one.

In practice, the Oracle is not available to the agent or the Cloud platform. The only feedback that the agent can obtain is the observed latency. For handling the optimization problem stated in Equation 4.2 as a reinforcement learning problem, a reward function based on the latency has to be built. Under the assumption that the workload does not change every time step, but every two time steps, the reward of the action $a \in \mathcal{A}$ is defined as:

$$r_{a,t} \equiv \begin{cases} r_{u,t} = 1 - \mathbb{1}\left\{l_t < l^*\right\}, \\ r_{s,t} = 1 - \mathbb{1}\left\{l_{t+1} \leq l^*\right\}, \\ r_{d,t} = \mathbb{1}\left\{l_{t+1} \leq l^*\right\}. \end{cases} \tag{4.3}$$

If the observed latency at time t is higher than the maximum latency, the agent has to increase the number of container: $r_{u,t} = 1$. Else the agent has to choose between the action stay s and the action down d. The choice of the action up is automatic when $l_t > l^*$, while the choice of the action stay s or down d is a 2-armed bandits. The agent is facing the explore / exploit dilemma: The agent can explore a loosely estimated action, or exploit the action with the higher estimated reward. Note that for evaluating the reward of the action down d, one time step has to be spent for evaluating the latency at time $t + 1$ when the number of containers has been decreased.

4.1.5 Optimization of Container Allocation as a Contextual Linear Bandit Problem.

In Algorithm 9, the reward at time t does not depend on the sequence of choices of the agent, but only of the choice of the agent at time t. That is why this reinforcement learning problem can be reduced as a contextual bandit problem. Before choosing an action, the agent observes the state of environment. The state of environment corresponds to a vector of observed variables, which depend on the workload w_t: the number of requests per second, the CPU occupancy rate, the RAM occupancy rate, and the latency l_t. The agent also knows the current number of containers c_t. Using these variables as a basis in a sliding time window, a context vector $\mathbf{x_t} \in \mathbb{R}^{\mathbf{D}}$ describing the state of the environment is built, where D is the number of contextual variables, and $\|\mathbf{x}_t\|_2 \leq 1$. For all actions $a \in \{s, d\}$, it is assumed that, there exists unknown weight vectors $\theta_a^* \in \mathbb{R}^D$ with $\|\theta_a^*\|_2 \leq 1$ so that $\forall t$:

$$\mathbb{E}[r_{a,t}|\mathbf{x}_t] = {\theta_a^*}^\top \mathbf{x}_t. \tag{4.4}$$

Using this linear assumption, the goal of the player is to minimize the pseudo-regret between the optimal linear policy and the linear policy played by the player:

$$R(2T) = \sum_{t=1;t=t+2}^{2T} \left({\theta_{a_t}^*}^\top \mathbf{x}_t - \theta_{a_t}(t)^\top \mathbf{x_t} \right), \tag{4.5}$$

where $\theta_{a_t^*}^*$ is the optimal parameter for the optimal action at time t, and $\theta_{a_t}(t)$ is the parameter at time t for the action a_t chosen by the player. To be consistent with the evaluation of the reward, which necessitates one additional time step after the choice of the action, the regret is evaluated at time $2T$ for T choices of actions.

Horizontal Scaling LINUCB

The proposed HSLINUCB algorithm (See Algorithm 10) is an extension of the LinUCB algorithm proposed in [Li et al., 2010a]; it includes a reactive action to increase the number of containers (line 8). LinUCB has a regret bound in the order of $O\left(D\sqrt{T\log T}\right)$ [Abbasi-yadkori et al., 2011], and hence HSLINUCB benefits of the same theoretical guarantee. The key idea of LinUCB algorithm is to apply online ridge regression to incoming data to obtain an estimate of the coefficients θ_a for $a = \{s, d\}$. At each time step the arm with the highest upper confidence bound of the reward $a_t = \arg\max_a(\theta_a^\top \mathbf{x}_t + b_a)$ is selected, where $b_a = \beta\sqrt{\mathbf{x}_t^\top \mathbf{A}_a^{-1}\mathbf{x}_t}$ is the standard deviation, β is a constant that depends on the probability of failure δ of the estimation done by the ridge regression, and $\mathbf{A}_a$ is the covariance matrix of the a-th arm context.

Algorithm 10 HSLINUCB for Optimization of Container Allocation

1: **Input:** $\delta \in (0, 1)$
2: $\beta \leftarrow 1 + \sqrt{\log(2/\delta)/2}$, $\forall a \in \{s, d\}$, $\mathbf{A}_a \leftarrow \mathbf{I}_D$, $v_a \leftarrow \mathbf{0}_D$, $\theta_a \leftarrow \mathbf{0}_D$
3: **for** $t \leftarrow 1, 2, ..., 2T$ **do**
4: observe $\mathbf{x}_t$,l_t
5: **for** $a \in \{s, d\}$ **do**
6: $\theta_a \leftarrow \mathbf{A}_a^{-1}\mathbf{v}_a$, $b_a \leftarrow \beta\sqrt{\mathbf{x}_t^\top \mathbf{A}_k^{-1}\mathbf{x}_t}$,
7: **end for**
8: **if** $l_t > l^*$ **then** play $a_t \leftarrow u$
9: **else** play arm $a_t \leftarrow \arg\max_a(\theta_a^\top \mathbf{x}_t + b_a)$
10: **end if**
11: l_{t+1} is revealed and $r_{a_t,t}$ is evaluated according to Equation 4.3
12: **if** $a_t \neq u$ **then**
13: $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_t\mathbf{x}_t^\top$,
14: $\mathbf{v}_a \leftarrow \mathbf{v}_a + r_{a_t,t}\mathbf{x}_t$
15: **end if**
16: **end for**

4.1.6 Experimentation

Cloud scaling has been extensively studied on simulated environments or on few private real implementations. Evaluation on a real environment is necessary to capture all the system dynamics. To provide reproducible results and make experiments, the complete installed environment and the implemented algorithms have been shared on a GitHub repository[4]. The following section provides a general overview but a more in-depth environment description is available on the GitHub repository.

Experimental Environment

The environment (Figure 4.1) is running inside virtual machines hosted on an Openstack Cloud. The agent server (4 VCPU/ 4 GB RAM) handles the reinforcement learning agent algorithm. It makes Cloud resource adaptation

[4]https://github.com/Orange-OpenSource/HSLinUCB

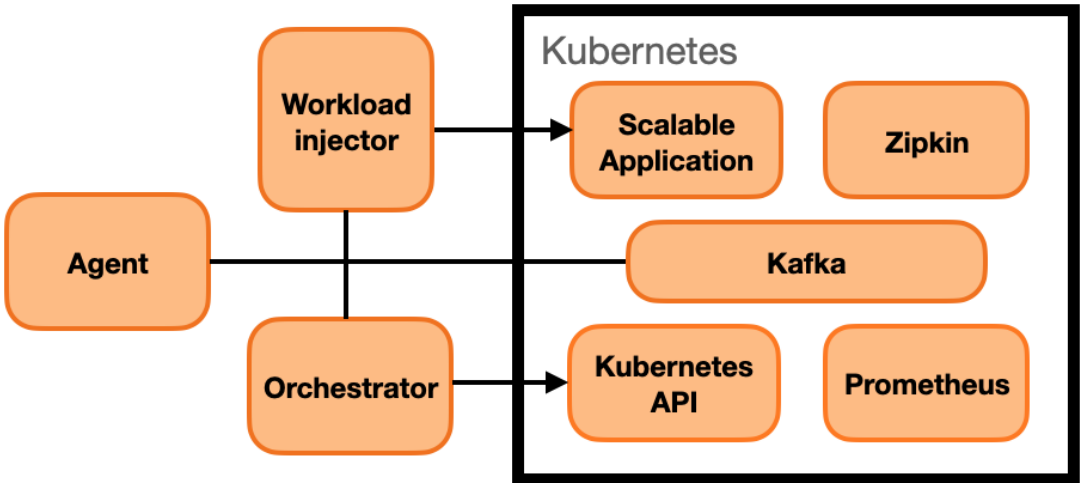


Figure 4.1: Overall architecture view

and drives the workload injector server. The workload injector is based on the Locust software, it allows to change the injected user number. The Kafka message bus allows the components to inter-operate with each other. The Kubernetes platform is composed of 5 servers each with 8 VCPU/ 16 GB RAM. A service MESH has been implemented inside the Kubernetes platform with the ISTIO framework. It implements by default a system monitoring with Prometheus and a latency monitoring with Zipkin. The orchestrator component collects metrics, computes contexts for the agent and executes scaling operation on Kubernetes. The orchestrator supports the raised context storage into an HDF5 file allowing to create a dataset. When a change on the container number is requested, the orchestrator realizes the modification on the Kubernetes API and waits until the monitored metrics reflect the change. This waiting time allows for system stabilization accounting. During real experiments, the contexts and actions are respectively retrieved and executed on the platform. During simulation, the contexts are retrieved from real stored datasets and actions are simulated through changes in a variable which describes the number of containers.

Evaluated Web Application. The scalable application is an Apache server running a python script with the Kubernetes configuration (the units correspond to resource units in Kubernetes) set to Requests, Limits : $[CPU : 500m, RAM : 512Mi]$. The python code simply performs some array computation by adding and removing elements a number of times specified in the client request. In our experiments, an injected user always asks the component to perform 200 cycles of array manipulation and waits for a randomly chosen time between 0 and 2 seconds before sending another request. With these settings a new container is needed approximately every 10 new users injected. As the maximum number of injected user is around 50, these settings allow to provide a reasonable elasticity while avoiding an important number of injection servers.

Context, Actions, Reward Function and Common Experimentation Settings. The orchestrator component permanently collects the observed latency and system metrics and keeps them inside an history buffer. The system

metrics are collected every 5 seconds. For each request sent to the Web application, the history buffer contains the response latency with its timestamp. For the system metrics, each entry in the history buffer contains the mean of the CPU and RAM requests and limits[5] over all currently allocated containers. When a context is requested by the agent, the orchestrator is computing, from the history buffer, the 95^{th} percentile latency and the number of requests per second within a 10 seconds time window. For the system metrics the mean of the CPU and RAM requests and limits are computed on the last 10 entries of the history buffer. The following context vector is used by all algorithms: $\mathbf{x}_t$ = [number of containers, 95^{th} percentile latency, requests per second, cpu request mean, cpu limit mean, ram request mean, ram limit mean].

As defined in Section 4.1.4, three actions are used: the up action is automatically handled, while stay and down are handled by the reinforcement algorithms. All experiments have been repeated 10 times to capture means and confidence intervals. The latency l^* is set to 600ms.

Baselines.

Different algorithms have been implemented as baselines for the simulated and/or the real experiments. All the algorithms parameters have been experimentaly tuned with a grid-search.

Q-Learning. To reduce the state-action space, Q-Learning [Sutton and Barto, 2018] uses the following context information: [number of containers, 95^{th} percentile latency rounded to the hundred, requests per second rounded to the ten, mean cpu request rounded to the decimal, mean ram request rounded to the decimal]. The learning rate α is set to 0.1 and the discount factor γ is set to 0.9. ϵ-greedy starts in pure exploration and increases to pure exploitation by 0.000166 at each step.

Deep-Q-Learning. Deep-Q-Learning [Mnih et al., 2015] uses DQN architecture with experience replay memory. The neural network is composed of 2 hidden layers with 24 and 12 neurons using Relu activation function and Huber loss function. The context variables are scaled between $[0, 1]$ for neural network compliance. The learning rate α is set to 0.01, the discount factor γ is set to 0.9 and the batch size is set to 50. For the simulated experiments the target network is updated every 100 steps. ϵ-greedy starts in pure exploration and increases to pure exploitation by 0.000333 at each step. For the experiments on the real environment, as there are less experimentation steps, the target network is updated every 50 steps, and the increase of ϵ is set to 0.00125.

Kubernetes Threshold Algorithm. To evaluate a threshold algorithm, the Kubernetes horizontal pod auto-scaler algorithm has been implemented[6]. The threshold auto-scaler uses the metrics context directly from Kubernetes and the threshold level was set to the average CPU = 470m. With this value, experimentally tuned, a new container is created approximately every 10 new users injected allowing the algorithm to stick with the environment setting.

Oracle. A simulated optimal policy (see Section 4.1.3) is also used as a baseline for the simulated experiments.

[5] https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[6] https://github.com/kubernetes/kubernetes/tree/master/pkg/controller/podautoscaler

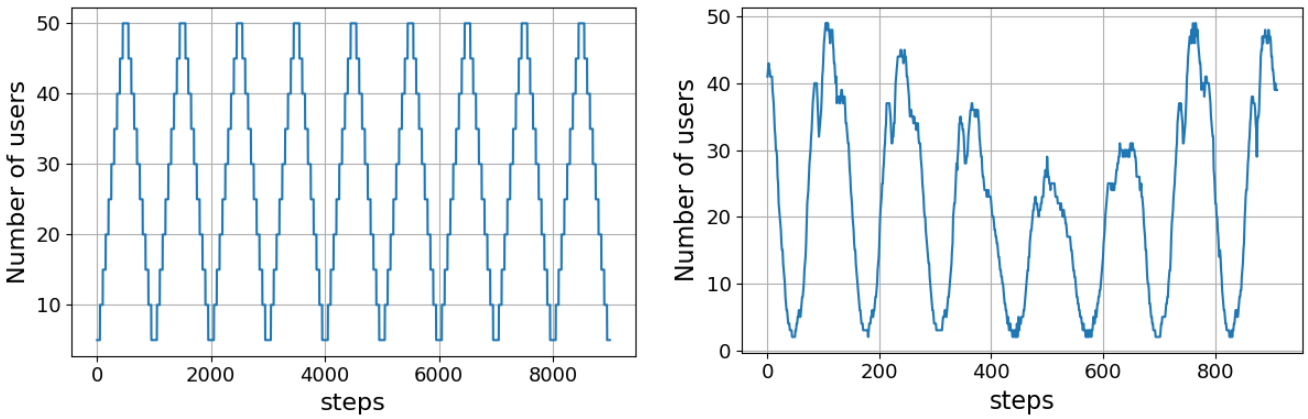


Figure 4.2: (a) : 9000 steps workload pattern for simulated experiments. (b): 900 steps workload pattern extracted from a week of Wikibench traces from 9/18/2007 to 09/24/2007 for real experiments.

This policy can only be computed for a known dataset generated by a real platform and then used for simulation. To compute this policy from a dataset: for a given workload (number of users), the minimal number of containers that satisfies $l_t \leq l^*$ is searched in the dataset.

Experiments in Simulated Environment

The simulated environment is not using classical Cloud simulator. It is based on Python code which simulates a Cloud environment by reading metrics from a dataset and changing the virtual number of containers regarding the action chosen by the evaluated algorithm.

Dataset Description For all experiments done in simulation, a dataset has been used for providing to algorithms the contexts (see Section 4.1.6) and the corresponding latencies. The dataset is a set of records of different contexts, and latencies built on variations of the injected user numbers in the real platform. Users varied from 5 to 50 with a step of 5 users. For each number of users, the number of containers varied from 1 to 10. As expected, for the same injected user number and number of containers there are different values in the captured contexts since a random waiting time is implemented at the injected user side (see Section 4.1.6). This allows us to collect a set of metrics to build the dataset. To analyse the algorithm behavior when learning on data and acting on unseen ones, two different datasets D_1, D_2 have been captured on the same real environment. For each tuple [user number;container number], 20 and 40 contexts have been stored respectively in the dataset D_1 and D_2.

Workload Pattern. In simulated experiments the workload pattern used is depicted in Figure 4.2(a). This synthetic pattern is a 9000 steps repetitive triangular pattern with the number of users increasing/decreasing by 5 users every 50 steps.

Cold-start and Hot-start Evaluation Protocol. In this section the behavior of baselines and HSLinUCB are evaluated with different starting points. First, each algorithm is evaluated from a cold-start where the agent has no knowledge about the environment with context information extracted from the dataset D_1. Then, the experiment is restarted on the same dataset D_1 but with a hot-start where the agent uses the knowledge learnt from the cold-start experiment. Finally, the experiment is restarted with a hot-start using a new dataset D_2 and using the knowledge from the cold-start experiment on dataset D_1. ϵ-greedy exploration is activated for Q-learning and Deep Q-learning only for the first cold-start experiment. Pure exploitation is used for the second and the third experiments. The first experiment will demonstrate how the agent performs on cold-start while learning a new environment. The second, how the agent performs on an already learnt environment. The last, if the agent can perform self-adaptation on un-learnt contexts.

Cold-start and Hot-start Evaluation Results. In the case of cold start, HSLinUCB outperforms the other algorithms by quickly converging to the Oracle policy in around 500 steps (Figure 4.3). DQN requires less steps than Q-Learning thanks to the Neural Network that captures the model. HSLinUCB realizes 183 times less errors than Q-Learning and 92 times less than Deep Q-Learning while allocating a quite similar mean number of containers (Table 4.1). This is due to the efficient exploration, and to the robust ridge regression used in LinUCB.

In the case of hot start, for seen and unseen contexts HSLinUCB obtains quite similar results than DQN, which benefits for the high learning performances of Deep Neural Networks. Q-learning obtains worse results than other baselines on unseen contexts while it obtains similar results on seen contexts. This is due to the fact that for unseen contexts, Q-learning faces new contexts and does not know what is the optimal action to perform.

Table 4.1: Numerical results for different algorithms in simulator.

Mode	**Algorithm**	**Mean** l	**Mean errors** l	**Mean** c
	Oracle	286 ± 1.95	0 ± 0	3 ± 0.0248
32.8cm**Cold-start**	HSLinUCB	$\mathbf{291 \pm 1.64}$	$\mathbf{6.1 \pm 0.584}$	2.97 ± 0.025
	DQN	345 ± 3.72	560 ± 14.7	3 ± 0.026
	Q-learning	438 ± 4.16	1118 ± 13.1	$\mathbf{2.72 \pm 0.025}$
32.8cm**Hot-start seen contexts**	HSLinUCB	281 ± 1.83	$\mathbf{0 \pm 0}$	3.02 ± 0.024
	DQN	$\mathbf{266 \pm 1.57}$	0.1 ± 0.186	3.12 ± 0.0265
	Q-learning	304 ± 1.74	0.6 ± 0.303	$\mathbf{2.93 \pm 0.0265}$
32.8cm**Hot-start unseen contexts**	HSLinUCB	255 ± 1.65	2.0 ± 0	3.23 ± 0.023
	DQN	$\mathbf{252 \pm 1.47}$	$\mathbf{0 \pm 0}$	3.24 ± 0.026
	Q-learning	324 ± 1.77	226 ± 30.4	$\mathbf{2.91 \pm 0.026}$

To conclude on simulated experiments, HSLinUCB provides better results in cold-start, performs almost as good as the Oracle with just a few over container allocation in hot-start with seen contexts. It is also quite as good as DQN in hot start with unseen contexts.

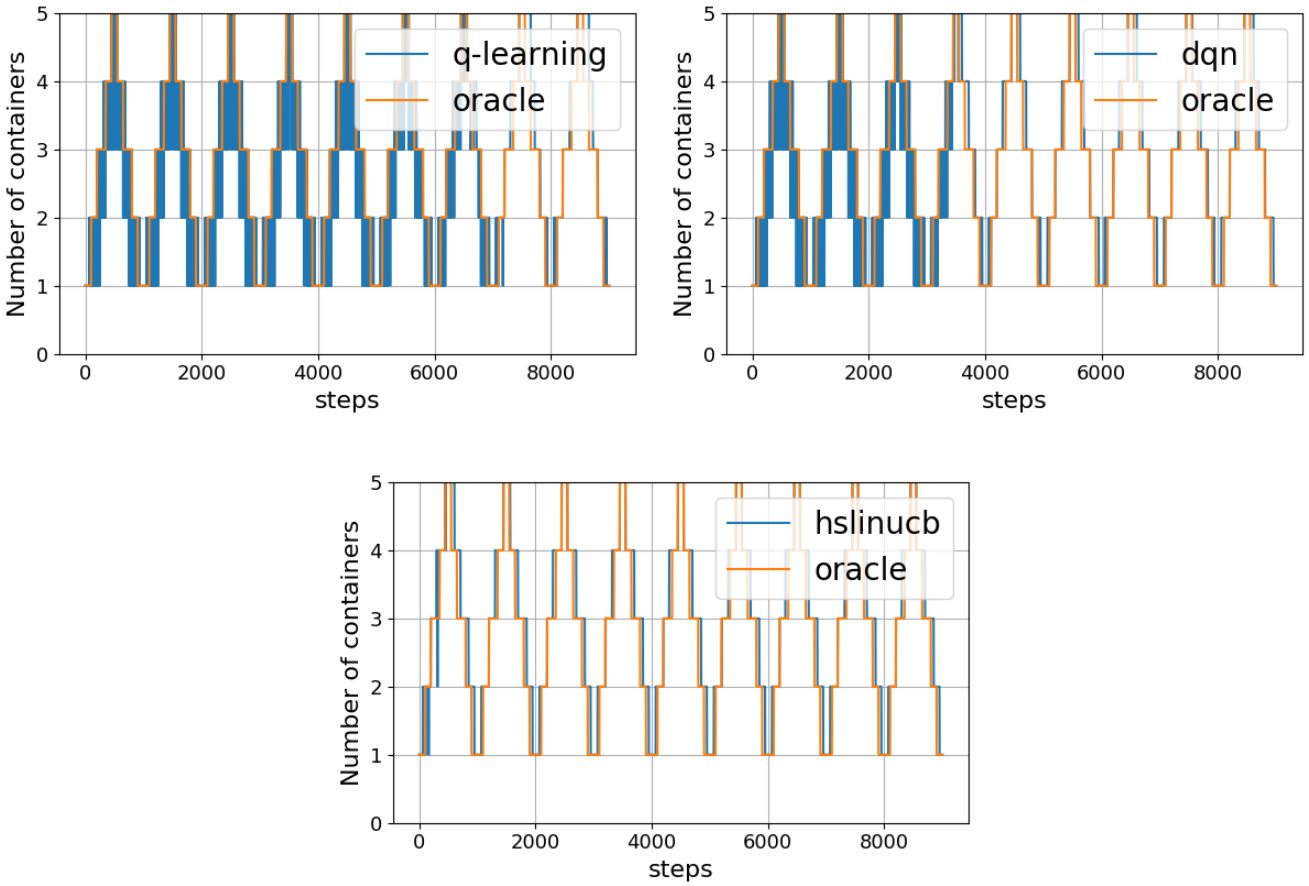


Figure 4.3: Number of containers on Cold-start for Q-Learning, Deep Q-Learning and HSLinUCB.

Experimental Results on a Real Platform

Workload Pattern. A real pattern of traffic containing HTTP requests traces is extracted from one week of Wikipedia website [Urdaneta et al., 2009]. In order to obtain a reasonable computational time, the seven days of traffic have been reproduced and translated in 900 steps load variation (Figure 4.2(b)). Then, the load variation of the Wikipedia servers has been reproduced on the real platform with users injections.

Real Environment Evaluation Protocol. In this section the best algorithms from the previous simulated experiments are compared in cold-start to the standard threshold based auto-scaler. Moreover, in order to illustrate the expected performance of DQN trained in a simulated environment and deployed in a real environment, hot-start DQN trained in simulation (Section 4.1.6) is deployed in the real environment. Finally, notice that contrary to the simulated experiments the Oracle can not be estimated in real environment. Indeed, in a simulated environment, the Oracle is computed from datasets and the experiments use the same datasets. This allows a stable baseline for algorithm policy comparison. In a real environment it is impossible to perfectly estimate the future, hence it is impossible to build a stable baseline and to accurately compare an algorithm policy to an unknown perfect one.

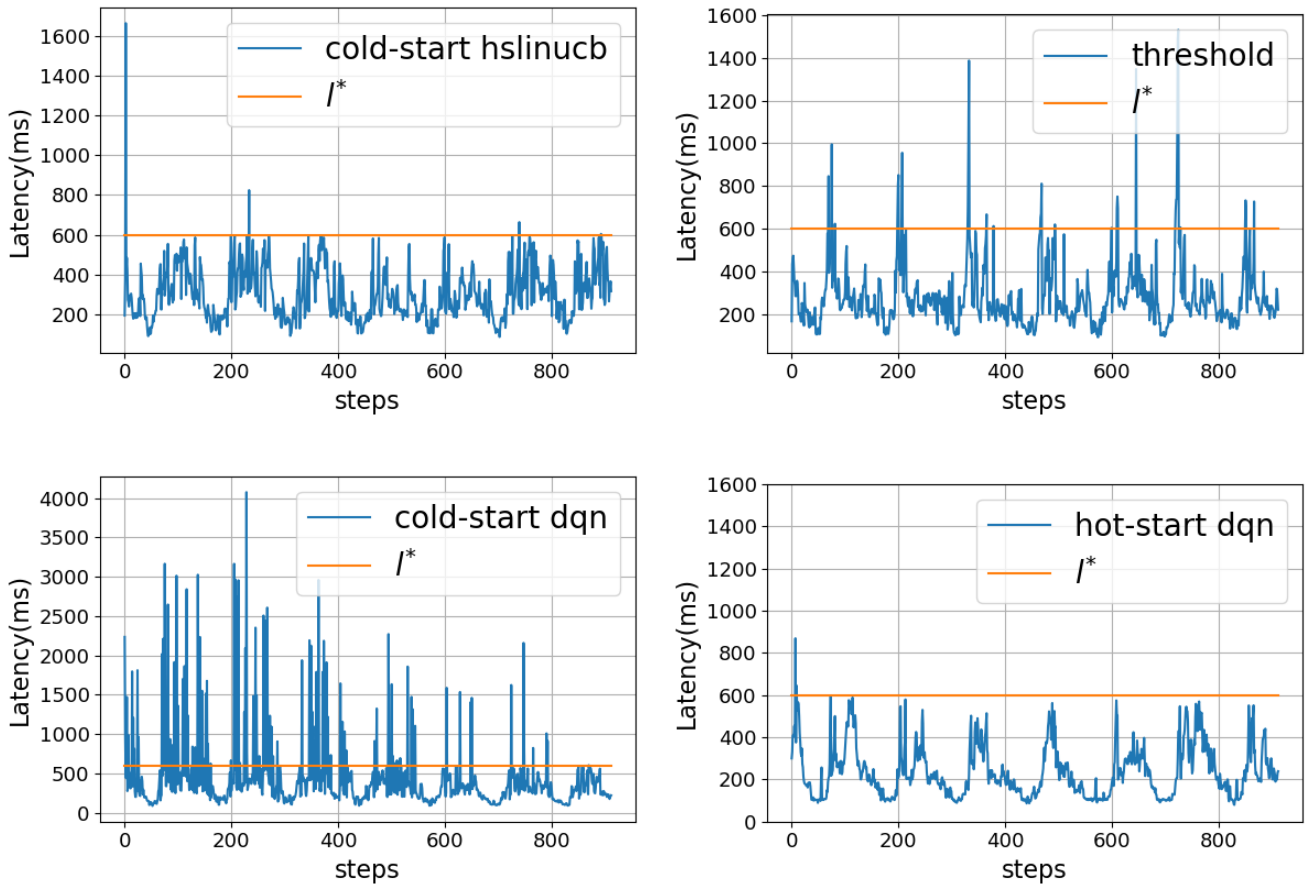


Figure 4.4: Latency captured on a real platform for HSLinUCB, threshold, Deep Q-Learning (vertical scale is different) in cold-start, and Deep Q-Learning trained in simulations, then used in hot-start.

Real Environment Evaluation Results. The experiments duration on a real environment is around 15 hours for the threshold algorithm, 16 hours for HSLinUCB and 37 hours for DQN. These differences are explained by the number of changes on the Cloud performed by each algorithm mainly during the exploration stages. While our solution is designed for a unlimited number of containers, the workload has been scaled in the experiments to limit the running time with a maximum of about 5 containers. Figure 4.4 depicts the latency results for the evaluated algorithms and the Table 4.2 presents some numerical results. Cold-start HSLinUCB achieves to maintain the latency under l^* with 4.4 SLA violations on average which is 7.25 times less than the threshold algorithm and 26.3 times less than Cold-start DQN. In cold-start, HSLinUCB outperforms the other algorithms while allocating fewer

Table 4.2: Numerical results for different algorithms on a real platform.

Algorithm	**Mean** l	**Mean errors** l	**Mean** c
Cold-start HSLinUCB	285 ± 6.55	4.4 ± 0.495	2.56 ± 0.0768
Cold-start DQN	428 ± 15.9	109 ± 6.89	$\mathbf{2.39 \pm 0.07}$
Threshold	279 ± 7.45	31.9 ± 2.75	2.6 ± 0.086
Hot-start DQN	$\mathbf{237 \pm 7}$	$\mathbf{1.1 \pm 0.51}$	3 ± 0.057

containers than the threshold algorithm. Even optimally tuned and configured to react as quickly as possible, the threshold auto-scaler reacts later on workload increase which leads to SLA violation. It also reacts later on workload decrease which drives to over allocation. In the case of cold start, DQN is still penalized by its exploration strategy as it explores more the down action than HSLinUCB which results in less number of containers on average and more SLA violations on average. In the case of hot start, despite the fact that the synthetic workload pattern and the real load pattern look alike (Figure 4.2), hot-start DQN over-allocates (20 % more containers than HSLinUCB), which explains its low mean latency and mean number of errors (Table 4.2).

To conclude on real experiments, HSLinUCB provides better results in cold-start without the complex and costly parameters tuning of the threshold algorithm. The HSLinUCB internal exploration strategy allows to quickly follow the workload with few SLA violations while optimizing the container usage (less over-provisioning). DQN, thanks to its neural network is a good performer but is penalized by its random exploration strategy.

4.1.7 Conclusion

In this paper, the horizontal scaling problem in container-based environments has been solved using contextual bandits. A mathematical formulation has been presented. A new algorithm HSLinUCB based on LinUCB has been proposed. A simulated environment with synthetic workload pattern and a real platform with the Wikipedia workload pattern has been used to evaluate the approach. The complete experimental environment has been made publicly available through a GitHub repository. An intensive comparison between HSLinUCB and state of the art algorithms has been made with different starting points. All the experiments conclude that the proposed algorithm HSLinUCB performs very well compared to other state of the art algorithms and is close to the optimal policy in simulation. On a real platform, HSLinUCB is able to quickly learn from a cold start allowing it to be used for a new Web application deployment or for a minor and major changes in existing application. In this work the container resource configuration and the Kubernetes node computation power are always the same during an experiment. The workload, even issued from a production trace, is regular and does not contain some extreme or sudden changes. As future works, we plan to study these workload and Cloud stationarity changes. Although HSLinUCB continue to work with more injected users and more managed containers, the current environment settings allows HSLinUCB to satisfy the SLA by adding or removing a container one by one. Hence we also plan to evaluate the HSLinUCB performance with more actions and explore how this approach could be applied to vertical scaling.

4.2 Bandit Forest for optimizing marketing campaigns

Abstract. To address the contextual bandit problem, we propose an online *random forest* algorithm. The analysis of the proposed algorithm is based on the sample complexity needed to find the optimal decision stump. Then, the

decision stumps are recursively stacked in a random collection of decision trees, BANDIT FOREST. We show that the proposed algorithm is near optimal. The dependence of the sample complexity upon the number of contextual variables is logarithmic. The computational cost of the proposed algorithm with respect to the time horizon is linear. These analytical results allow the proposed algorithm to be efficient in real applications, where the number of events to process is huge, and where we expect that some contextual variables, chosen from a large set, have potentially non-linear dependencies with the rewards. In the experiments done to illustrate the theoretical analysis, BANDIT FOREST obtain promising results in comparison with state-of-the-art algorithms.

4.2.1 Targeted use case: optimizing messages of marketing campaigns

The aim of a marketing campaign is to promote products and services to prospects (and customers, which have some products and services but not those promoted). Optimizing marketing campaigns consists in choosing the best assignment between prospects and current marketing campaigns in order to maximize the number of positive outcomes of contacts. Here we focus on a particular marketing optimization problem, which is choosing the message of a marketing campaign (figure 5.1). For instance, for promoting a television offer, should one communicate on sports, on movies, on the number of television channels... ? In order to minimize the marketing pressure on prospects, a strict requirement is to contact at most one time a prospect for a given marketing campaign. Hence, only the outcome of the played marketing campaign is known.

Figure 4.5: Finding the best message of a marketing campaign.

Moreover, the prospects which are usually users or customers are often well known, and one disposes of a profile for each one. This profile contains information about the prospect, for instance his usages of products and services, and provides valuable information about his preference. Indeed, the prospects and customers have different preferences, and could be more sensible to different messages (figure 4.6). For instance a message promoting sports could have different outcomes on rugby lovers than on other people.

Finally, the marketing departments, which are also in charge of building new offers, are interested to understand which offers attract which customers. The choice of decision tree based algorithm, which can be easily interpreted, is well suited for handling optimization marketing campaigns.

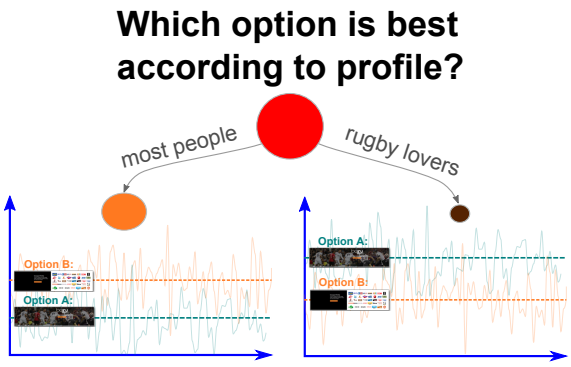


Figure 4.6: Contextual best message identification

4.2.2 Introduction

By interacting with streams of events, machine learning algorithms are used for instance to optimize the choice of ads on a website, to choose the best human machine interface, to recommend products on a web shop, to insure self-care of set top boxes, to assign the best wireless network to mobile phones. With the now rising *internet of things*, the number of decisions (or actions) to be taken by more and more autonomous devices further increases. In order to control the cost and the potential risk to deploy a lot of machine learning algorithms in the long run, we need scalable algorithms which provide strong theoretical guarantees.

Most of these applications necessitate to take and optimize decisions with a partial feedback. Only the reward of the chosen decision is known. Does the user click on the proposed ad ? The most relevant ad is never revealed. Only the click on the proposed ad is known. This well known problem is called *multi-armed bandits* (MAB). In its most basic formulation, it can be stated as follows: there are K decisions, each having an unknown distribution of bounded rewards. At each step, one has to choose a decision and receives a reward. The performance of a MAB algorithm is assessed in terms of *regret* (or opportunity loss) with regards to the unknown optimal decision. Optimal solutions have been proposed to solve this problem using a stochastic formulation in [Auer et al., 2002a], using a Bayesian formulation in [Kaufmann et al., 2012], or using an adversarial formulation in [Auer et al., 2002b]. While these approaches focus on the minimization of the expected regret, the PAC setting (see [Valiant, 1984]) or the (ϵ, δ)-best-arm identification, focuses on the sample complexity (i.e. the number of time steps) needed to find an ϵ-approximation of the best arm with a failure probability of δ. This formulation has been studied for MAB problem in [Even-Dar et al., 2002, Bubeck et al., 2009], for dueling bandit problem in [Urvoy et al., 2013], and for linear bandit problem in [Soare et al., 2014].

Several variations of the MAB problem have been introduced in order to fit practical constraints coming from ad serving or marketing optimization. These variations include for instance the death and birth of arms in [Chakrabarti et al., 2008], the availability of actions in [Kleinberg et al., 2010] or a drawing without replacement in [Féraud and Urvoy, 2012, Féraud and Urvoy, 2013]. But more importantly, in most of these applications, a rich contextual

information is also available. For instance, in ad-serving optimization we know the web page, the position in the web page, or the profile of the user. This contextual information must be exploited in order to decide which ad is the most relevant to display. Following [Langford and Zhang, 2007, Dudik et al., 2011], in order to analyze the proposed algorithms, we formalize below the *contextual bandit problem* (see Algorithm 11).

Let $\mathbf{x_t} \in \{0,1\}^M$ be a vector of binary values describing the environment at time t. Let $\mathbf{y_t} \in [0,1]^K$ be a vector of bounded rewards at time t, and $y_{k_t}(t)$ be the reward of the action (decision) k_t at time t. Let $D_{x,y}$ be a joint distribution on $(\mathbf{x}, \mathbf{y})$. Let $\mathcal{A}$ be a set of K actions. Let $\pi : \{0,1\}^M \to \mathcal{A}$ be a policy, and Π the set of policies.

Algorithm 11 The contextual bandit problem

repeat
  $(\mathbf{x_t}, \mathbf{y_t})$ is drawn according to $D_{x,y}$
  $\mathbf{x_t}$ is revealed to the player
  The player chooses an action $k_t := \pi_t(\mathbf{x_t})$
  The reward $y_{k_t}(t)$ is revealed
  The player updates its policy π_t
  $t := t + 1$
until $t = T$

Notice that this setting can easily be extended to categorical variables through a binary encoding. The optimal policy π^* maximizes the expected gain:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{D_{x,y}} \left[y_{\pi(\mathbf{x_t})} \right]$$

Let $k_t^* = \pi^*(\mathbf{x_t})$ be the action chosen by the optimal policy at time t. The performance of the player policy is assessed in terms of expectation of the accumulated regret against the optimal policy with respect to $D_{x,y}$:

$$\mathbb{E}_{D_{x,y}} [R(T)] = \sum_{t=1}^{T} \mathbb{E}_{D_{x,y}} \left[y_{k_t^*}(t) - y_{k_t}(t) \right] ,$$

where $R(T)$ is the accumulated regret at time horizon T.

4.2.3 Our contribution

Decision trees work by partitioning the input space in hyper-boxes. They can be seen as a combination of rules, where only one rule is selected for a given input vector. Finding the optimal tree structure (i.e. the optimal combination of rules) is NP-hard. For this reason, a greedy approach is used to build the decision trees offline (see [Breiman et al., 1984]) or online (see [Domingos and Hulten, 2000]). The key concept behind the greedy decision trees is the decision stump. While Monte-Carlo Tree Search approaches (see [Kocsis and Szepesvári, 2006a]) focus on the regret minimization (i.e. maximization of gains), the analysis of proposed algorithms is based on the sample complexity needed to find the optimal decision stump with high probability. This formalization facilitates the analysis

of decision tree algorithms. Indeed, to build a decision tree under limited resources, one needs to eliminate most of possible branches. The sample complexity is the stopping criterion we need to stop exploration of unpromising branches. In BANDIT FOREST, the decision stumps are recursively stacked in a random collection of L decision trees of maximum depth D. We show that BANDIT FOREST algorithm is near optimal with respect to a strong reference: a *random forest* built knowing the joint distribution of the contexts and rewards.

In comparison to algorithms based on search of the best policy from a finite set of policies (see [Auer et al., 2002b, Dudik et al., 2011, Agarwal et al., 2014]) our approach has several advantages. First, we take advantage of the fact that we know the structure of the set of policies to obtain a linear computational cost with respect to the time horizon T. Second, as our approach does not need to store a weight for each possible tree, we can use deeper rules without exceeding the memory resources. In comparison to the approaches based on a linear model (see LINUCB in [Li et al., 2010a]), our approach also has several advantages. First, it is better suited for the case where the dependence between the rewards and the contexts is not linear. Second, the dependence of regret bounds of the proposed algorithms on the number of contextual variables is in the order of $O(\log M)$ while the one of linear bandits is in $O(\sqrt{M})$ [7]. Third, its computational cost with respect to time horizon in $O(LMDT)$ allows to process large set of variables, while linear bandits are penalized by the update of a $M \times M$ matrix at each update, which leads to a computational cost in $O(KM^2T)$.

4.2.4 The decision stump

In this section, we consider a model which consists of a decision stump based on the values of a single contextual variable, chosen from at set of M binary variables.

A gentle start

In order to explain the principle and to introduce the notations, before describing the decision stump used to build BANDIT FOREST, we illustrate our approach on a toy problem. Let k_1 and k_2 be two actions. Let x_{i_1} and x_{i_2} be two binary random variables, describing the context. In this illustrative example, the contextual variables are assumed to be independent. Relevant probabilities and rewards are summarized in Table 5.1. $\mu_{k_2}^{i_1}|v$ denotes the conditional expected reward of the action k_2 given $x_{i_1} = v$, and $P(x_{i_1} = v)$ denotes the probability to observe $x_{i_1} = v$.

We compare the strategies of different players. Player 1 only uses uncontextual expected rewards, while Player 2 uses the knowledge of x_{i_1} to decide. According to Table 5.1, the best strategy for Player 1 is to always choose action k_2. His expected reward will be $\mu_{k_2} = 7/16$. Player 2 is able to adapt his strategy to the context: his best strategy is

[7] In fact the dependence on the number of contextual variables of the gap dependent regret bound is in $O(M^2)$ (see Theorem 5 in [**?**]).

Table 4.3: The mean reward of actions k_1 and k_2 knowing each context value, and the probability to observe this context.

	v_0	v_1
$\mu^{i_1}_{k_1}\|v$	0	1
$\mu^{i_1}_{k_2}\|v$	$3/5$	$1/6$
$P(x_{i_1}=v)$	$5/8$	$3/8$
$\mu^{i_2}_{k_1}\|v$	$1/4$	$3/4$
$\mu^{i_2}_{k_2}\|v$	$9/24$	$5/8$
$P(x_{i_2}=v)$	$3/4$	$1/4$

to choose k_2 when $x_{i_1}=v_0$ and k_1 when $x_{i_1}=v_1$. According to Table 5.1, his expected reward will be:

$$\begin{aligned}\mu^{i_1} &= P(x_{i_1}=v_0)\cdot\mu^{i_1}_{k_2}|v_0 + P(x_{i_1}=v_1)\cdot\mu^{i_1}_{k_1}|v_1\\ &= \mu^{i_1}_{k_2,v_0} + \mu^{i_1}_{k_1,v_1} = 3/4\ ,\end{aligned}$$

where $\mu^{i_1}_{k_2,v_0}$ and $\mu^{i_1}_{k_1,v_1}$ denote respectively the expected reward of the action k_2 and $x_{i_1}=v_0$, and the expected reward of the action k_1 and $x_{i_1}=v_1$. Whatever the expected rewards of each value, the player, who uses this knowledge, is the expected winner. Indeed, we have:

$$\mu^i = \max_k \mu^i_{k,v_0} + \max_k \mu^i_{k,v_1} \geq \max_k \mu_k$$

Now, if a third player uses the knowledge of the contextual variable x_{i_2}, his expected reward will be:

$$\mu^{i_2} = \mu^{i_2}_{k_2,v_0} + \mu^{i_2}_{k_1,v_1} = 15/32$$

Player 2 remains the expected winner, and therefore x_{i_1} is the best contextual variable to decide between k_1 and k_2.

The best contextual variable is the one which maximizes the expected reward of best actions for each of its values. We use this principle to build a reward-maximizing decision stump.

Variable selection

Let $\mathcal{V}$ be the set of variables, and $\mathcal{A}$ be the set of actions. Let $\mu^i_k|v = \mathbb{E}_{D_y}[y_k \cdot \mathbb{1}_{x_i=v}]$ be the expected reward of the action k conditioned to the observation of the value v of the variable x_i. Let $\mu^i_{k,v} = P(v).\mu^i_k|v = \mathbb{E}_{D_{x,y}}[y_k \cdot \mathbb{1}_{x_i=v}]$ be the expected reward of the action k and the value v of the binary variable x_i. The expected reward when using variable x_i to select the best action is the sum of expected rewards of the best actions for each of its possible values:

$$\mu^i = \sum_{v\in\{0,1\}} P(v).\max_k \mu^i_k|v = \sum_{v\in\{0,1\}} \max_k \mu^i_{k,v}$$

The optimal variable to be used for selecting the best action is: $i^* = \arg\max_{i \in \mathcal{V}} \mu^i$.

The algorithm VARIABLE SELECTION chooses the best variable. The $\mathrm{Round\text{-}robin}$ function sequentially explores the actions in $\mathcal{A}$ (see Algorithm 12 line 4). Each time t_k the reward of the selected action k is unveiled, the estimated expected rewards of the played action k and observed values $\hat{\mu}^i_{k,v}$ and all the estimated rewards of variables $\hat{\mu}^i$ are updated (see VE function lines 2-7). This *parallel exploration strategy* allows the algorithm to explore efficiently the variable set. When all the actions have been played once (see VE function line 8), irrelevant variables are eliminated if:

$$\hat{\mu}^{i'} - \hat{\mu}^i + \epsilon \geq 4\sqrt{\frac{1}{2t_k} \log \frac{4KMt_k^2}{\delta}}\,, \tag{4.6}$$

where $i' = \arg\max_i \hat{\mu}^i$, and t_k is the number of times the action k has been played.

Algorithm 12 VARIABLE SELECTION

Inputs: $\epsilon \in [0, 1)$, $\delta \in (0, 1]$
Output: an ϵ-approximation of the best variable
1: $t := 0$, $\forall k\ t_k := 0$, $\forall (i, k, v)\ \hat{\mu}^i_{k,v} := 0$, $\forall i\ \hat{\mu}^i := 0$
2: **repeat**
3: Receive the context vector $\mathbf{x_t}$
4: Play $k := \mathrm{Round\text{-}robin}\,(\mathcal{A})$
5: Receive the reward $y_k(t)$
6: $t_k := t_k + 1$
7: $\mathcal{V}$:=VE$(t_k, k, \mathbf{x_t}, y_k, \mathcal{V}, \mathcal{A})$
8: $t := t + 1$
9: **until** $|\mathcal{V}| = 1$

1: **Function** VE$(t, k, \mathbf{x_t}, y_k, \mathcal{V}, \mathcal{A})$
2: **for** each remaining variable $i \in \mathcal{V}$ **do**
3: **for** each value v **do**
4: $\hat{\mu}^i_{k,v} := \frac{y_k}{t} \mathbb{1}_{x_i = v} + \frac{t-1}{t} \hat{\mu}^i_{k,v}$
5: **end for**
6: $\hat{\mu}^i := \sum_{v \in \{0,1\}} \max_k \hat{\mu}^i_{k,v}$
7: **end for**
8: **if** $k := \mathrm{LastAction}(\mathcal{A})$ **then**
9: Remove irrelevant variables from $\mathcal{V}$ according to equation 4.6, or 4.8
10: **end if**
11: **return** $\mathcal{V}$

The parameter $\delta \in (0, 1]$ corresponds to the probability of failure. The use of the parameter ϵ comes from practical reasons. The parameter ϵ is used in order to tune the convergence speed of the algorithm. In particular, when two variables provide the highest expected reward, the use of $\epsilon > 0$ ensures that the algorithm stops. The value of ϵ has to be in the same order of magnitude as the best mean reward we want to select. In the analysis of algorithms, we will consider the case where $\epsilon = 0$. Lemma 4 analyzes the sample complexity (the number of iterations before stopping) of VARIABLE SELECTION.

Lemma 4. *when $K \geq 2$, $M \geq 2$, and $\epsilon = 0$, the sample complexity of* VARIABLE SELECTION *needed to obtain*

$\mathbb{P}\left(i' \neq i^*\right) \leq \delta$ *is:*

$$t^* = \frac{64K}{\Delta_1^2} \log \frac{8KM}{\delta \Delta_1}, \text{ where } \quad \Delta_1 = \min_{i \neq i^*} (\mu^{i^*} - \mu^i)$$

Lemma 4 shows that the dependence of the sample complexity needed to select the optimal variable is in $O(\log M)$. This means that VARIABLE SELECTION can be used to process large set of contextual variables, and hence can be easily extended to categorical variables, through a binary encoding with only a logarithmic impact on the sample complexity. Finally, Lemma 5 shows that the variable selection algorithm is optimal up to logarithmic factors.

Lemma 5. *There exists a distribution $D_{x,y}$ such that any algorithm finding the optimal variable i^* has a sample complexity of at least:*

$$\Omega\left(\frac{K}{\Delta_1^2} \log \frac{1}{\delta}\right)$$

Action selection

To complete a decision stump, one needs to provide an algorithm which optimizes the choice of the best action knowing the selected variable. Any stochastic bandit algorithm such as UCB (see [Auer et al., 2002a]), TS (see [Thompson, 1933, Kaufmann et al., 2012]) or BESA (see [Baransi et al., 2014]) can be used. For the consistency of the analysis, we choose SUCCESSIVE ELIMINATION in [Even-Dar et al., 2002, Even-Dar et al., 2006] (see Algorithm 13), that we have renamed ACTION SELECTION. Let $\mu_k = \mathbb{E}_{D_y}[y_k]$ be the expected reward of the action k taken with respect to D_y. The estimated expected reward of the action is denoted by $\hat{\mu}_k$.

Algorithm 13 ACTION SELECTION

Inputs: $\epsilon \in [0, 1)$, $\delta \in (0, 1]$
Output: an ϵ-approximation of the best arm

$t := 0$, $\forall k\ \hat{\mu}_k := 0$ and $t_k := 0$
repeat
  Play $k := \text{Round-robin}\,(A)$
  Receive the reward $y_k(t)$
  $t_k := t_k + 1$
  $\mathcal{A} :=$AE$(t_k, k, y_k(t), \mathcal{A})$
  $t := t + 1$
until $|\mathcal{A}| = 1$

The irrelevant actions in the set $\mathcal{A}$ are successively eliminated when:

$$\hat{\mu}_{k'} - \hat{\mu}_k + \epsilon \geq 2\sqrt{\frac{1}{2t_k} \log \frac{4Kt_k^2}{\delta}}\,, \tag{4.7}$$

where $k' = \arg\max_k \hat{\mu}_k$, and t_k is the number of times the action k has been played.

1: **Function** AE$(t, k, \mathbf{x_t}, y_k, \mathcal{A})$
2: $\hat{\mu}_k := \frac{y_k}{t} + \frac{t-1}{t}\hat{\mu}_k$
3: **if** $k :=$LastAction$(\mathcal{A})$ **then**
4: Remove irrelevant actions from $\mathcal{A}$ according to equation 4.7, or 4.9
5: **end if**
6: **return** $\mathcal{A}$

Lemma 6. *when $K \geq 2$, and $\epsilon = 0$, the sample complexity of* ACTION SELECTION *needed to obtain $\mathbb{P}(k' \neq k^*) \leq \delta$ is:*

$$t^* = \frac{64K}{\Delta_2^2} \log \frac{4K}{\delta \Delta_2}, \textit{ where } \quad \Delta_2 = \min_k (\mu_{k^*} - \mu_k).$$

The proof of Lemma [**?**] is the same than the one provided for SUCCESSIVE ELIMINATION in [Even-Dar et al., 2006]. Finally, Lemma 7 states that the action selection algorithm is optimal up to logarithmic factors (see [Mannor and Tsitsiklis, 2004] Theorem 1 for the proof).

Lemma 7. *There exists a distribution $D_{x,y}$ such that any algorithm finding the optimal action k^* has a sample complexity of at least:*

$$\Omega\left(\frac{K}{\Delta_2^2} \log \frac{1}{\delta}\right)$$

Analysis of a decision stump

The decision stump uses the values of a contextual variable to choose the actions. The optimal decision stump uses the best variable to choose the best actions. It plays at time t: $k_t^* = \arg\max_k \mu_k^{i^*}|v$, where $i^* = \arg\max_i \mu^i$, and $v = x_{i^*}(t)$. The expected gain of the optimal policy is:

$$\mathbb{E}_{D_{x,y}}\left[y_{k_t^*}(t)\right] = \sum_v P(v)\mu_{k^*}^{i^*}|v = \sum_v \mu_{k^*,v}^{i^*} = \mu^{i^*}$$

To select the best variable, one needs to find the best action of each value of each variable. In DECISION STUMP algorithm (see Algorithm 14), an action selection task is allocated for each value of each contextual variable. When the reward is revealed, all the estimated rewards of variables $\hat{\mu}^i$ and the estimated rewards of the played action knowing the observed values of variables $\hat{\mu}_k^i|v$ are updated (respectively in VE and AE functions): the variables and the actions are simultaneously explored. However, the elimination of actions becomes effective only when the best variable is selected. Indeed, if an action k is eliminated for a value v_0 of a variable i, the estimation of $\hat{\mu}_{k,v_1}^i$, the mean expected reward of the action k for the value v_1, is biased. As a consequence, if an action is eliminated during the exploration of variables, the estimation of the mean reward μ^i can be biased. That is why, the lower bound of decision stump problem is the sum of lower bound of variable and action selection problems (see Theorem 17). The

only case where an action can be eliminated before the best variable be selected, is when this action is eliminated for all values of all variables. For simplicity of the exposition of the DECISION STUMP algorithm we did not handle this case here.

Algorithm 14 DECISION STUMP

$t := 0$, $\forall k\ t_k := 0$, $\forall i\ \hat{\mu}^i := 0$, $\forall (i,v,k)\ t_{i,v,k} := 0$, $\mathcal{A}_{i,v} := \mathcal{A}$, $\hat{\mu}^i_{k,v} := 0$ and $\hat{\mu}^i_k|v := 0$
repeat
  Receive the context vector $\mathbf{x_t}$
  if $|\mathcal{V}| > 1$ **then** Play $k := \text{Round-robin}\,(\mathcal{A})$
  else Play $k := \text{Round-Robin}\,(\mathcal{A}_{i,v})$
  Receive the reward $y_k(t)$
  $t_k := t_k + 1$
  $\mathcal{V} :=$VE$(t_k, k, \mathbf{x_t}, y_k, \mathcal{V}, \mathcal{A})$
  for each variable $i \in \mathcal{V}$ **do**
    $v := x_i(t)$, $t_{i,v,k} := t_{i,v,k} + 1$
    $\mathcal{A}_{i,v} :=$ AE$(t_{i,v,k}, k, \mathbf{x_t}, y_k, \mathcal{A}_{i,v})$
  end for
  $t := t + 1$
until $t = T$

Theorem 16. *When $K \geq 2$, $M \geq 2$, and $\epsilon = 0$, the sample complexity needed by* DECISION STUMP *to obtain $\mathbb{P}\left(i' \neq i^*\right) \leq \delta$ and $\mathbb{P}\left(k'_t \neq k^*_t\right) \leq \delta$ is:*

$$t^* = \frac{64K}{\Delta_1^2} \log \frac{4KM}{\delta \Delta_1} + \frac{64K}{\Delta_2^2} \log \frac{4K}{\delta \Delta_2},$$

$$\textit{where } \Delta_1 = \min_{i \neq i^*} \left(\mu^{i^*} - \mu^i\right),$$
$$\textit{and } \Delta_2 = \min_{k \neq k^*, v \in \{0,1\}} \left(\mu^{i^*}_{k^*,v} - \mu^{i^*}_{k,v}\right).$$

Theorem 17 provides a lower bound for the sample complexity, showing that the factors $1/\Delta_1^2$ and $1/\Delta_2^2$ are inherent of the decision stump problem. Notice that for the linear bandit problem, the same factor $1/\Delta^2$ was obtained in the lower bound (see Lemma 2 in [Soare et al., 2014]).

Theorem 17. *It exists a distribution $D_{x,y}$ such that any algorithm finding the optimal decision stump has a sample complexity of at least:*

$$\Omega\left(\left(\frac{1}{\Delta_1^2} + \frac{1}{\Delta_2^2}\right) K \log \frac{1}{\delta}\right)$$

Remark 15. *The factors in $\log 1/\Delta_1$, $\log 1/\Delta_2$ and $\log 1/\Delta$ could vanish in Theorem 1 following the same approach as that Median Elimination in [Even-Dar et al., 2006]. Despite the optimality of this algorithm, we did not choose it for the consistency of the analysis. Indeed, as it suppresses $1/4$ of variables (or actions) at the end of each elimination phase, Median Elimination is not well suited when a few number of variables (or actions) provide lot of rewards and*

the others not. In this case this algorithm spends a lot of times to eliminate non-relevant variables. This case is precisely the one, where we would like to use a local model such as a decision tree.

Remark 16. *The extension of the analytical results to an ϵ-approximation of the best decision stump is straightforward using $\Delta_\epsilon = \max(\epsilon, \Delta)$.*

4.2.5 BANDIT FOREST

A decision stump is a weak learner which uses only one variable to decide the best action. When one would like to combine D variables to choose the best action, a tree structure of $\binom{M}{D}.2^D$ multi-armed bandit problems has to be allocated. To explore and exploit this tree structure with limited memory resources, our approach consists in combining greedy decision trees. When decision stumps are stacked in a greedy tree, they combine variables in a greedy way to choose the action. When a set of randomized trees vote, they combine variables in a non-greedy way to choose the action.

As highlighted by empirical studies (see for instance [Fernández-Delgado et al., 2014]), *random forests* of [Breiman, 2001] have emerged as a serious competitors to state-of-the-art methods for classification tasks. In [Biau, 2012], the analysis of *random forests* shows that the reason of these good performances comes from the fact that the convergence of its learning procedure is consistent, and its rate of convergence depends only on the number of strong features, which is assumed to be low in comparison to the number of features. In this section, we propose to use a *random forest* built with the knowledge of $D_{x,y}$ as a reference for the proposed algorithm BANDIT FOREST.

Algorithm 15 θ-OGT $\left(c^*_\theta, d_\theta, \mathcal{V}_{c^*_\theta}\right)$

Inputs: $\theta \in [0,1]$
Output: the θ-optimal greedy tree when θ-OGT $((), 0, \mathcal{V})$ is called

if $d_\theta < D_\theta$ **then**
  $\mathcal{S}_{c^*_\theta} := \left\{i \in \mathcal{V}_{c^*_\theta} : f(\theta, c^*_\theta, i) = 1\right\}$
  $i^*_{d_\theta+1} := \arg\max_{i \in \mathcal{S}_{c^*_\theta}} \mu^i | c^*_\theta$
  θ-OGT $\left(c^*_\theta + (x_{i^*_{d_\theta+1}} = 0), d_\theta + 1, \mathcal{V}_{c^*_\theta} \setminus \{i^*_{d_\theta+1}\}\right)$
  θ-OGT $\left(c^*_\theta + (x_{i^*_{d_\theta+1}} = 1), d_\theta + 1, \mathcal{V}_{c^*_\theta} \setminus \{i^*_{d_\theta+1}\}\right)$
  else $k^* | c^*_\theta := \arg\max_k \mu_k | c^*_\theta$
end if

Let Θ be a random variable, which is independent of $\mathbf{x}$ and $\mathbf{y}$. Let $\theta \in [0,1]$ the value of Θ. Let D_θ be the maximal depth of the tree θ. Let $c_\theta = \{(x_{i_1} = v), ..., (x_{i_{d_\theta}} = v)\}$ be the index of a path of the tree θ. We use $c_\theta(\mathbf{x_t})$ to denote the path c_θ selected at time t by the tree θ. Let $f(\theta, i, j) : [0,1] \times \{0,1\}^{2^D} \times M \to \{0,1\}$ be a function which parametrizes $\mathcal{V}_{c_\theta}$ the sets of available variables for each of 2^{D_θ} splits. We call θ-optimal greedy tree, the greedy tree built with the knowledge of $D_{x,y}$ and conditioned to the value θ of the random variable Θ (see Algorithm 15). We call optimal *random forest* of size L, a *random forest*, which consists of a collection of L θ-optimal greedy trees. At each

time step, the optimal *random forest* chooses the action k_t^*, which obtains the higher number of votes:

$$k_t^* = \arg\max_k \sum_{i=1}^{L} \mathbb{1}_{k_{\theta_i,t}^*=k} \ ,$$

where $k_{\theta_i,t}^*$ is the action chosen by the optimal greedy tree θ_i at time t.

Algorithm 16 BANDIT FOREST

1: $t := 0$, $\forall \theta$ $c_\theta := ()$ and $d_\theta := 0$, $\forall \theta$ NewPath$(c_\theta, \mathcal{V})$
2: **repeat**
3: Receive the context vector $\mathbf{x_t}$
4: **for** each θ **do** select the context path $c_\theta(\mathbf{x_t})$
5: **if** $\forall \theta$ $d_\theta = D_\theta$ and $|\mathcal{S}_{c_\theta}| = 1$, and $\forall (\theta, i, v)$ $|\mathcal{A}_{c_{\theta,i,v}}| = 1$ **then** $k := \arg\max_k \sum_{i=1}^{L} \mathbb{1}_{k_{\theta_i,t}=k}$
6: **else** $k =: \text{Round-robin}(\mathcal{A})$
7: **endif**
8: Receive the reward $y_k(t)$
9: **for** each θ **do**
10: $t_{c_\theta,k} := t_{c_\theta,k} + 1$
11: $\mathcal{S}_{c_\theta} :=VE(t_{c_\theta,k}, k, \mathbf{x_t}, y_k, \mathcal{S}_{c_\theta}, \mathcal{A})$
12: **for** each remaining variable i **do**
13: $v =: x_i(t)$, $t_{c_\theta,i,v,k} := t_{c_\theta,i,v,k} + 1$
14: $\mathcal{A}_{c_\theta,i,v} :=$ AE$(t_{c_\theta,i,v,k}, k, \mathbf{x_t}, y_k, \mathcal{A}_{c_\theta,i,v})$
15: **end for**
16: **if** $|\mathcal{S}_{c_\theta}| = 1$ **and** $d_\theta < D_\theta$ **then**
17: $\mathcal{V}_{c_\theta} := \mathcal{V}_{c_\theta} \setminus \{i\}$
18: NewPath$(c_\theta + (x_i = 0), \mathcal{V}_{c_\theta})$
19: NewPath$(c_\theta + (x_i = 1), \mathcal{V}_{c_\theta})$
20: **end if**
21: **end for**
22: $t := t + 1$
23: **until** $t = T$

1: **Function** NewPath$(c_\theta, \mathcal{V}_{c_\theta})$
2: $\mathcal{S}_{c_\theta} := \{i \in \mathcal{V}_{c_\theta} : f(\theta, c_\theta, i) = 1\}$
3: $d_\theta := d_\theta + 1$, $\forall (i, v)$ $\mathcal{A}_{c_\theta,i,v} := \mathcal{A}$
4: $\forall k$ $t_{c_\theta,k} := 0$, $\forall (i, v, k)$ $t_{c_\theta,i,v,k} := 0$, $\forall i$ $\hat{\mu}^i | c_\theta := 0$
5: $\forall (i, k, v)$ $\hat{\mu}_{k,v}^i | c_\theta := 0$ and $\hat{\mu}_k^i | (v, c_\theta) := 0$

BANDIT FOREST algorithm explores and exploits a set of L decision trees knowing $\theta_1, ..., \theta_L$ (see Algorithm 16). When a context $\mathbf{x_t}$ is received (line 3):

- For each tree θ, the path c_θ is selected (line 4).
- An action k is selected:
  - If the learning of all paths c_θ is finished, then each path vote for its best action (line 5).
  - Else the actions are sequentially played (line 6).
- The reward $y_k(t)$ is received (line 8).

- The decision stumps of each path c_θ are updated (lines 9-15).
- When the set of remaining variables of the decision stump corresponding to the path c_θ contains only one variable and the maximum depth D_θ is not reached (line 16), two new decision stumps corresponding to the values 0,1 of the selected variable are allocated (lines 18-20). The random set of remaining variables $\mathcal{V}_{c_\theta}$, the counts, and the estimated means are initialized in function NewPath.

To take into account the L decision trees of maximum depth D_θ, irrelevant variables are eliminated using a slight modification of inequality (4.6). A possible next variable x_i is eliminated when:

$$\hat{\mu}^{i'}|c_\theta - \hat{\mu}^{i}|c_\theta + \epsilon \geq 4\sqrt{\frac{1}{2t_{c_\theta,k}} \log \frac{4 \times 2^D K M D_\theta L t_{c_\theta,k}^2}{\delta}} \,, \tag{4.8}$$

where $i' = \arg\max_{i \in \mathcal{V}_{c_\theta}} \hat{\mu}^i|c_\theta$, and $t_{c_\theta,k}$ is the number of times the path c_θ and the action k have been observed. To take into account the L decision trees, irrelevant actions are eliminated using a slight modification of inequality (4.7):

$$\hat{\mu}_{k'|c_\theta} - \hat{\mu}_{k|c_\theta} + \epsilon \geq 2\sqrt{\frac{1}{2t_{c_\theta,i,v,k}} \log \frac{4 \times 2^D K L t_{c_\theta,i,v,k}^2}{\delta}} \,, \tag{4.9}$$

where $k' = \arg\max_k \hat{\mu}_{k|c_\theta}$, and $t_{c_\theta,i,v,k}$ is the number of times the action k has been played when the path c_θ, and the value v of the variable i have been observed.

Theorem 18. *When $K \geq 2$, $M \geq 2$, and $\epsilon = 0$, the sample complexity needed by* BANDIT FOREST *learning to obtain the optimal random forest of size L with a probability at least $1 - \delta$ is:*

$$t^* = 2^D \left(\frac{64K}{\Delta_1^2} \log \frac{4KMDL}{\delta \Delta_1} + \frac{64K}{\Delta_2^2} \log \frac{4LK}{\delta \Delta_2} \right) \,,$$

where $\Delta_1 = \min_{\theta, c_\theta^*, i \neq i^*} P(c_\theta^*) \left(\mu^{i^*}|c_\theta^* - \mu^i|c_\theta^* \right)$, $\Delta_2 = \min_{\theta, c_\theta^*, k \neq k^*} P(c_\theta^*) \left(\mu_{k^*}|c_\theta^* - \mu_k|c_\theta^* \right)$, and $D = \max_\theta D_\theta$.

The dependence of the sample complexity on the depth D is exponential. This means that like all decision trees, BANDIT FOREST is well suited for cases, where there is a small subset of relevant variables belonging to a large set of variables ($D << M$). This usual restriction of local models is not a problem for a lot of applications, where one can build thousands of contextual variables, and where only a few of them are relevant.

Theorem 19. *There exists a distribution $D_{x,y}$ such that any algorithm finding the optimal random forest of size L has a sample complexity of at least:*

$$\Omega \left(2^D \left[\frac{1}{\Delta_1^2} + \frac{1}{\Delta_2^2} \right] K \log \frac{1}{\delta} \right)$$

Theorem 19 shows that BANDIT FOREST algorithm is near optimal. The result of this analysis is supported by empirical evidence in the next section.

Remark 17. *We have chosen to analyze* BANDIT FOREST *algorithm in the case of* $\epsilon = 0$ *in order to simplify the concept of the optimal policy. Another way is to define the set of* ϵ*-optimal random forests, built with decision stumps which are optimal up to an* ϵ *approximation factor. In this case, the guarantees are given with respect to the worst policy of the set. When* $\epsilon = 0$*, this set contains only the optimal random forest of size* L *given the values of* Θ*.*

4.2.6 Experimentation

In order to illustrate the theoretical analysis with reproducible results on large sets of contextual variables, we used three datasets from the *UCI Machine Learning Repository* (*Forest Cover Type*, *Adult*, and *Census1990*). We recoded each continuous variable using *equal frequencies* into 5 binary variables, and each categorical variable into disjunctive binary variables. We obtained 94, 82 and 255 binary variables, for *Forest Cover Type*, *Adult* and *Census1990* datasets respectively. For *Forest Cover Type*, we used the 7 target classes as the set of actions. For *Adult*, the categorical variable *occupation* is used as a set of 14 actions. For *Census1990*, the categorical variable *Yearsch* is used as a set of 18 actions. The gain of policies was evaluated using the class labels of the dataset with a reward of 1 when the chosen action corresponds to the class label and 0 otherwise. The datasets, respectively composed of 581000, 48840 and 2458285 instances, were shuffled and played in a loop to simulate streams. In order to introduce noise between loops, at each time step the value of each binary variable has a probability of 0.05 to be inverted. Hence, we can consider that each context-reward vector is generated by a stationary random process. We set the time horizon to 10 millions of iterations. The algorithms are assessed in terms of accumulated regret against the optimal *random forest* of size 100. The optimal *random forest* is obtained by training a *random forest* of size 100 not limited by depth on the whole dataset with full information feedback and without noise.

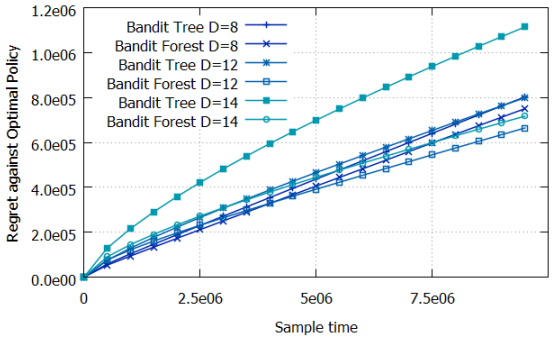


Figure 4.7: The accumulated regret of BANDIT FOREST and BANDIT TREE with different depths against the optimal policy averaged over ten trials on *Forest Cover Type* dataset played in a loop with noisy inputs.

To effectively implement BANDIT FOREST, we have done two modifications of the analyzed algorithms. Firstly, the Round-robin function is replaced by an uniform random draw from the union of the remaining actions of the paths.

Secondly, the rewards are normalized using *Inverse Propensity Scoring* (see [Horvitz and Thompson, 1952]): the obtained reward is divided by the probability to draw the played action. First of all, notice that the regret curves of BANDIT FOREST algorithm are far from those of explore then exploit approaches: the regret is gradually reduced over time (see Figure 4.7). Indeed, BANDIT FOREST algorithm uses a *localized explore then exploit* approach: most of paths, which are unlikely, may remain in exploration state, while the most frequent ones already vote for their best actions. The behavior of the algorithm with regard to number of trees L is simple: the higher L, the greater the performances, and the higher the computational cost (see Figure 4.9). To analyze the sensitivity to depth of BANDIT FOREST algorithm, we set the maximum depth and we compared the performances of a single tree without randomization (BANDIT TREE) and of BANDIT FOREST with $L = 100$. The trees of the forest are randomized at each node with different values of ϵ (between 0.4 and 0.8), and with different sets of available splitting variables (random subset of 80% of remaining variables). For each tested depth, a significant improvement is observed thanks to the vote of randomized trees (see Figure 4.7). Moreover, BANDIT FOREST algorithm appears to be less sensible to depth than a single tree: the higher the difference in depth, the higher the difference in performance.

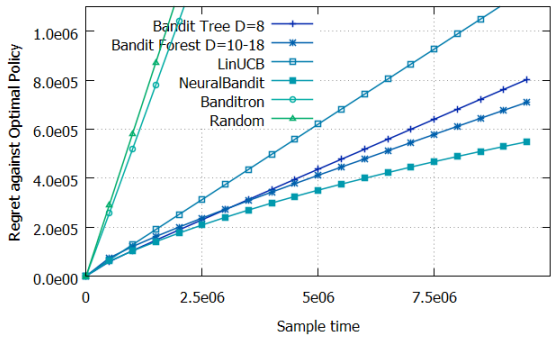


Figure 4.8: The accumulated regret against the optimal policy averaged over ten trials for the dataset *Forest Cover Type* played in a loop with noisy inputs.

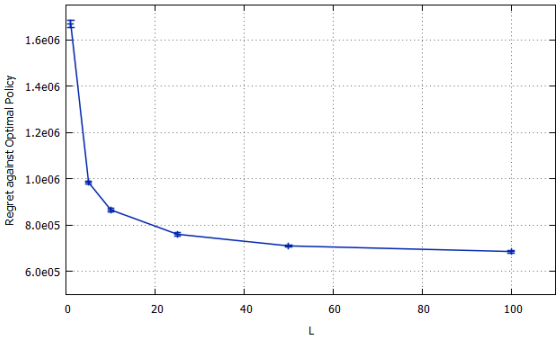


Figure 4.9: Sensitivity to the size of BANDIT FOREST on *Census 1990* dataset.

To compare with state-of-the-art, the trees in the forest are randomized with different values of the parameters D (between 10 and 18). On the three datasets, BANDITRON ([Kakade et al., 2008b]) is clearly outperformed by the other tested algorithms (see Figures 4.8, 4.10, 4.11). NEURAL BANDIT ([Allesiardo et al., 2014]) is a Committee of

Multi-Layer Perceptrons (MLP). Due to the non convexity of the error function, MLP trained with the back-propagation algorithm ([Rumelhart et al., 1986]) does not find the optimal solution. That is why finding the regret bound of such model is still an open problem. However, since MLPs are universal approximators ([Hornik et al., 1989]), NEURAL BANDIT is a very strong baseline. It outperforms LINUCB ([Li et al., 2010a]) on *Forest Cover Type* and on *Adult*. BANDIT FOREST clearly outperforms LINUCB and BANDITRON on the three datasets. In comparison to NEURAL BANDIT, BANDIT FOREST obtains better results on *Census 1990* and *Adult*, and it is outperformed on *Forest Cover Type*, where the number of strong features seems to be high. Finally as shown by the worst case analysis, the risk to use BANDIT FOREST on a lot of optimization problems is controlled, while NEURAL BANDIT, which has no theoretical guaranty, can obtain poor performances on some problems, such as *Census 1990* where it is outperformed by the linear solution LINUCB. This uncontrolled risk increases with the number of actions, since the probability to obtain a non robust MLP linearly increases with the number of actions.

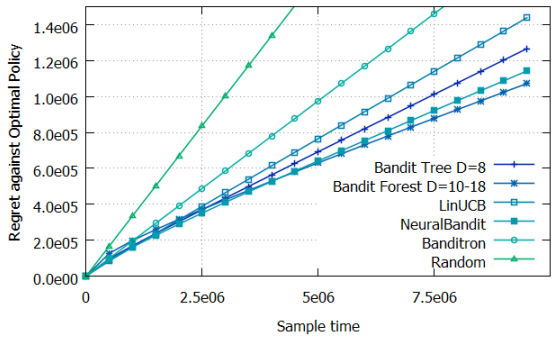


Figure 4.10: The accumulated regret against the optimal policy averaged over ten trials for the dataset *Adult* played in a loop with noisy inputs.

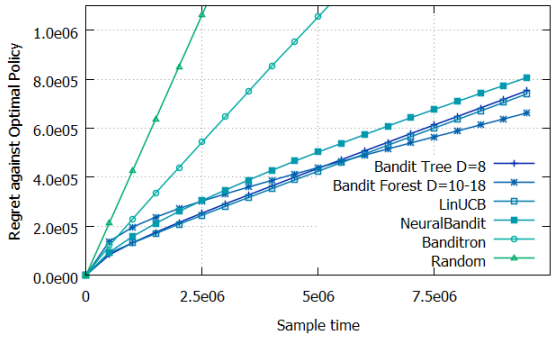


Figure 4.11: The accumulated regret against the optimal policy averaged over ten trials for the dataset *Census1990* played in a loop with noisy inputs.

We provide below (see Table 4.4) the classification rates to compare the asymptotical performances of each algorithm, and the processing times.

Table 4.4: Summary of results on the datasets played in a loop. The regret against the optimal *random forest* is evaluated on ten trials. Each trial corresponds to a random starting point in the dataset. The confidence interval is given with a probability 95%. The classification rate is evaluated on the last 100000 contexts. The mean running time was evaluated on a simple computer with a quad core processor and 6 GB of RAM.

Algorithm	Regret	Classification rate	Running time
Forest Cover Type, action: Cover Type (7 types)			
BANDITRON	$1.99\ 10^6\ \pm 10^5$	49.1%	10 min
LINUCB	$1.23\ 10^6\ \pm 10^3$	60%	360 min
NEURALBANDIT	$0.567\ 10^6\ \pm 2.10^4$	68%	150 min
BANDIT TREE $D = 8$	$0.843\ 10^6\ \pm 2.10^5$	64.2%	5 min
BANDIT FOREST $D10 - 18$	$0.742\ 10^6\ \pm 5.10^4$	65.8%	500 min
Adult, action: occupation (14 types)			
BANDITRON	$1.94\ 10^6\ \pm 3.10^4$	21.1%	20 min
LINUCB	$1.51\ 10^6\ \pm 4.10^4$	25.7%	400 min
NEURALBANDIT	$1.2\ 10^6\ \pm 10^5$	29.6%	140 min
BANDIT TREE $D = 8$	$1.33\ 10^6\ \pm 10^5$	27.9%	4 min
BANDIT FOREST $D10 - 18$	$1.12\ 10^6\ \pm 7.10^4$	31%	400 min
Census1990, action: Yearsch (18 types)			
BANDITRON	$2.07\ 10^6\ \pm 2.10^5$	27%	26 min
LINUCB	$0.77\ 10^6\ \pm 5.10^4$	40.3%	1080 min
NEURALBANDIT	$0.838\ 10^6\ \pm 10^5$	41.7%	300 min
BANDIT TREE $D = 8$	$0.78\ 10^6\ \pm 2.10^5$	41%	10 min
BANDIT FOREST $D10 - 18$	$0.686\ 10^6\ \pm 5.10^4$	43.2%	1000 min

4.2.7 Conclusion

We have shown that the proposed algorithm is optimal up to logarithmic factors with respect to a strong reference: a *random forest* built knowing the joint distribution of the contexts and rewards. In the experiments, BANDIT FOREST clearly outperforms LINUCB, which is a strong baseline with known regret bound, and performs as well as NEURAL BANDIT, for which we do not have theoretical guaranty. Finally, for applications where the number of strong features is low in comparison to the number of possible features, which is often the case, BANDIT FOREST shows valuable properties:

- its sample complexities have a logarithmic dependence on the number of contextual variables, which means that it can process a large amount of contextual variables with a low impact on regret,

- its low computational cost allows to process efficiently infinite data streams,

- like all decision tree algorithms, it is well suited to deal with non linear dependencies between contexts and rewards.

4.2.8 Proofs

Notations

In order to facilitate the reading of the paper, we provide below (see Table 4.5) the list of notations.

Table 4.5: Notations

notation	description
K	number of actions
M	number of contextual variables
D_θ	maximum depth of the tree θ
L	number of trees
T	time horizon
$\mathcal{A}$	set of actions
$\mathcal{V}$	set of variables
$\mathcal{S}$	set of remaining variables
$\mathbf{x}$	context vector $\mathbf{x} = (x_1, \ldots, x_M)$
$\mathbf{y}$	reward vector $\mathbf{y} = (y_1, \ldots, y_K)$
k_t	action chosen at time t
c_θ	context path the tree θ, $c_\theta = (x_{i_1}, v_{i_1}), ..., (x_{i_{d_\theta}}, v_{i_{d_\theta}})$
d_θ	current depth of the context path c_θ
μ_k	expected reward of action k, $\mu_k = \mathbb{E}_{D_y}[y_k]$
$\mu_k^i \vert v$	expected reward of action k conditioned to $x_i = v$, $\mu_k^i \vert v = \mathbb{E}_{D_y}[y_k \cdot \mathbb{1}_{x_i = v}]$
$\mu_{k,v}^i$	expected reward of action k and $x_i = v$, $\mu_{k,v}^i = \mathbb{E}_{D_{x,y}}[y_k \cdot \mathbb{1}_{x_i = v}]$
μ^i	expected reward for the use of the variable x_i to select the best actions
δ	probability of error
ϵ	approximation error
Δ_1	minimum of difference with the expected reward of the best action μ_{k^*} and the expected reward of a given action k: $\Delta_1 = \min_{k \neq k^*}(\mu_{k^*} - \mu_k)$
Δ_2	minimum of difference with the best variable expected reward μ^{i^*} and the expected reward for other variables: $\Delta_2 = \min_{i \neq i^*}(\mu^{i^*} - \mu^i)$
t^*	sample complexity of the decision stump

Lemma 4

Proof. We cannot use directly Hoeffding inequality (see [Hoeffding, 1994]) to bound the estimated gains of the use of variables. The proof of Lemma 4 overcomes this difficulty by bounding each estimated gain μ^i by the sum of the bounds over the values of the expected reward of the best action $\mu_{k^*,v}^i$ (inequality 4.12). From Hoeffding's inequality, at time t we have:

$$\mathbb{P}\left(\left|\hat{\mu}_{k,v}^i - \mu_{k,v}^i\right| \geq \alpha_{t_k}\right) \leq 2\exp(-2\alpha_{t_k}^2 t_k) = \frac{\delta}{4KMt_k^2} ,$$

where $\alpha_{t_k} = \sqrt{\frac{1}{2t_k}\log\frac{4KMt_k^2}{\delta}}$.

Using Hoeffding's inequality on each time t_k, applying the union bound and then $\sum 1/t_k^2 = \pi^2/6$, the following inequality holds for any time t with a probability $1 - \frac{\delta\pi^2}{12KM}$:

$$\hat{\mu}^i_{k,v} - \alpha_{t_k} \leq \mu^i_{k,v} \leq \hat{\mu}^i_{k,v} + \alpha_{t_k} \tag{4.10}$$

If the inequality (4.10) holds for the actions $k' = \arg\max_k \hat{\mu}^i_{k,v}$, and $k^* = \arg\max_k \mu^i_{k,v}$, we have:

$$\begin{aligned} &\hat{\mu}^i_{k',v} - \alpha_{t_k} \leq \mu^i_{k',v} \leq \mu^i_{k^*,v} \leq \hat{\mu}^i_{k^*,v} + \alpha_{t_k} \leq \hat{\mu}^i_{k',v} + \alpha_{t_k} \\ &\Rightarrow \hat{\mu}^i_{k',v} - \alpha_{t_k} \leq \mu^i_{k^*,v} \leq \hat{\mu}^i_{k',v} + \alpha_{t_k} \end{aligned} \tag{4.11}$$

If the previous inequality (4.11) holds for all values v of the variable x_i, we have:

$$\begin{aligned} &\sum_{v\in\{0,1\}} \left(\hat{\mu}^i_{k',v} - \alpha_{t_k}\right) \leq \sum_{v\in\{0,1\}} \mu^i_{k^*,v} \leq \sum_{v\in\{0,1\}} \left(\hat{\mu}^i_{k',v} + \alpha_{t_k}\right) \\ &\Leftrightarrow \hat{\mu}^i - 2\alpha_{t_k} \leq \mu^i \leq \hat{\mu}^i + 2\alpha_{t_k} \end{aligned} \tag{4.12}$$

If the previous inequality holds for $i' = \arg\max_i \hat{\mu}^i$, we have:

$$\hat{\mu}^{i'} - 2\alpha_{t_k} \leq \mu^{i'} \leq \mu^{i^*} \tag{4.13}$$

As a consequence, the variable x_i cannot be the best one when:

$$\hat{\mu}^i + 2\alpha_{t_k} \leq \hat{\mu}^{i'} - 2\alpha_{t_k}\ , \tag{4.14}$$

Using the union bound, the probability of making an error about the selection on the next variable by eliminating each variable x_i when the inequality (4.14) holds is bounded by the sum for each variable x_i and each value v that the inequality 4.11 does not hold for k' and k^*:

$$\mathbb{P}\left(i^* \neq i'\right) \leq \sum_{i\in\mathcal{V}} \frac{K\delta\pi^2}{24KM} \leq \sum_{i\in\mathcal{V}} \frac{\delta}{M} \leq \delta \tag{4.15}$$

Now, we have to consider t^*_k, the number of steps needed to select the optimal variable. If the best variable has not been eliminated (probability $1 - \delta$), the last variable x_i is eliminated when:

$$\hat{\mu}^{i^*} - \hat{\mu}^i \geq 4\alpha_{t^*_k}$$

The difference between the expected reward of a variable x_i and the best next variable is defined by:

$$\Delta_i = \mu^{i^*} - \mu^i$$

Assume that:

$$\Delta_i \geq 4\alpha_{t_k} \tag{4.16}$$

The following inequality holds for the variable x_i with a probability $1 - \frac{\delta}{KM}$:

$$\hat{\mu}^i - 2\alpha_{t_k} \leq \mu^i \leq \hat{\mu}^i + 2\alpha_{t_k}$$

Then, using the previous inequality in the inequality (4.16), we obtain:

$$(\hat{\mu}^{i^*} + 2\alpha_{t_k}) - (\hat{\mu}^i + 2\alpha_{t_k}) \geq \mu^{i^*} - \mu^i \geq 4\alpha_{t_k}$$

Hence, we have:

$$\hat{\mu}^{i^*} - \hat{\mu}^i \geq 4\alpha_{t_k}$$

The condition $\Delta_i \geq 4\alpha_{t_k}$ implies the elimination of the variable x_i. Then, we have:

$$\Delta_i \geq 4\alpha_{t_k}$$

$$\Rightarrow \Delta_i^2 \geq \frac{8}{t_k} \log \frac{4KMt_k^2}{\delta} \tag{4.17}$$

The time t_k^*, where all non optimal variables have been eliminated, is reached when the variable corresponding to the minimum of Δ_i^2 is eliminated.

$$\Rightarrow \Delta^2 \geq \frac{8}{t_k^*} \log \frac{4KMt_k^{*2}}{\delta} \ , \tag{4.18}$$

where $\Delta = \min_{i \neq i^*} \Delta_i$.

The inequality (4.18) holds for all variables i with a probability $1 - \delta$ for:

$$t_k^* = \frac{64}{\Delta^2} \log \frac{4KM}{\delta.\Delta} \tag{4.19}$$

Indeed, if we replace the value of t_k^* in the right term of the inequality (4.17), we obtain:

$$\frac{\Delta^2}{8\log\frac{4KM}{\delta.\Delta}}\left(\log\frac{4KM}{\delta}+2\log\frac{64}{\Delta^2}+2\log\log\frac{4KM}{\delta.\Delta}\right)=$$
$$\frac{\Delta^2}{8\log\frac{4KM}{\delta.\Delta}}\left(\log\frac{4KM}{\delta}-4\log\Delta+12\log 2+2\log\log\frac{4KM}{\delta.\Delta}\right)\leq$$
$$\frac{\Delta^2}{8\log\frac{4KM}{\delta.\Delta}}\left(4\log\frac{4KM}{\delta.\Delta}+12\log 2+2\log\log\frac{4KM}{\delta.\Delta}\right)$$

For $x \geq 13$, we have:

$$12\log 2+2\log\log x<4\log x$$

Hence, for $4KM \geq 13$, we have:

$$\frac{\Delta^2}{8\log\frac{4KM}{\delta.\Delta}}\left(4\log\frac{4KM}{\delta.\Delta}+12\log 2+2\log\log\frac{4KM}{\delta.\Delta}\right)\leq$$
$$\frac{\Delta^2}{8\log\frac{4KM}{\delta.\Delta}}8\log\frac{4KM}{\delta.\Delta}=\Delta^2$$

Hence, we obtain:

$$t_k^*=\frac{64}{\Delta^2}\log\frac{4KM}{\delta}\text{ , with a probability } 1-\delta$$

As the actions are chosen the same number of times (Round-robin function), we have $t = Kt_k$, and thus:

$$t^*=\frac{64K}{\Delta^2}\log\frac{4KM}{\delta}\text{ , with a probability } 1-\delta$$

□

Lemma 5

Proof. Let $y^i_{k,v}$ be a bounded random variable corresponding to the reward of the action k when the value v of the variable i is observed. Let y^i be a random variable such that:

$$y^i=\max_k y^i_{k,v}$$

We have:

$$\mathbb{E}_{D_{x,y}}[y^i]=\mu^i$$

Each y^i is updated each step t_k when each action has been played once. Let Θ be the sum of the binary random variables $\theta_1,...,\theta_{t_k},...,\theta_{t_k^*}$ such that $\theta_{t_k}=\mathbb{1}_{y^i(t_k)\geq y^j(t_k)}$. Let p_{ij} be the probability that the use of variable i leads to more rewards than the use of variable j. We have:

$$p_{ij} = \frac{1}{2} - \Delta_{ij} \text{ , where } \Delta_{ij} = \mu^i - \mu^j.$$

Slud's inequality (see [Slud, 1977]) states that when $p \leq 1/2$ and $t_k^* \leq x \leq t_k^*.(1-p)$, we have:

$$P(\Theta \geq x) \geq P\left(Z \geq \frac{x - t_k^*.p}{\sqrt{t_k^*.p(1-p)}}\right) , \tag{4.20}$$

where Z is a normal $\mathcal{N}(0,1)$ random variable.

To choose the best variable between i and j, one needs to find the time t_k^* where $P(\Theta \geq t_k^*/2) \geq \delta$. To state the number of trials t_k^* needed to estimate Δ_{ij}, we recall and adapt the arguments developed in [Mousavi, 2010]. Using Slud's inequality (see equation 4.20), we have:

$$P(\Theta \geq t_k^*/2) \geq P\left(Z \geq \frac{t_k^*.\Delta_{ij}}{\sqrt{t_k^*.p_{ij}(1-p_{ij})}}\right) , \tag{4.21}$$

Then, we use the lower bound of the error function (see [Chu, 1955]):

$$P(Z \geq z) \geq 1 - \sqrt{1 - \exp\left(-\frac{z^2}{2}\right)}$$

Therefore, we have:

$$\begin{aligned} P(\Theta \geq t_k^*/2) &\geq 1 - \sqrt{1 - \exp\left(-\frac{t_k^*.\Delta_{ij}^2}{2p_{ij}(1-p_{ij})}\right)} \\ &\geq 1 - \sqrt{1 - \exp\left(-\frac{t_k^*.\Delta_{ij}^2}{p_{ij}}\right)} \\ &\geq \frac{1}{2}\exp\left(-\frac{t_k^*.\Delta_{ij}^2}{p_{ij}}\right) \end{aligned}$$

As $p_{ij} = 1/2 - \Delta_{ij}$, we have:

$$\log\delta = \log\frac{1}{2} - \frac{t_k^*.\Delta_{ij}^2}{1/2 - \Delta_{ij}} \geq \log\frac{1}{2} - 2t_k^*.\Delta_{ij}^2$$

Hence, we have:

$$t_k^* = \Omega\left(\frac{1}{\Delta_{ij}^2}\log\frac{1}{\delta}\right)$$

Then, we need to use the fact that as all the values of all the variables are observed when one action is played: the $M(M-1)/2$ estimations of bias are solved in parallel. In worst case, $\min_{ij}\Delta_{ij} = \min_j \Delta_{i^*j} = \Delta$. Thus any

algorithm needs at least a sample complexity t^*, where:

$$t^* = K.t_k^* = \Omega\left(\frac{K}{\Delta^2}\log\frac{1}{\delta}\right)$$

□

Theorem 16

Proof. Lemma 4 states that the sample complexity needed to find the best variable is:

$$t_1^* = \frac{64K}{\Delta_1^2}\log\frac{4KM}{\delta\Delta_1}, \text{ where } \quad \Delta_1 = \min_{i\neq i^*}(\mu^{i^*} - \mu^i)$$

Lemma 6 states that the sample complexity needed to find the optimal action for a value v of the best variable is:

$$t_{2,v}^* = \frac{64K}{\Delta_{2,v}^2}\log\frac{4K}{\delta\Delta_{2,v}}, \text{ where } \quad \Delta_{2,v} = \min_{k\neq k^*}(\mu_{k^*,v}^{i^*} - \mu_{k,v}^{i^*}).$$

The sample complexity of decision stump algorithm is bounded by the sum of the sample complexities of variable selection and action elimination algorithms:

$$t^* = t_1^* + t_2^* \text{ , where } t_2^* = \max_v t_{2,v}^*.$$

□

Theorem 17

Proof. In worst case, all the values of variables have different best actions, and $K = 2M$. If an action is suppressed before the best variable is selected, the estimation of the mean reward of one variable is underestimated. In worst case this variable is the best one, and a sub-optimal variable is selected. Thus, the best variable has to be selected before an action be eliminated. The lower bound of the decision stump problem is the sum of variable selection and best arm identification lower bounds, stated respectively in Lemma 5 and Lemma 7.

□

Theorem 18

Proof. The proof of Theorem 18 uses Lemma 4 and Lemma 6. Using the slight modification of the variable elimination inequality proposed in section 3, Lemma 4 states that for each decision stump, we have:

$$\mathbb{P}\left(i_{d_\theta}^*|c_\theta \neq i'_{d_\theta}|c_\theta\right) \leq \frac{\delta}{2^{D_\theta}L}$$

For the action corresponding to the path c_θ, Lemma 6 states that:

$$\mathbb{P}(k \neq k^*) \leq \frac{\delta}{2^{D_\theta} L}$$

From the union bound, we have:

$$\mathbb{P}(\exists c_\theta \text{such that} c_\theta \neq c_\theta^*) \leq \delta \text{ and } \mathbb{P}(\exists k \text{such that} k \neq k^*) \leq \delta$$

Using Lemma 4 and Lemma 6, and summing the sample complexity of each 2^{D_θ} variable selection tasks and the sample complexity of each 2^{D_θ} action selection tasks, we bound the sample complexity of any tree θ by:

$$t^* \leq 2^D \frac{64K}{\Delta_1^2} \log \frac{4KMDL}{\delta \Delta_1} + 2^D \frac{64K}{\Delta_2^2} \log \frac{4KL}{\delta \Delta_2} \ ,$$

where $D = \max D_\theta$.

□

Theorem 19

Proof. To build a decision tree of depth D_θ, any greedy algorithm needs to solve $\sum_{d<D_\theta} 2^d = 2^{D_\theta}$ variable selection problems (one per node), and 2^{D_θ} action selection problems (one per leaf). Then, using Lemma 5 and Lemma 7, any greedy algorithm needs a sample complexity of at least:

$$t^* \geq \Omega \left(2^D \left[\frac{1}{\Delta_1^2} + \frac{1}{\Delta_2^2} \right] K \log \frac{1}{\delta} \right)$$

□

4.3 Context Attentive Bandits for improving dialog

Abstract. In various recommender system applications, from medical diagnosis to dialog systems, due to observation costs only a small subset of a potentially large number of context variables can be observed at each iteration; however, the agent has a freedom to choose which variables to observe. In this paper, we analyze and extend an online learning framework known as *Context-Attentive Bandit*, We derive a novel algorithm, called *Context-Attentive Thompson Sampling (CATS)*, which builds upon the Linear Thompson Sampling approach, adapting it to *Context-Attentive Bandit* setting. We provide a theoretical regret analysis and an extensive empirical evaluation demonstrating advantages of the proposed approach over several baseline methods on a variety of real-life datasets.

4.3.1 Targeted use case: optimizing dialog in after sale service

The after sale service is in charge to support customers when they encounter a technical problem with a product or service. For handling a flow of repetitive problems, bots are used to identify each problem and propose a solution. Currently, the bots use a tree script (figure 4.12). The drawback of this approach is that the tree structure, designed to minimize the number of interactions, imposes to ask questions that seem not relevant for the customer. For instance, if a significant cluster of people calls the after sale service, while they simply forgot to switch on their livebox, for all people the first question will be "is the switch on ?", which is annoying for a customer that has a non-trivial problem.

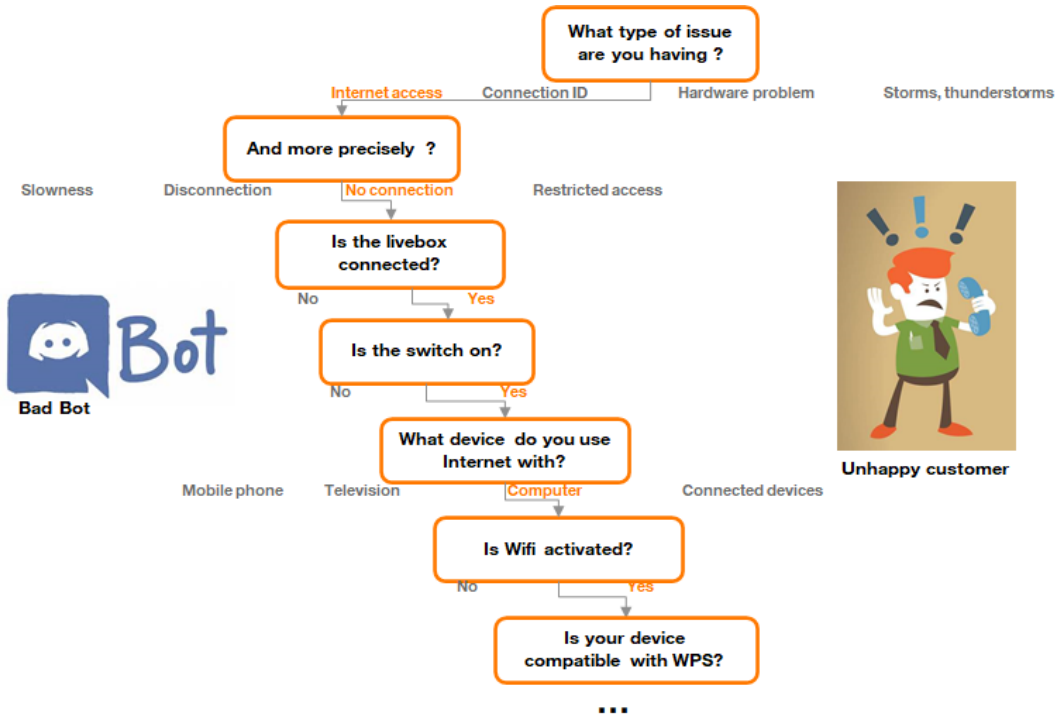


Figure 4.12: Bad Bot for after sale services.

For improving the customer experience, one can use all the elements that we dispose:

- the customer comes to the hot-line with a query in natural language, which can be transformed into a vector using word embedding techniques [Tomas et al., 2013],
- the customer has a profile (products and services, billing, interactions, usages...), which is a vector,
- the after sale service has a closed list of questions that can be useful for identifying the encountered problem (for instance those that are asked in the tree script),
- the after sale service has a closed list of solutions corresponding to identified problems.

In this study, we propose to decompose the bot in two tasks: first the bot asks few additional questions for precising the query of the customer, then the bot identifies the encountered problem and proposes a solution. We formalize this two tasks as two nested contextual bandit problem. The goal of the first contextual bandit is to find the right questions to ask knowing the customer query and profile, while the goal of the second contextual bandit is to identify the encountered problem knowing the customer query and profile and the answers of the customer. The reward is computed using an additional question at the end of the dialog (figure 4.13).

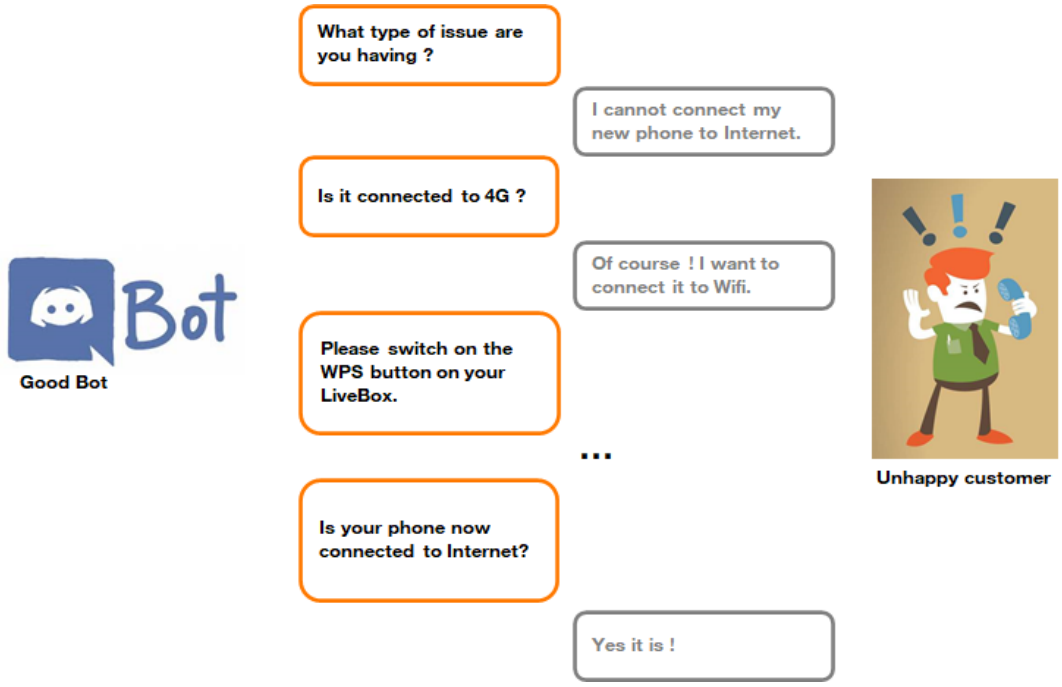


Figure 4.13: Good bot for after sale services.

4.3.2 Introduction

The contextual bandit (CB) problem has been extensively studied in the past, and a variety of solutions have been proposed. In LINUCB [Li et al., 2010a, Abbasi-yadkori et al., 2011, Bouneffouf et al., 2019, Bouneffouf and Rish, 2019], Neural Bandit [Allesiardo et al., 2014] and in linear Thompson Sampling [Agrawal and Goyal, 2013, Balakrishnan et al., 2019a, Balakrishnan et al., 2019b], a linear dependency is assumed between the expected reward given the context and an action taken after observing this context; the representation space is modeled using a set of linear predictors.

Recently, a promising variant of contextual bandits, called a *Context Attentive Bandit (CAB)* was proposed in [Bouneffouf et al., 2017], where no context is given by default, but the agent can request to observe (to focus its "attention" on) a limited number of context variables at each iteration. We propose here an extension of this problem setting: a small subset of V context variables is revealed at each iteration (i.e. partially observable context), followed by the agent's choice of additional U features, where V, U are fixed at all iterations. The agent must learn to select

both the best additional features and, subsequently, the best arm to play, given the resulting $V+U$ observed features. (The original *Context Attentive Bandit* corresponds to $V = 0$.) The proposed setting is motivated by several real-life applications where observing the full context is impossible and the agent is allowed to observe just a small subset of information at the onset. For instance, in a clinical setting, a doctor may first take a look at patient's medical record (partially observed context) to decide which medical test (additional context variables) to perform, before choosing a treatment plan (selecting an arm to play). It is often too costly or even impossible to conduct all possible tests (i.e., observe the full context); therefore, given the limit on the number of tests, the doctor must decide which subset of tests will result into maximally effective treatment choice (maximize the reward).

Similar problems can arise in multi-skill orchestration for AI agents. For example, in dialog orchestration, a user's query is first directed to a number of domain-specific agents, each providing a different response, and then the best response is selected. However, it might be too costly to request the answers from all domain-specific experts, especially in multi-purpose dialog systems with a very large number of domains experts. Given a limit on the number of experts to use for each query, the orchestration agent must choose the best subset of experts to use. In this application, the query is the immediately observed part of the overall context, while the responses of domain-specific experts are the initially unobserved features from which a limited subset must be selected and observed, before choosing an arm, i.e. deciding on the best out of the available responses. For multi-purpose dialog systems, such as, for example, personal home assistants, retrieving features or responses from every domain-specific agent is computationally expensive or intractable, with the potential to cause a poor user experience, again underscoring the need for effective feature selection.

Overall, the main contributions of this paper include:

(1) a generalization of *Context Attentive Bandit*, and a first lower bound for this problem.

(2) an algorithm called *Context Attentive Thompson Sampling* for stationary and non-stationary environments, and its regret bound in the case of stationary environment.

(3) an extensive empirical evaluation demonstrating advantages of our proposed algorithm over the previous context-attentive bandit approach [Bouneffouf et al., 2017], on a range of datasets in both stationary and non-stationary settings.

This paper is organized as follows. In section 4.3.3 we review related works. The section 3 introduces some background concepts. Then we introduce *Context-Attentive Bandit*, the proposed algorithms for both stationary and non-stationary environments, and we provide a lower bound of the proposed problem and a regret bound of the proposed algorithm in the case of stationary environment. Experimental evaluation on several datasets, for varying parameter settings, is presented afterward. Finally, the last section concludes the paper and points out possible directions for future works.

4.3.3 Related Work

In the *contextual bandit problem* [Langford and Zhang, 2007, Agarwal et al., 2009, Li et al., 2010a], the objective of is to learn the relationship between the context and reward, in order to find the best arm-selection policy for maximizing cumulative reward over time. Motivated by dimensionality reduction tasks, [Abbasi-Yadkori et al., 2012] studied a sparse variant of stochastic linear bandits, where only a relatively small and unknown subset of features is relevant to a multivariate function optimization. It presents an application to the problem of optimizing a function that depends on many features, where only a small, initially unknown subset of features is relevant. Similarly, [Carpentier and Munos, 2012] also considered high-dimensional stochastic linear bandits with sparsity. There the authors combined ideas from compressed sensing and bandit theory to derive a novel algorithm. In [Oswal et al., 2020], authors explores a new form of the linear bandit problem in which the algorithm receives the usual stochastic rewards as well as stochastic feedback about which features are relevant to the rewards and propose an algorithm that can achieve $O(\sqrt{T})$ regret, without prior knowledge of which features are relevant.

In [Bouneffouf et al., 2017] the authors proposed the novel framework of *contextual bandit with restricted context*, where observing the whole feature vector at each iteration is too costly or impossible for some reasons; however, the agent can request to observe the values of an arbitrary subset of features within a given budget, i.e. the limit on the number of features observed. This paper explores a more general problem, and unlike [Bouneffouf et al., 2017], we provide a theoretical analysis of the proposed problem and the proposed algorithm.

The *Context Attentive Bandit* problem is related to the *budgeted learning* problem, where a learner can access only a limited number of attributes from the training set or from the test set (see for instance [Cesa-Bianchi et al., 2011]). In [Foster et al., 2016], the authors studied the *online budgeted learning* problem. They showed a significant negative result: for any $\delta > 0$ no algorithm can achieve regret bounded by $O(T^{1-\delta})$ in polynomial time. For overcoming this negative result, an additional assumption is necessary. Here, following [Durand and Gagné, 2014], we assume that the expected reward of selecting a subset of features is the sum of the expected rewards of selecting individually the features. We obtain an efficient algorithm, which has a linear algorithmic complexity in terms of time horizon.

4.3.4 Context Attentive Bandit Problem

We now introduce the problem setting, outlined in Algorithm 5.2.3. Let C be a set of N features. At each time point t the environment generates a feature vector $\boldsymbol{c}(t) = (c_1(t), ..., c_N(t), 1) \in \mathbb{R}^{N+1}$, which the agent cannot observe fully, but a partial observation of the context is allowed: the values of a subset of $V < N$ *observed* features $C^V \subset C$, are revealed: $\boldsymbol{c}^V = (c_1^V(t), ..., c_N^V(t), 1) \in \mathbb{R}^{N+1}$, $\forall i \in C, c_i^V(t) = c_i(t)\mathbb{1}\{i \in C^V\}$. Based on this partially observed context, the agent is allowed to request an additional subset of U *unobserved* features $C^U \subset C \setminus C^V$, $V + U \leq N$. The goal of the agent is to maximize its total reward over time via (1) the optimal choice of the additional set of

features C^U, given the initial observed features $\boldsymbol{c}^V(t)$, and (2) the optimal choice of an arm $k \in [K] = \{1, ..., K\}$ based on $\boldsymbol{c}^{V+U}(t) = (c_1^{V+U}(t), ..., c_N^{V+U}(t), 1) \in \mathbb{R}^{N+1}$, $\forall i \in C, c_i^{V+U}(t) = c_i(t)\mathbb{1}\{i \in C^V \vee i \in C^U\}$. We assume $P_r(r|\boldsymbol{c}, k)$, an unknown probability distribution of the reward given the context and the action taken in that context. In the following the expectations are taken over the probability distribution $P_r(r|c, k)$.

Algorithm 17 Context Attentive Bandit Problem (CAB)

for $t := 1$ to T **do**
  Context $\boldsymbol{c}(t)$ is chosen by the environment
  The values $\boldsymbol{c}^V(t)$ of a subset $C^V \subset C$ are revealed
  The agent selects a subset $C^U \subseteq C \setminus C^V$
  The values $\boldsymbol{c}^U(t)$ are revealed;
  The agent chooses an arm $k := \pi(\boldsymbol{c}^{V+U}(t))$
  The reward $r_k(t)$ is sampled from distribution $P_r(r|\boldsymbol{c}, k)$ and it is revealed
  The agent updates the policy $\pi \in \Pi_{C^{V+U}}$

end for

The contextual bandit problem. Following [Langford and Zhang, 2007], this problem is defined as follows. At each time point $t \in \{1, ..., T\}$, an agent is presented with a *context* (i.e. *feature vector*) $\boldsymbol{c}(t) \in \mathbb{R}^{N+1}$ before choosing an arm $k \in [K]$. Let $\boldsymbol{r}(t) = (r_1(t), ..., r_K(t)) \in \mathbb{R}^K$ denote a reward vector, where $r_k(t)$ is the reward at time t associated with the arm k. Let $\pi : \mathbb{R}^{N+1} \to [K]$ denote a policy, mapping a context $\boldsymbol{c}(t) \in \mathbb{R}^{N+1}$ into an arm $k \in [K]$. We assume that the expected reward is a linear function of the context.

Assumption 3 (linear contextual bandit)**.** *Whatever the subset of selected features $C^U \subset C \setminus C^V$ the expected reward is a linear function of the context: $\mathbb{E}[r_k|\boldsymbol{c}^{V+U}(t)] = \boldsymbol{c}^{V+U}(t)^\top \boldsymbol{\mu}_k$, where $\boldsymbol{\mu}_k \in \mathbb{R}^{N+1}$ is an unknown parameter, $\forall k \in [K]$ $||\boldsymbol{\mu}_k|| \leq 1$, and $||\boldsymbol{c}^{V+U}(t)|| \leq 1$.*

Contextual Combinatorial Bandit. The contextual combinatorial bandit problem [Qin et al., 2014] can be viewed as a game where the agent sequentially observes a context $\boldsymbol{c}^V(t)$, selects a subset $C^U(t) \subset C \setminus \{C^V\}$ and observes the reward corresponding to the selected subset. The goal is to maximize the reward over time. Let $r_U|\mathbf{c}^V(t), \pi \in \mathbb{R}$ be the reward associated with the set of selected features C^U knowing the context vector $\boldsymbol{c}^{U+V}(t)$ and the policy π. We have $r_U|\boldsymbol{c}^V(t), \pi = r_{k(t)}|\boldsymbol{c}^{V+U}(t)$, where $k(t) = \pi(\boldsymbol{c}^{V+U}(t))$. Each feature $i \in C^U$ is associated with the corresponding random variable $r_i|\boldsymbol{c}^V(t), \pi \in \mathbb{R}$ which indicates the reward obtained when choosing the i-th feature at time t.

Assumption 4 (linear contextual combinatorial bandit)**.** *the mean reward of selecting the set of features $C^U \subset C \setminus \{C^V\}$ is: $\mathbb{E}[r_U|\boldsymbol{c}^V(t), \pi] = \sum_{i \in C^U} \mathbb{E}[r_i|\boldsymbol{c}^V(t), \pi]$, and the expectation of the reward of selecting the feature i is a linear function of the context vector $\mathbf{c}^V(t)$: $\mathbb{E}[r_i|\boldsymbol{c}^V(t), \pi] = \boldsymbol{c}^V(t)^\top \boldsymbol{\theta}_i$, where $\boldsymbol{\theta}_i \in \mathbb{R}^{N+1}$ is an unknown weight vector associated with the feature i, $\forall i \in C \setminus C^V$ $||\boldsymbol{\theta}_i|| \leq 1$, and $||\boldsymbol{c}^V(t)|| \leq 1$.*

Let $\Pi_{C^{V+U}}$ be the set of linear policies such that only the features coming from C^{V+U} are used, where C^V is a fixed subset of C, and C^U is any subset of $C \setminus C^V$. The objective of *Contextual Attentive Bandit* (Algorithm 17) is to

find an optimal policy $\pi^* \in \Pi_{C^{V+U}}$, over T iterations or time points, so that the total reward is maximized.

Definition 22 (Optimal Policy for CAB)**.** *The optimal policy π^* for handling the CAB problem is selecting the arm at time t:*

$$k^*(t) = \arg \max_{k \in [K], C^U \in C \setminus C^V} \boldsymbol{c}^{U+V}(t)^\top \boldsymbol{\mu}_k = \arg \max_{k \in [K]} \boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_k,$$

where

$$C^{U^*} = \arg \max_{C^U \subset C \setminus C^V} r_U | \boldsymbol{c}^V(t), \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K = \arg \max_{C^U \subset C \setminus C^V} \sum_{i \in C^U} \boldsymbol{c}^V(t)^\top \boldsymbol{\theta}_i$$

.

Definition 23 (Cumulative regret)**.** *The cumulative regret over T iterations of the policy $\pi \in \Pi_{C^{V+U}}$, is defined as* $R(T) = \sum_{t=1}^T \mathbb{E}[r_{\pi^*(\boldsymbol{c}(t))}] - \sum_{t=1}^T \mathbb{E}[r_{\pi(\boldsymbol{c}(t))}]$.

Proposition 2 (Regret decomposition)**.** *The cumulative regret over T iterations of the policy $\pi \in \Pi_{C^{V+U}}$ can be rewritten as following:*

$$\begin{aligned}
R(T) =& \sum_{t=1}^T \mathbb{E}[r_{\pi^*(\boldsymbol{c}(t))}] - \sum_{t=1}^T \mathbb{E}[r_{\pi(\boldsymbol{c}(t))}] \\
=& \sum_{t=1}^T \left[\boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k^*(t)} - \boldsymbol{c}^{V+U}(t)^\top \boldsymbol{\mu}_{k(t)} \right] \\
=& \sum_{t=1}^T \left[\boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k^*(t)} - \boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k(t)} \right] + \sum_{t=1}^T \left[\boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k(t)} - \boldsymbol{c}^{V+U}(t)^\top \boldsymbol{\mu}_{k(t)} \right] \\
=& \sum_{t=1}^T \left[\boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k^*(t)} - \boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k(t)} \right] + \sum_{t=1}^T \left[\mathbb{E}[r_{U^*} | \boldsymbol{c}^V(t), \pi] - \mathbb{E}[r_{U(t)} | \boldsymbol{c}^V(t), \pi] \right] \\
=& \sum_{t=1}^T \left[\boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k^*(t)} - \boldsymbol{c}^{V+U^*}(t)^\top \boldsymbol{\mu}_{k(t)} \right] + \sum_{t=1}^T \left[\sum_{i \in C^{U^*}} \boldsymbol{c}^V(t)^\top \boldsymbol{\theta}_i - \sum_{i \in C^{U(t)}} \boldsymbol{c}^V(t)^\top \boldsymbol{\theta}_i \right].
\end{aligned}$$

Remark 18. *CAB problem generalizes the contextual bandit with restricted context problem ([Bouneffouf et al., 2017]). Indeed, when the subset of observed context C^V is empty, the reward of selecting the feature i is given by $\theta_{i,N+1}$, which is the coordinate $N+1$ of the vector $\boldsymbol{\theta}_i$.*

Before introducing an algorithm for handling the above CAB problem, we will derive a lower bound on the expected regret of any algorithm used to solve this problem.

Theorem 20. *For any policy $\pi \in \Pi_{C^{V+U}}$ handling Context Attentive Bandit problem (Algorithm 1) under Assumption 3 and 4, there exists probability distribution $P_r(r|\boldsymbol{c}, k)$, such that the lower bound of the regret accumulated by π*

over T iterations is :

$$\Omega\left(\sqrt{(U+V)T} + U\sqrt{VT}\right).$$

Proof. The left term of the regret (see Proposition 2) is lower bounded by the lower bound of linear bandits in dimension $U+V$ (Theorem 2 in [Chu et al., 2011]), while the right term is lower bounded by the lower bound of U linear bandits in dimension V. □

4.3.5 Context Attentive Thompson Sampling (CATS)

Algorithm 18 Context Attentive Thompson Sampling (CATS)

Require: N, V, U, K, T, C^V, $\alpha > 0$, $\lambda(t)$
Initialize: $\forall k \in [K], A_k := I_{N+1}$, $\boldsymbol{g}_k := \mathbf{0}_{N+1}$, $\hat{\boldsymbol{\mu}}_k := \mathbf{0}_{N+1}$, and $\forall i \in [N]$, $B_i := I_{N+1}$, $\boldsymbol{z}_i := \mathbf{0}_{N+1}$, $\hat{\boldsymbol{\theta}}_i := \mathbf{0}_{N+1}$.
Foreach $t = 1, 2, ..., T$ **do**
  observe $\boldsymbol{c}^V(t)$
  Foreach context feature $i \in C \setminus C^V$ **do**
    Sample $\tilde{\boldsymbol{\theta}}_i$ from $\mathcal{N}(\hat{\boldsymbol{\theta}}_i, \alpha^2 B_i^{-1})$
  End do
  Sort $(c^V(t)^\top \tilde{\boldsymbol{\theta}}_1, ...\boldsymbol{c}^V(t))^\top \tilde{\boldsymbol{\theta}}_N)$ in decreasing order
  Select $C^U(t) := \{i \in C \setminus C^V, \boldsymbol{c}^V(t)^\top \tilde{\boldsymbol{\theta}}_i \geq \boldsymbol{c}^V(t)^\top \tilde{\boldsymbol{\theta}}_V\}$
  observe values $\boldsymbol{c}^{V+U}(t)$
  Foreach arm $k = 1, ..., K$ **do**
    Sample $\tilde{\boldsymbol{\mu}}_k$ from $\mathcal{N}(\hat{\boldsymbol{\mu}}_k, \alpha^2 A_k^{-1})$ distribution
  End do
  Select arm $k(t) := \arg\max_{k \subset [K]} \boldsymbol{c}^{V+U}(t)^\top \tilde{\boldsymbol{\mu}}_k$
  Observe $r_k(t)$
  $A_k := A_k + \boldsymbol{c}^{V+U}(t)\boldsymbol{c}^{V+U}(t)^\top, \quad \boldsymbol{g}_k := \boldsymbol{g}_k + \boldsymbol{c}^{V+U}(t) r_k(t), \quad \hat{\boldsymbol{\mu}}_k := A_k^{-1}\boldsymbol{g}_k$
  Foreach $i \in C^U$
    $B_i := \lambda(t) B_i + \boldsymbol{c}^V(t)\boldsymbol{c}^V(t)^\top, \quad \boldsymbol{z}_i := \boldsymbol{z}_i + \boldsymbol{c}^V(t) r_k(t), \quad \hat{\boldsymbol{\theta}}_i := \lambda(t) B_i^{-1} \boldsymbol{z}_i$
  End do
End do

We now propose an algorithm for solving the CAB problem, called *Context-Attentive Thompson Sampling (CATS)*, and summarize it in Algorithm 18. The basic idea of CATS is to use linear Thompson Sampling [Agrawal and Goyal, 2013] for solving the U linear bandit problems for selecting the set of additional relevant features C^{U^*} knowing $\boldsymbol{c}^V(t) \in C^V$, and for selecting the best arm knowing $\boldsymbol{c}^{V+U^*}(t) \in C^{V+U^*}$. Linear Thompson Sampling assumes a Gaussian prior for the likelihood function, which corresponds in CAB for arm k to $\tilde{\boldsymbol{\mu}}_k \sim \mathcal{N}(\boldsymbol{c}^{V+U}(t)^\top \hat{\boldsymbol{\mu}}_k, \alpha^2)$, and for feature i to, $\tilde{\boldsymbol{\theta}}_i \sim \mathcal{N}(\boldsymbol{c}^V(t)^\top \hat{\boldsymbol{\theta}}_i, \alpha^2)$. Then the posterior at time $t+1$ are respectively $P(\tilde{\boldsymbol{\mu}}_k | r_k(t)) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_k, \alpha^2 A^-1_k)$ for arm k, and $P(\tilde{\boldsymbol{\mu}}_k | r_k(t)) \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_i, \alpha^2 B^-1_k)$ for feature i.

The algorithm takes the total number of features N, the number of features initially observed V, the number of additional features to observe U, the set of observed features C^V, the number of actions K, the time horizon T, the distribution parameter $\alpha > 0$ used in linear Thompson Sampling, and a function of time $\lambda(t)$, which is used for adapting the algorithm to non-stationary linear bandits.

At each iteration t the values $\boldsymbol{c}^V(t)$ of features in the subset C^V are observed (line 4 Algorithm 18). Then the vector parameters $\tilde{\boldsymbol{\theta}}_i$ are sampled for each feature $i \in C^V$ (lines 5-7) from the posterior distribution (line 6). Then the subset of best estimated features at time t is selected (lines 8-9). Once the feature vector $\boldsymbol{c}^{U(t)}$ is observed line 10, Linear Thompson Sampling is applied in steps 11-15 to choose an arm. When the reward of selected arm is observed (line 15) the parameters are updated lines 16-19.

Remark 19 (Algorithmic complexity)**.** *At each time set, Algorithm 18 sorts a set of size V and inverts $U+1$ matrices in dimensions $N+1$ that leads to an algorithmic complexity in $O(V \log V + (U+1)(N+1)^2)T$.*

Due to assumption 2, CATS algorithm benefits from a linear algorithmic complexity, overcoming the negative result stated in [Foster et al., 2016]. Before providing an upper bound of the regret of CATS in $\tilde{O}(\sqrt{T})$ ($\tilde{O}$ hides logarithmic factors), we need an additional assumption on the noise $\eta_k(t)$.

Assumption 5 (Sub-Gaussian noise)**.** $\forall\, C^{U+V} \in C$ *and* $\forall k \in [K]$*, the noise* $\eta_k(t) = r_{k(t)} | \boldsymbol{c}^{V+U}(t) - \boldsymbol{c}^{U+V}(t)^\top \mu_k$ *is conditionally* ρ*-sub-Gaussian with* $\rho \geq 0$*, that is for all* $t \geq 1$*,*

$$\forall\, \lambda \in \mathbb{R}, \quad \mathbb{E}[e^{\lambda \eta_k(t)}] \leq \exp\left(\frac{\lambda^2 \rho^2}{2}\right) .$$

Lemma 8. *(Theorem 2 in [Agrawal and Goyal, 2013]) When the measurement noise $\eta_k(t)$ satisfies Assumption 5, $\forall k \in [K]$, $\forall t \in \{1, ..., T\}$ $||\boldsymbol{c}_k(t) \leq 1||$, $||\boldsymbol{\mu}_k \leq 1||$, $\alpha = \rho\sqrt{9N \log \frac{T}{\delta}}$, and $\epsilon = \frac{1}{\log T}$ the regret $R(T)$ of Thompson Sampling in the Linear bandit problem with K parameters is upper bounded with a probability $1-\delta$ by:*

$$O\left(N\sqrt{KT \log K}(\log T)^{3/2} \log \frac{1}{\delta}\right), \text{ where } 0 < \delta < 1.$$

We can now derive the following result.

Theorem 21. *When the measurement noise $\eta_k(t)$ satisfies Assumption 5,$\forall k \in [K]$, $\forall t \in \{1, ..., T\}$ $||\boldsymbol{c}_k(t) \leq 1||$, $||\boldsymbol{\mu}_k \leq 1||$, $\alpha = \rho\sqrt{9N \log \frac{T}{\delta}}$, $\lambda(t) = 1$ and $\epsilon = \frac{1}{\log T}$ the regret $R(T)$ of CATS (Algorithm 2) is upper bounded with a probability $1-\delta$ by:*

$$O\left(\left((U+V)\sqrt{KT \log K} + UV\sqrt{(N-V)T \log(N-V)}\right)(\log T)^{3/2} \log \frac{1}{\delta}\right).$$

Proof. For upper bounding the left term of the regret (see Property 2), we apply Lemma 8, and for upper bounding the right term, which is the regret of Thompson Sampling in U linear bandit problems in V dimensions with respectively $N-V, N-V-1, ..., N-V-U-1$ parameters, we apply Lemma 8. □

Theorem 2 states that the regret of CATS depends on the following two terms: the left term is the regret due to selecting a sub-optimal arm, while the right term is the regret of selecting a sub-optimal subset of features. We can

see that there is still a gap between the lower bound of the *Context Attentive Bandit* problem and the upper bound of the proposed algorithm. The left term of the lower bound scales in $\Omega(\sqrt{(U+V)T})$, while the left term of the upper bound of CATS scales in $\tilde{O}((U+V)\sqrt{KT\log K})$, where $\tilde{O}$ hides logarithmic factor. The right term of the lower bound scales in $\Omega(U\sqrt{VT})$, while the right term of the upper bound of CATS scales in $\tilde{O}((U+V)\sqrt{(N-V)T\log(N-V)})$. These gaps are due to the upper bound of regret of CATS, which uses Lemma 8. This suggests that the use of linear bandits based on an upper confidence balls, which scale in $\tilde{O}\sqrt{dT}$ [Abbasi-yadkori et al., 2011] (d is the dimension of contexts), could reduce this theoretical gap. As we show in the next section, we choose the Thompson Sampling approach for its better empirical performances.

4.3.6 Experiments

We compare the proposed CATS algorithm with:

(1) Random-EI: in addition to the V observed features, this algorithm selects a Random subset of features of the specified size U at each Iteration (thus, Random-EI), and then invokes the linear Thompson sampling algorithm.

(2) Random-fix: this algorithm invokes linear Thompson sampling on a subset of $U+V$ features, where the subset V is randomly selected once prior to seeing any data samples, and remains fixed.

(3) The state-of-art method for context-attentive bandits proposed in [Bouneffouf et al., 2017], Thompson Sampling with Restricted Context (TSRC): TSRC solves the CBRC (contextual bandit with restricted context) problem discussed earlier: at each iteration, the algorithm decides on $U+V$ features to observe (referred to as *unobserved context*). In our setting, however, V out of N features are immediately observed at each iteration (referred to as *known context*), then TSRC decision mechanism is used to select U additional unknown features to observe, followed by linear Thompson sampling on U+V features.

(4) CALINUCB: where we replace the contextual TS in CATS with LINUCB.

(5) CATS-fix: is heuristic where we stop the features exploration after some iterations $T' = 10\%, 20\%....90\%$ (we report here the an average over the best results).

Evaluation on public datasets

Empirical evaluation of Random-fix, Random-EI, CATS-fix, CALINUCB, CATS [8] and TSRC was performed on several publicly available datasets, as well as on a proprietary corporate dialog orchestration dataset. Publicly available Covertype and CNAE-9 were featured in the original TSRC paper and Warfarin [Sharabiani et al., 2015] is a historically popular dataset for evaluating bandit methods.

To simulate the known and unknown context space, we randomly fix 10% of the context feature space of each dataset to be known at the onset and explore a subset of U unknown features. To consider the possibility of

[8] Note that we have used the same exploration parameter value used in [Chapelle and Li, 2011] for TS and LINUCB type algorithms which are $TS \in \{0.25, 0.5, 1\}$ and $LINUCB \in \{0.51, 1, 2\}$

nonstationarity in the unknown context space over time, we introduce a weight decay parameter $\lambda(t)$ that reduces the effect of past examples when updating the CATS parameters. We refer to the stationary case as CATS and fix $\lambda(t) = 1$. For the nonstationary setting, we simulate nonstationarity in the unknown feature space by duplicating each dataset, randomly fixing the known context in the same manner as above, and shuffling the unknown feature set - label pairs. Then we stochastically replace events in the original dataset with their shuffled counterparts, with the probability of replacement increasing uniformly with each additional event. We refer to the nonstationary case as NCATS and use $\lambda(t)$ as defined by the GP-UCB algorithm [Srinivas et al., 2009]. We compare NCATS to NCATS-fix and NCALINUCB which are the non stationary version of CATS-fix and CALINUCB. we have also compare NCATS to the Weighted TSRC (WTSRC), the nonstationary version of TSRC also developed by [Bouneffouf et al., 2017]. WTSRC makes updates to its feature selection model based only on recent events, where recent events are defined by a time period, or "window" w. We choose $w = 100$ for WTSRC. We report the total average reward divided by T over 200 trials across a range of U corresponding to various percentages of N for each algorithm in Tables 4.6, 4.7.

Table 4.6: Stationary setting: total average reward, $V = 10\%$

	Warfarin		
U	20%	40%	60%
TSRC	53.28 ± 1.08	57.60 ± 1.16	59.87 ± 0.69
CATS	53.65 ± 1.21	**58.55 ± 0.67**	**60.40 ± 0.74**
CATS-fix	**53.99 ± 1.02**	58.67 ± 0.65	60.07 ± 0.54
CALINUCB	52.17 ± 0.89	57.23 ± 0.53	60.29 ± 0.66
Random-fix	51.05 ± 1.31	53.55 ± 0.97	55.15 ± 0.83
Random-EI	43.65 ± 1.21	48.55 ± 1.67	50.40 ± 1.33
	Covertype		
U	20%	40%	60%
TSRC	54.64 ± 1.87	63.35 ± 1.87	69.59 ± 1.72
CATS	65.57 ± 2.17	**72.58 ± 2.36**	78.58 ± 2.35
CATS-fix	**65.88 ± 2.01**	72.67 ± 2.13	78.55 ± 2.25
CALINUCB	61.99 ± 1.53	72.54 ± 1.76	**79.69 ± 1.82**
Random-fix	53.11 ± 1.45	59.67 ± 1.07	64.18 ± 1.03
Random-EI	46.15 ± 2.61	52.55 ± 1.81	55.45 ± 1.5
	CNAE-9		
U	20%	40%	60%
TSRC	**33.57 ± 2.43**	38.62 ± 1.68	**42.05 ± 2.14**
CATS	29.84 ± 1.82	39.10 ± 1.41	40.52 ± 1.42
CATS-fix	29.82 ± 1.70	**39.57 ± 1.23**	41.43 ± 1.39
CALINUCB	28.53 ± 1.65	38.88 ± 1.35	39.73 ± 1.36
Random-fix	33.01 ± 1.82	37.67 ± 1.68	39.18 ± 1.52
Random-EI	32.05 ± 2.01	36.65 ± 1.90	37.47 ± 1.75

The results in Tables 4.6, 4.7 are promising, with *our methods outperforming the state of the art in the majority of cases across both settings*. The most notable exception is found for CNAE-9 dataset, where CATS sometimes outperforms or nearly matches TSRC performance. This outcome is somewhat expected, since in the original work on TSRC [Bouneffouf et al., 2017], the mean error rate of TSRC was only 0.03% lower than the error corresponding to randomly fixing a subset of unknown features to reveal for each event on CNAE-9. This observation suggests

Table 4.7: Nonstationary setting: total average reward, $V = 10\%$

	Warfarin		
U	20%	40%	60%
WTSRC	55.83 ± 0.55	58.00 ± 0.83	59.85 ± 0.60
NCATS	**59.47 ± 2.89**	**59.34 ± 2.04**	**63.26 ± 0.75**
NCATS-fix	59.01 ± 3.09	59.14 ± 2.33	62.42 ± 0.98
NCLINUCB	58.64 ± 2.77	58.43 ± 1.89	63.01 ± 0.66
Random-fix	43.91 ± 1.17	47.67 ± 1.08	54.18 ± 1.03
Random-EI	47.78 ± 2.11	52.55 ± 1.83	55.45 ± 1.54
	Covertype		
U	20%	40%	60%
WTSRC	**50.26 ± 1.58**	58.99 ± 1.81	64.91 ± 1.38
NCATS	48.50 ± 1.05	**68.17 ± 3.14**	**83.78 ± 5.51**
NCATS-fix	49.87 ± 1.20	68.04 ± 3.24	82.98 ± 5.83
NCLINUCB	48.12 ± 0.99	68.20 ± 3.11	83.91 ± 5.21
Random-fix	43.11 ± 3.05	49.67 ± 2.77	53.18 ± 2.33
Random-EI	44.45 ± 4.44	46.65 ± 3.88	53.45 ± 3.61
	CNAE-9		
U	20%	40%	60%
WTSRC	19.91 ± 2.67	30.86 ± 2.92	36.01 ± 2.88
NCATS	30.88 ± 0.96	**34.91 ± 1.93**	**42.04 ± 1.52**
NCATS-fix	29.92 ± 1.06	33.43 ± 1.83	40.04 ± 1.51
NCLINUCB	**31.07 ± 0.87**	34.61 ± 1.73	41.81 ± 1.62
Random-fix	13.01 ± 3.45	21.77 ± 3.08	24.18 ± 2.43
Random-EI	16.15 ± 2.44	22.55 ± 2.18	25.45 ± 2.15

that, for this particular dataset, there may not be a subset of features which would be noticeably more predictive of the reward than the rest of the features. We also observe that the LINUCB version of CATS has comparable performance with CATS with slight advantage to CATS. Another observation is that CATS-fix is performing better than CATS in some situations, the explanation could be that after finding the best features the algorithm do not need to explore anymore and focus on finding the best arms based on these features. Note that there may not be a subset of features which would be noticeably more predictive of the reward than the rest of the features, which is the underlying assumption of TSRC. On top of this assumption, CATS also assumes that there exist relationships between the known and unknown context features, which is likely to cause a small compounding of error.

We perform a deeper analysis of the Covertype dataset, examining multi-staged selection of the U unknown context feature sets. In CATS, the known context is used to select all U additional context feature sets at once. In a multi-staged approach, the known context grows and is used to select each of the U additional context features incrementally (one feature at a time). Maintaining $\lambda(t) = 1$, for the stationary case we denote these two cases of the CATS algorithm as CATS and CATS-Staged respectively and report their performance when 10% of the context is randomly fixed, across various U in Figure 4.14.

Note that when the remaining 90% of features are revealed, the CATS and TSRC methods all reduce to simple linear Thompson sampling with the full feature set. Similarly, when 0 additional feature sets are revealed, the methods all reduce to linear Thompson sampling with a sparsely represented known context. Observe that CATS

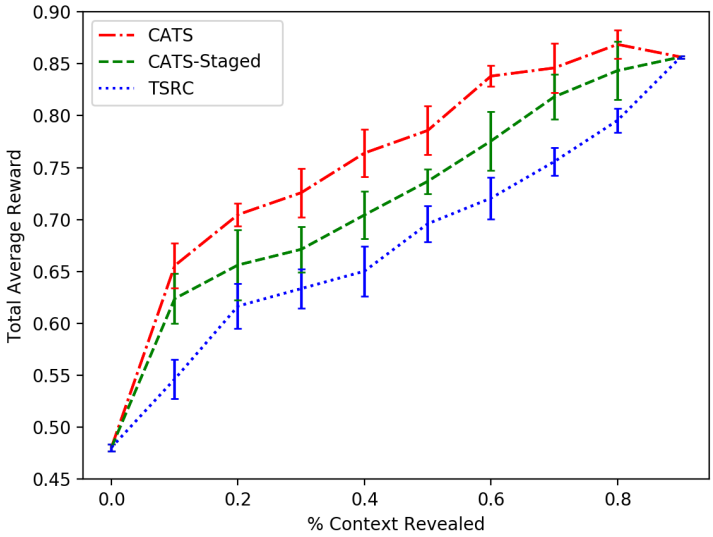


Figure 4.14: Stationary Setting: total Average Reward for Covertype

consistently outperforms CATS-Staged across all U tested.

CATS-Staged likely suffers because incremental feature selection adds nonstationarity to the known context - CATS learns relationships between the known and unknown features while CATS-Staged learns relationships between them as the known context grows. Nonetheless, both methods outperform TSRC. In the nonstationary case we use the GP-UCB algorithm for $\lambda(t)$, refer to the single and multi-staged cases as NCATS and NCATS-Staged, and illustrate their performance in Figure 4.15. Here we observe that NCATS and NCATS-Staged have comparable performance, and the improvement gain over baseline, in this case WTSRC, is even greater than in the stationary case.

Customer Assistant Evaluation

Next we evaluate our methods on *Customer Assistant*, a proprietary multi-skill dialog orchestration dataset. Recall that this kind of application motivates the CAB setting because there is a natural divide between the known and unknown context spaces; the query and its associated features are known at the onset and the potential responses and their associated features are only known for the domain specific agents the query is posed to. In this case, nonstationarity in the unknown context space could reflect independent updates to the underlying domain specific agents, perhaps improving or expanding their responses. As a result, similar queries, defining the known context, could elicit vastly different distributions of response features, the unknown context, over time. *The Customer Assistant* orchestrates 9 domain specific agents which we arbitrarily denote as $Skill_1, \ldots, Skill_9$ in the discussion that follows. In this application, example skills lie in the domains of payroll, compensation, travel, health benefits, and so on. In addition to a textual response to a user query, the skills orchestrated by *Customer Assistant* also return

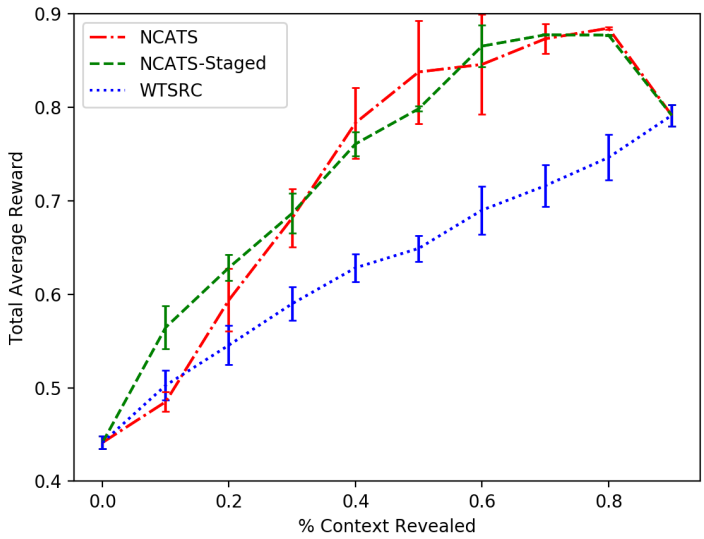


Figure 4.15: Nonstationary Setting: total Average Reward for Covertype

the following features: an *intent*, a short string descriptor that categorizes the perceived intent of the query, and a *confidence*, a real value between 0 and 1 indicating how confident a skill is that its response is relevant to the query. Skills have multiple intents associated with them. The orchestrator uses all the features associated with the query and the candidate responses from all the skills to choose which skill should carry the conversation.

The Customer Assistant dataset contains 28,412 events associated with a correct skill response. We encode each query by averaging 50 dimensional GloVe word embeddings [Pennington et al., 2014] for each word in each query and for each skill we create a feature set consisting of its confidence and a one-hot encoding of its intent. The skill feature set size for $Skill_1, \ldots, Skill_9$ are 181, 9, 4, 7, 6, 27, 110, 297, and 30 respectively.

We concatenate the query features and all of the skill features to form a 721 dimensional context feature vector for each event in this dataset. Recall that there is no need for simulation of the known and unknown contexts; in a live setting the query features are immediately calculable or known, whereas the confidence and intent necessary to build a skill's feature set are unknown until a skill is executed. Because the confidence and intent for a skill are both accessible post execution, we reveal them together. We accommodate this by slightly modifying the objective of CATS to reveal U unknown skill feature sets instead of U unknown individual features for each event. We perform a deeper analysis of the *Customer Assistant* dataset, examining multi-staged selection of the U unknown context feature sets. Maintaining $\lambda(t) = 1$, for the stationary case the results are summarized in Figure 4.16. Here both CATS-Staged and CATS methods outperform TSRC by a large margin.

For the nonstationary case we simulate nonstationarity in the same manner as the publicly available datasets, except using the natural partition of the query features as the known context and the skill feature sets as the unknown context instead of simulated percentages. We use the GP-UCB algorithm for $\lambda(t)$ and illustrate the performance of

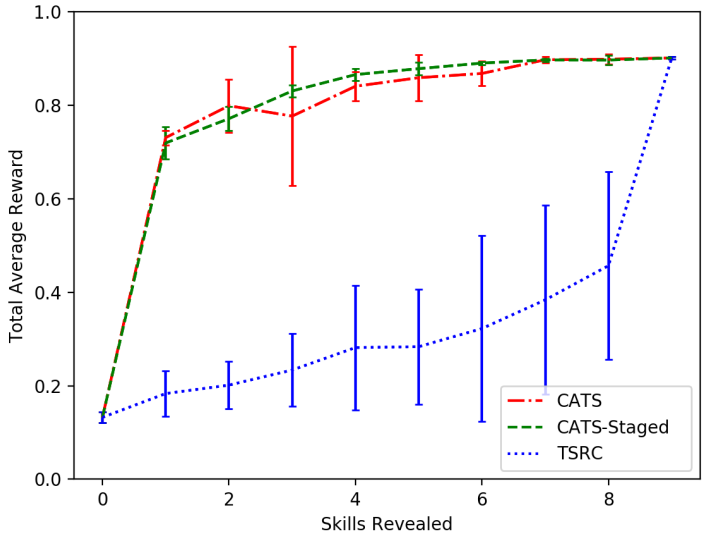


Figure 4.16: Stationary Setting: total Average Reward for Customer Assistant

NCATS and NCATS-Staged alongside WTSRC in Figure 4.17. Here we observe that NCATS slightly outperforms NCATS-Staged, and both outperform the WTSRC baseline.

4.3.7 Conclusions and Future Work

We have introduced here a novel bandit problem with only partially observable context and the option of requesting a limited number of additional observations. We also propose an algorithm, designed to take an advantage of the initial partial observations in order to improve its choice of which additional features to observe, and demonstrate its advantages over the prior art, a standard context-attentive bandit with no partial observations of the context prior to feature selection step. Our problem setting is motivated by several realistic scenarios, including medical applications as well as multi-domain dialog systems. Note that our current formulation assumes that all unobserved features have equal observation cost. However, a more practical assumption is that some features may be more costly than others; thus, in our future work, we plan to expand this notion of budget to accommodate more scenarios involving different feature costs. Other directions for future work include using non-bandit algorithms in the context feature selection stage.

4.3.8 Broader Impact

This problem has broader impacts in several domains such as voice assistants, healthcare and e-commerce.

- *Better medical diagnosis.* In a clinical setting, it is often too costly or infeasible to conduct all possible tests; therefore, given the limit on the number of tests, the doctor must decide which subset of tests will result into

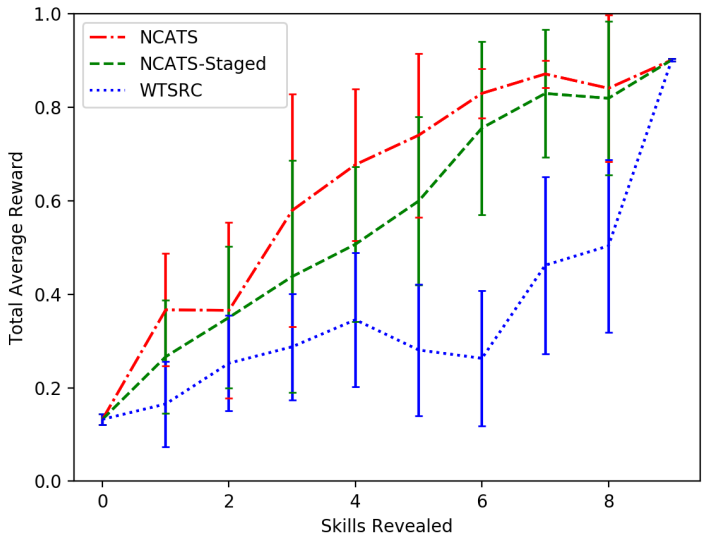


Figure 4.17: Nonstationary Setting: total Average Reward for Customer Assistant

maximally effective treatment choice in an iterative manner. A doctor may first take a look at patient's medical record to decide which medical test to perform, before choosing a treatment plan.

- *Better user preference modeling.* Our approach can help to develop better chatbots and automated personal assistants. For example, following a request such as, for example, "play music", an AI-based home assistant must learn to ask several follow-up questions (from a list of possible questions) to better understand the intent of a user and to remove ambiguities: e.g., what type of music do you prefer (jazz, pop, etc)? Would you like it on hi-fi system or on TV? And so on. Another example: a support desk chatbot, in response to user's complaint ("My Internet connection is bad") must learn to ask a sequence of appropriate questions (from a list of possible connection issues): how far is your WIFI hotspot? Do you have a 4G subscription? These scenarios are well-handled by the framework we proposed in this paper.

- *Better recommendations.* Voice assistants and recommendation systems in general tend to lock us in our preferences, which can have deleterious effects: e.g., recommendations based only on the past history of user's choices may reinforce certain undesirable tendencies, e.g., suggesting an online content based on a user's with particular bias (e.g., racist, sexist, etc). On the contrary, our approach could potentially help a user to break out of this loop, by suggesting the items (e.g. news) on additional questions (additional features) which can be used to broaden user's horizons.

Chapter 5

Bandits in multi-agent environments

5.1 Collaborative Exploration in Multi-Armed Bandits

Abstract. We consider the *decentralized exploration problem*: a set of players collaborate to identify the best arm by asynchronously interacting with the same stochastic environment. The objective is to ensure privacy in the best arm identification problem between asynchronous, collaborative, and thrifty players. In the context of a digital service, we advocate that this decentralized approach allows a good balance between conflicting interests: the providers optimize their services, while protecting privacy of users and saving resources. We define the privacy level with respect to the amount of information an adversary could infer by intercepting all the messages concerning a single user. We provide a generic algorithm DECENTRALIZED ELIMINATION, which uses any best arm identification algorithm as a subroutine. We prove that this algorithm ensures privacy, with a low communication cost, and that in comparison to the lower bound of the best arm identification problem, its sample complexity suffers from a penalty depending on the inverse square of the probability of the most frequent players. Then, thanks to the generality of the approach, we extend the proposed algorithm to the non-stationary bandits. Finally, experiments illustrate and complete the analysis.

5.1.1 Introduction

Motivations

For promoting their products in the digital world, most of companies perform marketing or advertising campaigns. For instance, when a cookie visits a web page, a banner containing marketing information is printed. If the cookie clicks on it, it is considered as a positive outcome. For maximizing the clicks on their banners, the companies need to carefully choose their messages. For instance for promoting Orange TV, should I highlight sport, series, movies,... (figure 5.1) ?

Figure 5.1: A/B testing - choosing the best message for promoting Orange TV.

For performing A/B testing, the interactions of the users with the service (application, web page...) and notably the clicks of the users are gathered by a server, and then are processed to find the best option. It works well, but this functional architecture (figure 5.3) allow companies to log all interactions of all users: privacy cannot be guaranteed.

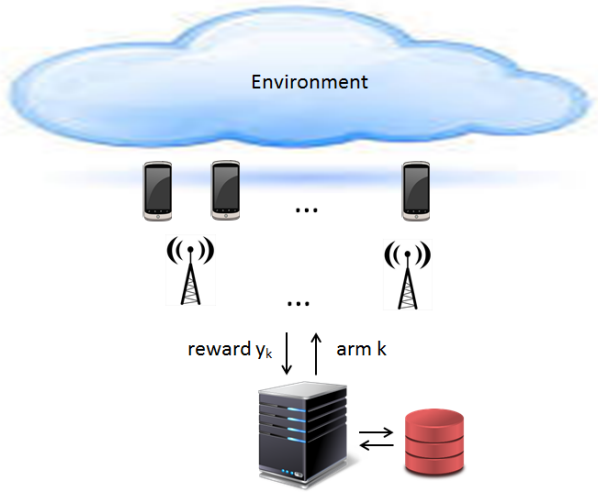


Figure 5.2: A/B testing: functional architecture.

The proposed approach consists in performing decentralized exploration for finding the best arm with high probability. The application is run in the devices of the users and the interactions of the users stay in the devices, and hence privacy of users is ensured. The issue of this architecture is that the needed number of interactions for finding the best option is in the order of few hundreds. For a centralized server that shares the experiences of thousands of users it is small, but for a single user it is too much. That is why, we allow the devices to exchange messages (figure 5.3)

5.1.2 Principle

When the event *"player n is active"* occurs, player n reads the messages received from other players and then chooses an arm to play. The reward of the played arm is revealed to player n. Finally, she may send a message to the other players for sharing information about the arms.

The decentralized approach presents significant advantages. First, the clicks of users contain information that

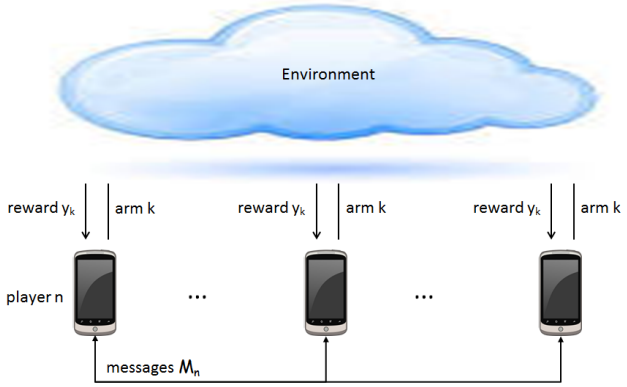


Figure 5.3: Collaborative Exploration.

may be embarrassing when revealed, or that can be used by a third party in an undesirable way. The decentralization of exploration favors privacy since the click stream is not transmitted. However, it is not sufficient. The messages sent by a user may still contain private information such as her favorite topics, and therefore her political views, sexual orientation... As the players broadcast messages to other players, a malicious adversary can pretend to be a player, and then listening the exchanged messages. To ensure privacy one must guarantee that no useful information can be inferred from the messages sent by a single user. Second, the decentralization of exploration reduces the communication cost. This is a significant requirement for the Internet of Thing applications, since the smart devices often run on batteries. Third and finally, for all digital applications and in particular for the mobile phone applications, the decentralization with a low communication cost increases the responsiveness of applications by minimizing the number of interactions between the application server and the devices.

Finally, the objective of the *decentralized exploration problem* is threefold:

1. *sample efficiency*: finding a near-optimal arm with high-probability using a minimal number of interactions with the environment.
2. *user privacy*: protecting information contained in the interaction history of a single player.
3. *low communication cost*: minimizing the number of exchanged messages.

Related works

The problem of the best arm identification has been studied in two distinct settings in the literature:

- the fixed budget setting: the duration of the exploration phase is fixed and is known by the forecaster, and the objective is to maximize the probability of returning the best arm [Bubeck et al., 2009, Audibert and Bubeck, 2010, Gabillon et al., 2012];
- the fixed confidence setting: the objective is to minimize the number of rounds needed to achieve a fixed confidence to return the best arm [Even-Dar et al., 2002, Kalyanakrishnan et al., 2012, Gabillon et al., 2012, Kaufmann and Kalyanakrishnan, 2013].

In this paper, we focus on the fixed confidence setting. Its theoretical analysis is based on the *Probably Approximately Correct* framework [Valiant, 1984], and focuses on the sample complexity to identify a near-optimal arm with high probability. This theoretical framework has been used to analyze the best arm identification problem in [Even-Dar et al., 2002], the dueling bandit problem in [Urvoy et al., 2013], the batched bandit problem in [Perchet et al., 2015], the linear bandit problem in [Soare et al., 2014], the contextual bandit problem in [Féraud et al., 2016], and the non-stationary bandit problem in [Allesiardo et al., 2017].

The *decentralized multi-player multi-armed bandits* have been studied for opportunistic spectrum access in [Liu and Zhao, 2010, Avner and Mannor, 2014, Nayyar et al., 2018] or for optimizing communications in Internet of Things, even when no sensing information is available [Besson and Kaufmann, 2018]. The objective is to avoid collisions between concurrent players that share the same channels, while choosing the best channels and minimizing the communication cost between players.

Recent years have seen an increasing interest for the study of the distributed collaborative scheme, where there is no collision when players choose the same arm at the same time. The distributed collaborative multi-armed bandits have been studied when the agents communicate through a neighborhood graph in [Szorenyi et al., 2013, Landgren et al., 2016]. Here, we allow each player to broadcast messages to all players. In [Chakraborty et al., 2017], a team of agents collaborate to handle the same multi-armed bandit problem. At each step the agent can broadcast her last obtained reward for the chosen arm to the team or pull an arm. The communication cost corresponds to the lost of the potential reward. As the pull of the arm of the agent is broadcasted, this approach does not ensure privacy of users. The tradeoff between the communication cost and the regret has been studied in the case of distributed collaborative non-stochastic experts [Kanade et al., 2012]. In [Hernández-Lobato et al., 2017], the best arm identification task with fixed budget is distributed using Thompson Sampling in order to accelerate the exploration of the chemical space. In [Hillel et al., 2013], the best arm identification task with fixed confidence is distributed on a parallel processing architecture. The analysis focuses on the trade-off between the number of communication rounds and the number of pulls per player. Here, we consider here that the players activation is under the control of the environment. As a consequence, synchronized communication rounds can no longer be used to control the communication cost. In our paper, the cost of communications is assessed by the number of exchanged messages.

Moreover, our purpose is also to protect privacy of players. In the current context of massive storage of personal data and massive usage of models inferred from personal data, privacy is an issue. Even if individual data are anonymized, the pattern of data associated with an individual is itself uniquely identifying. The k-anonymity approach [Sweeney, 2002] provides a guarantee to resist to direct linkage between stored data and the individuals. However, this approach can be vulnerable to composition attacks: an adversary could use side information that combined with the k-anonymized data allows to retrieve a unique identifier [Ganta et al., 2008]. The *differential privacy* [Dwork et al., 2006] provides an alternative approach. The sensitive data are hidden. The guarantee is

provided by algorithms that allow to extract information from data. An algorithm is differentially private if the participation of any record in the database does not alter the probability of any outcome by very much. The *differential privacy* has been extended to *local differential privacy* in which the data remains private even from the learner [Duchi et al., 2014]. In [Gajane et al., 2018], the authors propose an approach which handles the stochastic multi-armed bandit problem, while ensuring *local differential privacy*. The ϵ-differential privacy is ensured to the players by using a stochastic corruption of rewards. As all the rewards are transmitted to a centralized bandit algorithm, this approach has the maximum communication cost. Here, we define the privacy level with respect to the information about the preferred arms of a player, that an adversary could infer by intercepting the messages of this player. The messages could be corrupted feedbacks as in [Gajane et al., 2018], or as we choose a more compact representation of the same information.

Our contribution

In Section 2, we propose a new problem setting for ensuring privacy in the best arm identification problem between asynchronous, collaborative, and thrifty players. In Section 3, we propose a generic algorithm, DECENTRALIZED ELIMINATION, which handles the *decentralized exploration problem* using any best arm identification algorithm as a subroutine. Theorem 1 states that DECENTRALIZED ELIMINATION ensures privacy, finds an approximation of the best arm with high probability, and requires a low communication cost. Furthermore, Theorem 2 states a generic upper bound of the sample complexity of DECENTRALIZED ELIMINATION. More specifically, Corollary 1 and 2 state the sample complexity bound when respectively MEDIAN ELIMINATION and SUCCESSIVE ELIMINATION [Even-Dar et al., 2002] are used as subroutine. Then, in Section 4, we extend the algorithmic approach to the *decentralized exploration in non-stationary bandit problem*. In Section 5, to illustrate and complete the analysis, we empirically compare the performances of DECENTRALIZED ELIMINATION with two natural baselines [Kanade et al., 2012]: an algorithm that does not share any information between the players, and hence that ensures privacy with a zero communication cost, and a centralized algorithm that shares all the information between players, and hence that does not ensure privacy and that has the maximum communication cost.

5.1.3 The decentralized exploration problem

Let $\mathcal{N} = \{1, ..., N\}$ be a set of N players. Let $x \in \mathcal{N}$ be a discrete random variable which realization denotes the index n of the *active* player (the player for which an event occurs). Let P_x be the probability distribution of x which is assumed to be stationary and unknown to the players. Let $\mathcal{K} = \{1, ..., K\}$ be a set of K arms. Let $y_k^n \in [0, 1]$ be the bounded random variable which realization denotes the reward of arm k for player n, and μ_k^n be its mean reward. Let $\mathbf{y}_{x=n} = \{y_k^n\}_{k \in \mathcal{K}}$ be the vector of independent random variables y_k^n. Let $P_{\mathbf{y}}$ and $P_{x,\mathbf{y}}$ be respectively the probability distribution of $\mathbf{y}$ and the joint probability distribution of x and $\mathbf{y}$, which are assumed to be unknown to the

players.

Assumption 6 (stationary rewards)**.** *The mean reward of arms does not depend on time:* $\forall t$, $\forall n \in \mathcal{N}$,*and* $\forall k \in \mathcal{K}$, $\mu_k^n(t) = \mu_k^n$.

Assumption 7 (multi-armed bandits)**.** *The mean reward of arms does not depend on the player:* $\forall n \in \mathcal{N}$ *and* $\forall k \in \mathcal{K}$, $\mu_k^n = \mu_k$.

Assumption 6 and 7 are used to focus on the stochastic multi-armed bandits. This section lays the theoretical foundations of the *decentralized exploration problem* in its elementary form. The next section proposes an extension to the *decentralized exploration in non-stationary bandits*. The extension to the *decentralized exploration in contextual bandits* is discussed in future works.

Definition 24 (ϵ-optimal arm)**.** *An arm* $k \in \mathcal{K}$ *is said to be* ϵ*-optimal, if* $\mu_k \geq \mu_{k^*} - \epsilon$*, where* $k^* = \arg\max_{k \in \mathcal{K}} \mu_k$ *and* $\epsilon \in (0, 1]$. $\mathcal{K}_\epsilon$ *denotes the set of* ϵ*-optimal arms.*

Definition 25 (message)**.** *A message* $\lambda_k^n \in \{0, 1\}$ *is a binary random variable, that is sent by player* n *to other players, and where* $\lambda_k^n = 1$ *means that player* n *estimates that* k *is not an* ϵ*-optimal arm.*[1]

Let $\mathcal{M}_n$ be the set of sent messages by player n at stopping time. Let $\mathcal{K}^n(l^n) \subseteq \mathcal{K}$ be the set of remaining arms at epoch $l^n \in \{1, ..., L\}$ for player n, where L is the maximal number of epochs.

Definition 26 ((ϵ, η)-private)**.** *The decentralized algorithm* $\mathcal{A}$ *is* (ϵ, η)*-private for finding an* ϵ*-optimal arm, if for any player* n*, an adversary, that knows* $\mathcal{M}_n$*, the set of messages of player* n*, and the algorithm* $\mathcal{A}$*, cannot infer what arm is* ϵ*-optimal for player* n *with a probability higher than* $1 - \eta$*:*

$$\forall n \in \mathcal{N}, \forall l^n \in \{1, ..., L\}, \eta_1, 0 \leq \eta_1 < \eta \leq 1,$$
$$\mathbb{P}\left(\mathcal{K}^n(l^n) \subseteq \mathcal{K}_\epsilon | \mathcal{M}_n, \mathcal{A}\right) \geq 1 - \eta_1.$$

$1 - \eta$ is the confidence level associated to the decision of the adversary. If η is small, then the adversary can use the set of messages $\mathcal{M}_n$ to infer with high probability which arm is an ϵ-optimal arm for player n. If η is high, the only information, that can be inferred by the adversary, is that the probability that an arm is an ϵ-optimal of arm for player n is a little bit higher than 0, which can be much lesser than the random choice $1/K$. η is a parameter which allows to tune the level of privacy: the higher η, the higher the privacy protection.

The goal of the *decentralized exploration problem* (see Algorithm 19) is to design an algorithm, that, when run on each player, samples effectively to find an ϵ-optimal arm for each player, while ensuring (ϵ, η)-privacy to players, and minimizing the number of exchanged messages.

[1] We choose a Bernoulli random variable for the sake of clarity. Notice that any random variable could be used as message.

Algorithm 19 DECENTRALIZED EXPLORATION PROBLEM

Inputs: $\mathcal{K}$, $\epsilon \in [0,1]$, $\eta \in [0,1]$
Output: an arm in each set $\mathcal{K}^n(l^n)$
Initialization: $l^n := 1$, $\mathcal{K}^n(l^n) := \mathcal{K}$

1: **repeat**
2: a player is sampled: $n \sim P_x$
3: player n gets the messages of other players
4: arm $k \in \mathcal{K}^n(l^n)$ is played by player n
5: player n receives reward $y_k^n \sim P_{x=n,y}$
6: **if** player n updates $\mathcal{K}^n(l^n)$ **then** $l^n := l^n + 1$
7: player n sends a message to other players
8: **until** $(\forall n \in \mathcal{N}, |\mathcal{K}^n(l^n)| = 1)$

The lower bound of the number of samples in $P_{x,\mathbf{y}}$ needed to find with high probability an ϵ-optimal arm, which is $\Omega\left(\frac{K}{\epsilon^2}\log\frac{1}{\delta}\right)$ [Mannor and Tsitsiklis, 2004], holds for the *decentralized exploration problem*, since a message can be sent at each time an arm is sampled by a player. The number of messages, that has to be exchanged in order to find with high probability an ϵ-optimal arm, could be zero if each player independently handles the best arm identification problem.

Assumption 8 (all players are active)**.** $\forall n \in \mathcal{N}$, $P_x(x = n) \neq 0$.

Assumption 8 is a sanity check assumption for the *decentralized exploration problem*. Indeed, if it exists a player n such that $P_x(x = n) = 0$, then Algorithm 19 never stops (the stopping condition line 8 never happens).

5.1.4 Decentralized Elimination

ArmSelection subroutine

Before describing a generic algorithm for the *decentralized exploration problem*, we need to define an $\mathrm{ArmSelection}$ subroutine that handles all best arm identification algorithms. Let $\overline{\mathcal{K}^n}(l^n)$ and $\mathcal{K}^n(l^n)$ be respectively the set of eliminated arms and the set of remaining arms of player n at elimination epoch l^n, such that $\overline{\mathcal{K}^n}(l^n) \cup \mathcal{K}^n(l^n) = \mathcal{K}^n(l^n - 1)$.

Definition 27 (ArmSelection subroutine)**.** *an* $\mathrm{ArmSelection}$ *subroutine takes as parameters an approximation factor* ϵ*, a confidence level* $1 - \eta$*, and a set of remaining arm* $\mathcal{K}^n(l^n)$*. It samples a remaining arm in* $\mathcal{K}^n(l^n)$ *and returns the set of eliminated arms* $\overline{\mathcal{K}^n}(l^n)$*. An* $\mathrm{ArmSelection}$ *subroutine satisfies Properties 1 and 2.*

Let t^n be the number of calls of the $\mathrm{ArmSelection}$ subroutine. Let $\mathcal{H}_{t^n}$ be the sequence of rewards of chosen arms $\{(k_1, y_{k_1}^n), (k_2, y_{k_2}^n), ..., (k_{t^n}, y_{k_{t^n}}^n)\}$. Let $f : \{1, ..., L\} \to [0,1]$ be a function such that $\sum_{l^n=1}^{L} f(l^n) = 1$.

Property 1. (remaining ϵ-optimal arm)

$$\forall l^n \in \{1, ..., L\}, \mathcal{K}^n\left(l^n\right) \subset \mathcal{K}^n\left(l^n - 1\right),$$
$$\mathbb{P}\left(\{\mathcal{K}^n(l^n) \cap \mathcal{K}_\epsilon = \emptyset\} | \mathcal{H}_{t^n-1}, \mathcal{K}^n\left(l^n - 1\right) \cap \mathcal{K}_\epsilon \neq \emptyset\right) \leq \eta \times f(l^n).$$

Property 2. (finite sample complexity)

$$\forall \eta \in (0,1), \forall \epsilon \in (0,1], \exists t^n \geq 1,$$
$$\mathbb{P}\left(\{\mathcal{K}^n(L) \subset \mathcal{K}_\epsilon\} | \mathcal{H}_{t^n-1}\right) \geq 1 - \eta.$$

Property 1 ensures that with high probability at least an ϵ-optimal arm remains in the set of arms $\mathcal{K}^n(l^n)$, while Property 2 ensures that the $\mathrm{ArmSelection}$ subroutine finds in a finite time an ϵ-optimal arm whatever the confidence level $1 - \eta$ and the approximation factor ϵ. To the best of our knowledge, all best arm identification algorithms can be used as $\mathrm{ArmSelection}$ subroutine with straightforward transformations. We consider three classes of best arm identification algorithms.

The fixed-design algorithms use *uniform sampling* during a predetermined number of samples. NAIVE ELIMINATION ($L = 1$ and $f(l^n) = 1$) and MEDIAN ELIMINATION ($L = \log_2 K$ and $f(l^n) = 1/2^{l^n}$) [Even-Dar et al., 2002] are *fixed-design* algorithms which can be used as $\mathrm{ArmSelection}$ subroutines.

The successive elimination algorithms are based on *uniform sampling* and *arm eliminations*. At each time step a remaining arm is uniformly sampled. The empirical mean of the played arm is updated. The arms, which cannot be an ϵ-optimal arm with high probability, are discarded. If suboptimal arms are discarded the epoch l is increased by one. SUCCESSIVE ELIMINATION ($L = K$ and $f(l^n) = 1/K$) [Even-Dar et al., 2002], KL-RACING ($L = K$ and $f(l^n) = 1/K$) [Kaufmann and Kalyanakrishnan, 2013] are *successive elimination* algorithms which can be used as $\mathrm{ArmSelection}$ subroutines.

The explore-then-commit algorithms are based on *adaptive sampling* and *a stopping rule*. Rather than choosing arms uniformly, the *explore-then-commit* algorithms play one of the two critical arms: the empirical best arm, and the empirical suboptimal arm associated with the maximum upper confidence bound. The stopping rule simply tests if the difference, between the maximum of upper confidence bound of suboptimal arms and the lower confidence bound of the empirical best arm, is higher than the approximation factor ϵ. When the algorithm stops it returns the best arm. LUCB [Kalyanakrishnan et al., 2012], KL-LUCB [Kaufmann and Kalyanakrishnan, 2013], UGAPEC [Gabillon et al., 2012] can also be used as $\mathrm{ArmSelection}$ subroutines by returning the set of eliminated arms when the stopping event occurs ($L = 1$ and $f(l^n) = 1$).

Algorithm description

The basic idea of DECENTRALIZED ELIMINATION is to use the vote of independent players, which communicate the arm they would like to eliminate with a high probability of failure for ensuring privacy. As the players are independent, the probability of failure of the vote is the multiplication of the individual probability of failures. The number of players needed for eliminating an arm is provided by the analysis.

DECENTRALIZED ELIMINATION (see Algorithm 20) takes as parameters the privacy level η, the failure probability δ, the approximation factor ϵ, and an $\mathrm{ArmSelection}$ subroutine. It outputs an ϵ-optimal arm for each player with high probability. The algorithm sketch is described below.

When player n is active (i.e. when player n is sampled):

- player n gets messages from other players (line 3).
- When enough players have eliminated an arm, it is eliminated from the shared set of arms $\mathcal{K}(l)$ and from the set of arms $\mathcal{K}^n(l^n)$ of player n with a low probability of failure (lines 5-10).
- When there is only one arm in $\mathcal{K}(l)$, it is an ϵ-optimal arm with high probability $1-\delta$, and the set of arms of player n is $\mathcal{K}(l)$ (line 11).
- An $\mathrm{ArmSelection}$ subroutine, run with a low confidence level $1-\eta$ (i.e. high privacy level) on the set $\mathcal{K}^n(l^n)$, samples an arm and returns $\overline{\mathcal{K}}^n(l^n)$ the set of arms that player n has eliminated at step t^n (line 13).
- When player n has eliminated an arm, she communicates to other players the index of the arm (lines 14-20).

Analysis of the algorithm

Theorem 1 states the upper bound of the communication cost for obtaining with high probability an ϵ-optimal arm while ensuring (ϵ,η)-privacy to the players. The communication cost depends only on the problem parameters: the privacy constraint η, the probability of failure δ, the number of actions, and notably not on the number of samples. Notice that the probability of failure is low since the failure probability is lower than the level of privacy guarantee: $\delta < \eta$.

Theorem 22. *Using any* $\mathrm{ArmSelection}$ *subroutine,* DECENTRALIZED ELIMINATION *is an* (ϵ,η)*-private algorithm, that finds an* ϵ*-optimal arm with a failure probability* $\delta \leq \eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ *and that exchanges at most* $\lfloor \frac{\log \delta}{\log \eta} \rfloor K - 1$ *messages.*

To finely analyze the sample complexity of DECENTRALIZED ELIMINATION algorithm, one needs to handle the randomness of the voting process. Let $T_{P_{x,\mathbf{y}}}$ be the number of samples in $P_{x,\mathbf{y}}$ when DECENTRALIZED ELIMINATION stops with high probability. Let $T_{P_{\mathbf{y}}}$ be the number of samples in $P_{\mathbf{y}}$ needed by the $\mathrm{ArmSelection}$ subroutine to

Algorithm 20 DECENTRALIZED ELIMINATION

Inputs: $\epsilon \in (0,1]$, $\eta \in (0,1)$, $\delta \in [\eta^N, \eta^2]$, $\mathcal{K}$, an $\mathrm{ArmSelection}$ subroutine
Output: an arm in each set $\mathcal{K}^n(l^n)$
Initialization: $l := 1$, $\mathcal{K}(l) := \mathcal{K}$, $\forall n\ t^n := 1$, $l^n := 1$, $\mathcal{K}^n(l^n) := \mathcal{K}$, $\forall\,(k,n)\ \lambda_k^n := 0$

repeat
  player n is sampled: $n \sim P_x$
  player n gets the messages λ_k^j from other players
  if $|\mathcal{K}(l)| > 1$ **then**
    for all $k \in \mathcal{K}(l)$ **do**
      if $\sum_{j=1}^N \lambda_k^j \geq \lfloor \frac{\log \delta}{\log \eta} \rfloor$ **then**
        $\mathcal{K}(l) := \mathcal{K}(l) \setminus \{k\}$, $l := l+1$
        $\mathcal{K}^n(l^n) := \mathcal{K}^n(l^n) \setminus \{k\}$
      end if
    end for
    else $\mathcal{K}^n(l^n) := \mathcal{K}(l)$
  end if
  $\overline{\mathcal{K}}^n(l^n) := \mathrm{ArmSelection}(\epsilon, \eta, \mathcal{K}^n(l^n))$
  if $|\overline{\mathcal{K}}^n(l^n)| > 1$ **then**
    $l^n := l^n + 1$
    for all $k \in \overline{\mathcal{K}}^n(l^n)$ **do**
      $\mathcal{K}^n(l^n) := \mathcal{K}^n(l^n) \setminus \{k\}$
      $\lambda_k^n := 1$, λ_k^n is sent to other players
    end for
  end if
  $t^n := t^n + 1$
until $\forall n\ |\mathcal{K}^n(l^n)| = 1$

find an ϵ-optimal arm with high probability. Let $\mathcal{N}_M$ be the set of the $M = \lfloor \frac{\log \delta}{\log \eta} \rfloor$ most likely players, let $p^* = \min_{n \in \mathcal{N}_M} P_x(x=n)$, and let $p^\dagger = \min_{n \in \mathcal{N}} P_x(x=n)$.

Theorem 23. *Using any* $\mathrm{ArmSelection}$ *subroutine, with a probability higher than* $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ DECENTRALIZED ELIMINATION *stops after at most:*

$$T_{P_{x,\mathbf{y}}} = \frac{T_{P_\mathbf{y}}}{p^*} + \frac{1}{(p^*)^2}\sqrt{\frac{p^* T_{P_\mathbf{y}}}{2}\log\frac{1}{\delta}} + \frac{1}{2(p^*)^2}\log\frac{1}{\delta} \textit{ samples in } P_{x,\mathbf{y}},$$

where $I_a(b,c)$ *denotes the incomplete beta function evaluated at* a *with parameters* b, c.

As the number of players involved in the vote is set as small as possible $\lfloor \frac{\log \delta}{\log \eta} \rfloor$, Theorem 2 provides with high probability [2] the sample complexity of DECENTRALIZED ELIMINATION. Notice, that when the number of players is high, and when the distribution of players is far from the uniform distribution, we have $p^* \gg p^\dagger$.

Corollary 6. *With a probability higher than* $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ DECENTRALIZED MEDIAN ELIMINATION *stops after:*

$$\mathcal{O}\left(\left(\frac{K}{\lfloor \frac{\log \delta}{\log \eta} \rfloor \epsilon^2} + \frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta}\right) \textit{ samples in } P_{x,\mathbf{y}}.$$

[2] for instance, $I_{0.99}(500, 500) = 1.47 \times 10^{-302}$

Corollary 7. *With a probability higher than* $(1-\delta)\left(1-I_{1-p^*}\left(T_{P_{x,\mathbf{y}}}-T_{P_{\mathbf{y}}},1+T_{P_{\mathbf{y}}}\right)\right)^{\lfloor\frac{\log\delta}{\log\eta}\rfloor}$ DECENTRALIZED SUCCESSIVE ELIMINATION *stop after:*

$$\mathcal{O}\left(\left(\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2}+\frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta}+\frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}\right) \text{ samples in } P_{x,\mathbf{y}}.$$

Corollaries 6 and 7 state the number of samples in $P_{x,\mathbf{y}}$ needed to find an ϵ-optimal arm by DECENTRALIZED ELIMINATION using respectively MEDIAN ELIMINATION and SUCCESSIVE ELIMINATION as $\mathrm{ArmSelection}$ subroutines.

Remark 20. *In the case of the uniform distribution of players, with a failure probability at most* $\delta=\eta^N$ *the number of sample in* $P_{x,\mathbf{y}}$ *needed by* DECENTRALIZED MEDIAN ELIMINATION *to find an* ϵ*-optimal arm is:*

$$\mathcal{O}\left(\frac{K}{\epsilon^2}\log\frac{1}{\delta}+N^2\log\frac{1}{\delta}\right) \text{ samples in } P_{x,\mathbf{y}}.$$

In comparison to an optimal best arm identification algorithm, which communicates all the messages and does not provide privacy protection guarantee, which has a sample complexity in $\mathcal{O}\left(\frac{K}{\epsilon^2}\log\frac{1}{\delta}\right)$*, the sample complexity of* DECENTRALIZED ELIMINATION *mostly suffers from a penalty depending on the inverse of the probability of the most frequent players, that in the case of uniform distribution of players is quadratic with respect to the number of players.*

5.1.5 Decentralized exploration in non-stationary bandits

Recently, the best arm identification problem has been studied in the case of non-stationary bandits, where Assumption 6 does not hold [Allesiardo et al., 2017, Abbasi-Yadkori et al., 2018]. In the first reference, the authors analyze the non-stationary stochastic best-arm identification in the fixed confidence setting by splitting the game into independent sub-games where the best arm does not change. In the second reference, the authors propose a simple and anytime algorithm, which is analyzed for stochastic and adversarial rewards in the case of fixed budget setting. For the consistency of the paper, which focuses on fixed confidence setting, we choose to extend DECENTRALIZED ELIMINATION to SUCCESSIVE ELIMINATION with RANDOMIZED ROUND-ROBIN (SER3 [Allesiardo et al., 2017]). Basically, SER3 consists in shuffling the set of arms at each step of SUCCESSIVE ELIMINATION. SER3 works for the sequences where Assumption 9 holds.

Assumption 9 (Positive mean gap)**.** *For any* $k\in\mathcal{K}\setminus\{k^*\}$ *and any* $[\tau]\in\mathbb{T}(\tau)$ *with* $\tau\geq\log\frac{K}{\eta}$*, we have:*

$$\Delta_k^*\left([\tau]\right)=\frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{j=i}^{i+K_i-1}\frac{\Delta_{k^*,k}(j)}{K_i}>0,$$

where $\mathbb{T}(\tau)$ *is the set containing all possible realizations of* τ *round-robin steps,* $\Delta_{k^*,k}(t)$ *is the difference between*

the mean reward of the best arm and the mean reward of arm k at time t, and K_t is the number of remaining arms at time t.

We provide below the sample complexity bound of DECENTRALIZED SUCCESSIVE ELIMINATION with RANDOMIZED ROUND-ROBIN (DSER3), which is simply DECENTRALIZED ELIMINATION using SER3 as the ArmSelection subroutine.

Theorem 24. *For $K \geq 2$, $\delta \in (0, 0.5]$, $\epsilon \geq \delta/K$ for the sequences of rewards where Assumption 9 holds, DSER3 is an (ϵ, η)-private algorithm, that exchanges at most $\lfloor \frac{\log \delta}{\log \eta} \rfloor K - 1$ messages, that finds an ϵ-optimal arm with a probability at least $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_{\mathbf{y}}}, 1 + T_{P_{\mathbf{y}}}\right)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$, and that stops after:*

$$\mathcal{O}\left(\left(\frac{K}{\lfloor \frac{\log \delta}{\log \eta} \rfloor \epsilon^2} + \frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta} + \frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}\right)$$

samples in $P_{x,\mathbf{y}}$.

Finally DECENTRALIZED SUCCESSIVE ELIMINATION with RANDOMIZED ROUND-ROBIN AND RESET (DSER4) handles any sequence of rewards: when Assumption 9 does not hold a switch occurs (see figures 5.4, 5.5).

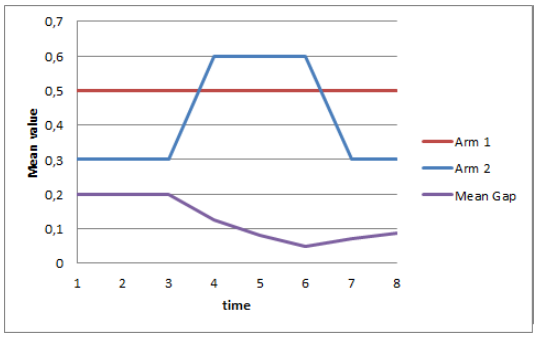


Figure 5.4: Mean gap versus time. Assumption 9 holds: the mean gap stays positive.

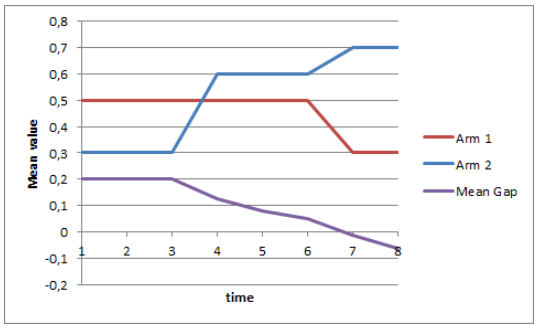


Figure 5.5: Mean gap versus time. Assumption 9 does not hold: a switch occurs a time 7.

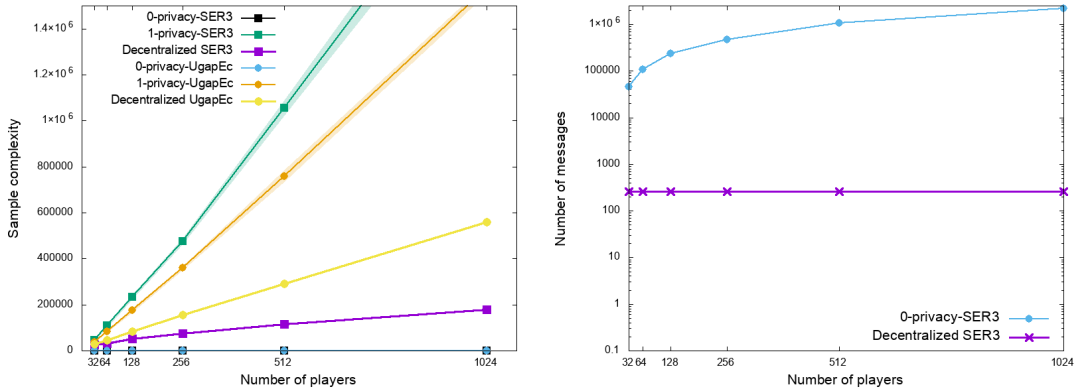
DSER4 consists in using SER4 [Allesiardo et al., 2017] as the ArmElimination subroutine in DECENTRALIZED ELIMINATION. In addition, when a reset occurs in SER4, DECENTRALIZED ELIMINATION is reset.

Theorem 25. *For $K \geq 2$, $\epsilon \geq \frac{\eta}{K}$, $\varphi \in (0,1]$, $\epsilon \geq \delta/K$ for any sequences of rewards,* DSER4 *is an (ϵ, η)-private algorithm, that exchanges on average at most $\varphi T(\lfloor \frac{\log \delta}{\log \eta} \rfloor K - 1)$ messages, and that plays, with an expected probability at most $\delta + \varphi T I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$, a suboptimal arm on average no more than:*

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\sqrt{\frac{\left(\frac{K}{p^*}\log K + \frac{K}{\lfloor \frac{\log \delta}{\log \eta} \rfloor}\right)\frac{S}{p^*}\log\frac{1}{\delta}}{\delta}} + \frac{1}{(p^*)^2}\log\frac{1}{\delta}\right),$$

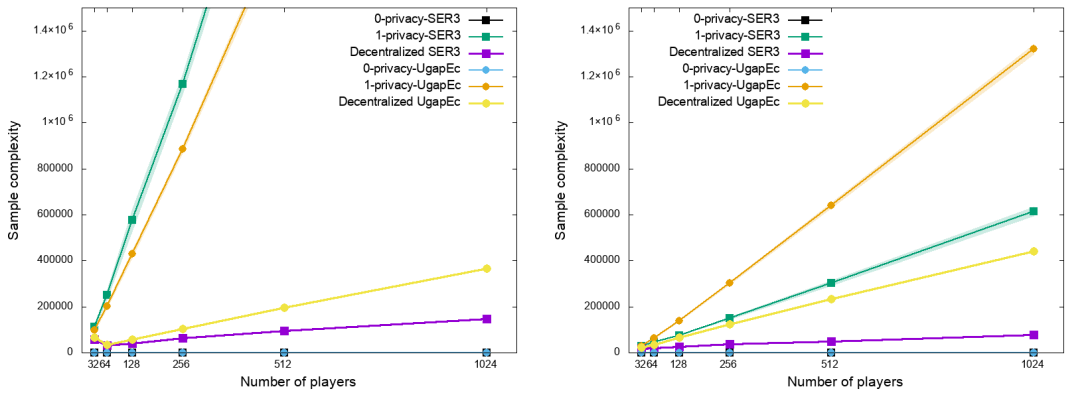
times, where S is the number of switches of best arms, φ is the probability of reset in SER4*, T is the time horizon, and the expected values are taken with respect to the randomization of resets.*

5.1.6 Experiments



(a) Problem 1 - Sample Complexity (b) Problem 1 - Number of messages

Figure 5.6: Uniform distribution of players. The sample complexities of 0-PRIVACY baselines are the same: 800.



(a) Problem 2 - Sample Complexity (b) Problem 3 - Sample Complexity

Figure 5.7: 50% of players generates 80% of events (a), and the mean rewards of suboptimal arms linearly decrease (b).

Experimental setting

To illustrate and complete the analysis of DECENTRALIZED ELIMINATION, we run three synthetic experiments:

- **Problem 1: Uniform distribution of players.** There are 10 arms. The optimal arm has a mean reward $\mu_1 = 0.7$, the second one $\mu_2 = 0.5$, the third one $\mu_3 = 0.3$, and the others have a mean reward of 0.1. Each player has a probability equal to $1/N$.
- **Problem 2: 50% of players generates 80% of events.** The same 10 arms are reused with an unbalanced distribution of players. The players are split in two groups of sizes $N/2$. When a player is sampled, a uniform random variable $z \in [0, 1]$ is drawn. If $z < 0.8$ the player is uniformly sampled from the first group, otherwise it is uniformly sampled from the second group.
- **Problem 3: non-stationary rewards.** The distribution of players is uniform. The same 10 arms are reused. The mean reward of the optimal arm does not change during time. The mean reward of suboptimal arms linearly decrease: $\mu(t) = \mu(0) - 10^{-5}t$.

As comparison points, we include two natural baselines:

- 1-PRIVACY: an $(\epsilon, 1)$-private algorithm that does not share any information between the players, and hence that runs at a zero communication cost. The ArmSelection subroutine is run with parameters $(\epsilon, \delta/N)$ to ensure that all the players find with a probability $1 - \delta$ an ϵ-optimal arm.
- 0-PRIVACY: an $(\epsilon, 0)$-private algorithm that shares all the information between players, and hence that runs at a minimal privacy and a maximal communication cost. This algorithm does not meet the original goal but is interesting as a reference to assess the sample efficiency loss stemming from the privacy constraint.

As ARMSELECTION subroutines, We choose two frequentist algorithmsbased on Hoeffding inequality: a *explore-then-commit* algorithm UGAPEC [Gabillon et al., 2012] and a *successive elimination* algorithm SER3 [Allesiardo et al., 2017], which handles non-stationary rewards. Combining DECENTRALIZED ELIMINATION and the two baselines with the two ARMSELECTION subroutines, we compare 6 algorithms (DECENTRALIZED SER3, DECENTRALIZED UGAPEC, 1-PRIVACY-SER3, 1-PRIVACY-UGAPEC, 0-PRIVACY-SER3, 0-PRIVACY-UGAPEC) on the three problems. The algorithms are compared with respect to two key performance indicators: the sample complexity and the communication cost. For all the experiments, ϵ is set to 0.25, and δ is set to 0.05. The privacy level η is set to 0.9. All the curves and the measures are averaged over 20 trials.

Results

The results reveal that the sample efficiency of 1-PRIVACY baselines is horrendous on both problems: it increases super-linearly as the number of players increases. Worse, when the distribution of players moves away from the

uniformity, which is the case in most of digital applications, the performances of 1-PRIVACY baselines decreases (see Figure 5.6a, 5.7a). Contrary to 1-PRIVACY baselines, the performances of DECENTRALIZED UGAPEC and DECENTRALIZED SER3 increases in Problem 2 (see Figure 5.7a). More precisely, the sample complexity curves of DECENTRALIZED UGAPEC and DECENTRALIZED SER3 exhibit two regimes: first the sample complexity decreases (between 32 to 64 players), and then the sample complexity linearly increases with the number of players. The values of hyper-parameters: $\delta = 0.05$ and $\eta = 0.9$, imply that the number $M = \lfloor \frac{\log \delta}{\log \eta} \rfloor$ of player votes required to eliminate an arm is 28. In Problem 2 with 32 players, it means that the algorithm has to wait for infrequent players votes to terminate. When the number of players is 64, this issue disappears. This is the reason why the sample complexity for 64 players is lower than for 32 players.

Concerning the ARMSELECTION subroutines, we observe that 1-PRIVACY-UGAPEC clearly outperforms 1-PRIVACY-SER3 on stationary problems (see Figures 5.6a and 5.7a). Moreover, the performance gain of 1-PRIVACY-UGAPEC increases with the number of players. This is due to the adaptive sampling strategy of UGAPEC: by sampling alternatively the empirical best arm and the most loosely estimated suboptimal arm, 1-PRIVACY-UGAPEC reduces the variance of the sample complexity, and thus reduces the maximum of sample complexities of players. However, when used as a subroutine in DECENTRALIZED ELIMINATION, the *successive elimination* algorithms such as SER3 are more efficient: thanks to the different suboptimal arms which are progressively eliminated by different groups of voting players, DECENTRALIZED SER3 clearly outperforms DECENTRALIZED UGAPEC (see Figure 5.6a and 5.7a).

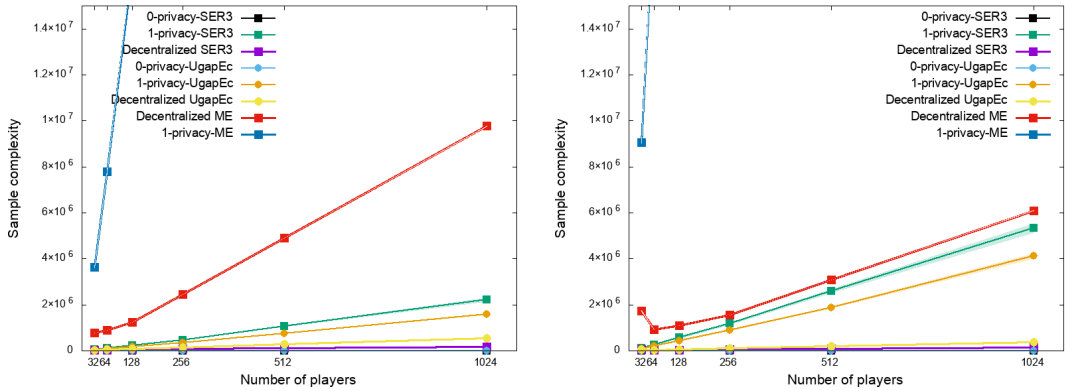
When the mean rewards of suboptimal arms are decreasing (Figure 5.7b), in comparison to SER3 the performances of UGAPEC, which is not designed for non-stationary rewards, collapse: 1-PRIVACY-UGAPEC and DECENTRALIZED UGAPEC are respectively outperformed by 1-PRIVACY-SER3 and DECENTRALIZED SER3. The optimistic approach used in the sampling rule of UGAPEC is too optimistic when the mean reward are decreasing.

The communication cost is the number of exchanged messages: 1-PRIVACY baselines send zero messages, while 0-PRIVACY baselines send $N - 1$ messages per time step until the ϵ-optimal arm is found. DECENTRALIZED SER3 needs three to four orders of magnitude less messages than 0-PRIVACY-SER3 (see Figure 5.6b).

Finally, MEDIAN ELIMINATION is designed to be order optimal in the worst case: its sample complexity is in $O(K \log \frac{1}{\delta})$. However, in practice it is clearly outperformed by SUCCESSIVE ELIMINATION or UGAPEC on both problems (see Figures 5.8(a), 5.8(b)).

5.1.7 Conclusion an future works

We have provided a new definition of privacy for the decentralized algorithms. We have proposed a new problem, the *decentralized exploration problem*, where players sampled from a distribution collaborate to identify a near-optimal arm with a fixed confidence, while ensuring privacy to players and minimizing the communication cost. We have designed and analyzed a generic algorithm for this problem: DECENTRALIZED ELIMINATION uses any best



(a) Problem 1:Uniform distribution of players

(b) Problem 2: 50% of players generates 80% of events

Figure 5.8: Decentralized Median Elimination

arm identification algorithm as an ArmSelection subroutine. Thanks to the generality of the approach, we have extended the analysis of the algorithm to the case where the distributions of rewards are not stationary. Finally, our experiments suggest that *successive elimination* algorithms are better suited for the *decentralized exploration problem* than *explore-then-commit* algorithms.

Future work may focus on user-dependent best arms. When Assumption 2 does not hold, DECENTRALIZED ELIMINATION finds with high probability the best arm of the most frequent players. However, in lot of applications the players can observe a context before choosing an arm. The extension of the proposed approach to *contextual bandits* is not straightforward because to collaborate for building a model, the players have to exchange messages about their favorite arms and their contextual variables, that also contain private information.

5.1.8 Proofs

Theorem 22. *Using any* $\mathrm{ArmSelection}$ *subroutine,* DECENTRALIZED ELIMINATION *is an* (ϵ, η)*-private algorithm, that finds an* ϵ*-optimal arm with a failure probability* $\delta \leq \eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ *and that exchanges at most* $\lfloor \frac{\log \delta}{\log \eta} \rfloor K - 1$ *messages.*

Proof. The proof of Theorem 22 is composed of three parts.

Part 1: (ϵ, η)**-privacy.** Let $E_{l^n} = \{\mathcal{K}^n(l^n) \cap \mathcal{K}_\epsilon = \emptyset\}$ be the event denoting that there is no ϵ-optimal arm in the remaining set of arm $\mathcal{K}^n(l^n)$ at epoch l^n, and $\neg E_{l^n}$ be the event denoting that there is at least an ϵ-optimal arm in the remaining set of arm $\mathcal{K}^n(l^n)$ at epoch l^n.

As DECENTRALIZED EXPLORATION ($\mathcal{A}$) performs an $\mathrm{ArmSelection}$ subroutine on each player, Property 1 ensures that for any player at epoch l^n:

$$\mathbb{P}\left(E_{l^n} | \mathcal{H}_{t^n-1}, \mathcal{A}, \neg E_{l^n-1}\right) \leq \eta \times f(l^n).$$

For the sake of simplicity, in the following we will omit the dependence on $\mathcal{A}$ of probabilities.

The message λ_k^n is sent by player n as soon as the arm k is eliminated from $\mathcal{K}^n(l^n)$ (see lines $17-18$ algorithm 20). Hence, we have:

$$\mathbb{P}\left(E_{l^n}|\mathcal{M}_n, \neg E_{l^n-1}\right) = \mathbb{P}\left(E_{l^n}|\mathcal{H}_{t^n-1}, \neg E_{l^n-1}\right) \leq \eta \times f(l^n),$$

where $t^n(l^n)$ is the time where epoch l^n has begun.

To infer what arm is an ϵ-optimal arm for player n on the basis of $\mathcal{M}_n$ and $\mathcal{A}$, we first consider the favorable case for the adversary, where player n has sent $K-1$ elimination messages which corresponds to epoch $l^n = L$. Using Property 1 of the subroutine used by $\mathcal{A}$ and the set of messages $\mathcal{M}_n$ the adversary can infer that:

$$\begin{aligned}\mathbb{P}\left(\{\mathcal{K}^n(L) \not\subseteq \mathcal{K}_\epsilon\}|\neg E_{L-1}\right) &= \sum_{l^n=1}^{L} \mathbb{P}\left(E_{l^n}|\mathcal{M}_n, \neg E_{l^n-1}\right) \\ &\leq \eta \sum_{l^n=1}^{L} f(l^n) = \eta.\end{aligned}$$

The previous equality holds since if at epoch l^n the event $\{\mathcal{K}^n(l^n) \not\subseteq \mathcal{K}_\epsilon\}$ holds, then it holds also for all following epochs. Then the inequality is obtained by applying Property 1 to each element of the sum. Hence, if $l^n = L$ knowing the set of messages $\mathcal{M}_n$ and Property 1, the adversary cannot infer what arm is an ϵ-optimal arm for player n with a probability higher that $1-\eta$.

Otherwise if $l^n < L$ then $\mathcal{K}^n(L) \subset \mathcal{K}^n(l^n)$, which implies that:

$$\mathbb{P}\left(\{\mathcal{K}^n(l^n) \not\subseteq \mathcal{K}_\epsilon\}|\mathcal{M}_n, \neg E_{l^n-1}\right) \geq \mathbb{P}\left(\{\mathcal{K}^n(L) \not\subseteq \mathcal{K}_\epsilon\}|\mathcal{M}_n, \neg E_{L-1}\right).$$

Hence, if $l^n < L$ the adversary cannot infer what arm is an ϵ-optimal arm with a probability higher that $1-\eta$.

Part 2: Low probability of failure. An arm is eliminated when the events $\{k \notin \mathcal{K}^n(l^n)\}$ occur for $\lfloor \frac{\log \delta}{\log \eta} \rfloor$ independent players. Assumption 8 ($\forall n \in \mathcal{N}$, $P_x(x=n) \neq 0$) and Property 2 ensures that it exists a time $t = \sum_{n=1}^{N} t^n$ such that for $K-1$ arms, there are $\lfloor \frac{\log \delta}{\log \eta} \rfloor$ voting players. Moreover, Property 1 implies that $\forall n \in \mathcal{N}$, $\forall l^n$:

$$\mathbb{P}\left(\{\mathcal{K}^n(l^n) \not\subseteq \mathcal{K}_\epsilon\}|\mathcal{M}_n, \neg E_{l^n}\right) \leq \eta \times f(l^n).$$

Hence, the $\lfloor \frac{\log \delta}{\log \eta} \rfloor$ independent voting players eliminate the ϵ-optimal arm with a probability at most:

$$\mathbb{P}\left(\{\mathcal{K}(l) \not\subseteq \mathcal{K}_\epsilon\}|\mathcal{M}, \neg E_l\right) \leq \left(\eta \times f(l)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor},$$

where $\mathcal{K}(l)$ denotes the shared set of remaining arms at elimination epoch l (see line 7 of Algorithm 20), and

$\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup ... \cup \mathcal{M}_N$.

If the algorithm fails, then the following event occurs : at stopping time, $\exists k \in \mathcal{K}(L), k \notin \mathcal{K}_\epsilon$. Using the union bound, we have:

$$\begin{aligned}\mathbb{P}\left(\{\mathcal{K}(L) \not\subseteq \mathcal{K}_\epsilon\} | \mathcal{M}, \neg E_{L-1}\right) &\leq \sum_{l=1}^{L} \left(\eta \times f(l)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor} \\ &\leq \eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}.\end{aligned}$$

Finally notice that:

$$\eta^{\lceil \frac{\log \delta}{\log \eta} \rceil} \leq \delta = \eta^{\frac{\log \delta}{\log \eta}} \leq \eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}.$$

Part 3: Low communication cost. The index of each arm is sent to other players no more than once per player (see line 17 of the algorithm 20). When $\lfloor \frac{\log \delta}{\log \eta} \rfloor$ messages have been sent for an arm, this arm is eliminated for all players (see lines $4 - 9$ of the algorithm 20).

Thus $\lfloor \frac{\log \delta}{\log \eta} \rfloor (K-1)$ messages are sent to eliminate the suboptimal arms. Then, at most $\lfloor \frac{\log \delta}{\log \eta} \rfloor - 1$ messages have been sent for the remaining arm. Thus, the number of sent messages is at most $\lfloor \frac{\log \delta}{\log \eta} \rfloor K - 1$.

□

Theorem 23. *Using any* $\mathrm{ArmSelection}(\epsilon, \eta, \mathcal{K})$ *subroutine, with a probability higher than* $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ DECENTRALIZED ELIMINATION *stops after at most:*

$$T_{P_{x,\mathbf{y}}} = \frac{T_{P_\mathbf{y}}}{p^*} + \frac{1}{(p^*)^2}\sqrt{\frac{p^* T_{P_\mathbf{y}}}{2} \log \frac{1}{\delta}} + \frac{1}{2(p^*)^2} \log \frac{1}{\delta} \textit{samples in } P_{x,\mathbf{y}},$$

where $I_a(b,c)$ *denotes the incomplete beta function evaluated at* a *with parameters* b, c*.*

Proof. Let $T_{P_{x,\mathbf{y}}}$ be the number of samples in $P_{x,\mathbf{y}}$ when DECENTRALIZED ELIMINATION stops with high probability, and $T_{P_\mathbf{y}}$ be the time step where the $\mathrm{ArmSelection}$ subroutine stops with high probability. Let T_n be the number of samples of player n at time T. T_n is a binomial law of parameters $T, P_x(x=n)$. Hence, we have:

$$\mathbb{E}_{P_x}[T_n] = P_x(x=n)T.$$

Let $\mathcal{B}_{\delta,\eta}$ be the set of players that have the $\lfloor \frac{\log \delta}{\log \eta} \rfloor$ highest T_n. The algorithm does not stop, if the following event occurs: $E_1 = \{\exists n \in \mathcal{B}_{\delta,\eta}, T_n < T_{P_\mathbf{y}}\}$.

Applying Hoeffding inequality, we have:

$$\mathbb{P}\left(T_n - P_x(x=n)T \leq -\epsilon\right) \leq \exp(-\frac{2\epsilon^2}{T})$$

When $\neg E_1$ occurs, $\forall n \in \mathcal{B}_{\delta,\eta}$ we have with a probability at most δ:

$$T_{P_{\mathbf{y}}} - P_x(x=n)T \leq -\sqrt{\frac{T}{2}\log\frac{1}{\delta}}.$$

$$\Leftrightarrow -P_x(x=n)T + \sqrt{\frac{T}{2}\log\frac{1}{\delta}} + T_{P_{\mathbf{y}}} \leq 0,$$

$$\Leftrightarrow \sqrt{T} \geq \frac{1}{2P_x(x=n)}\left(\sqrt{\frac{1}{2}\log\frac{1}{\delta}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta} + 4P_x(x=n)T_{P_{\mathbf{y}}}}\right),$$

$$\Leftrightarrow T \geq \frac{1}{4(P_x(x=n))^2}\left(\sqrt{\frac{1}{2}\log\frac{1}{\delta}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta} + 4P_x(x=n)T_{P_{\mathbf{y}}}}\right)^2$$

Then, when $\neg E_1$ occurs we have with a probability at most δ:

$$T \geq \frac{1}{4(p_{\delta,\eta})^2}\left(\sqrt{\frac{1}{2}\log\frac{1}{\delta}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta} + 4p_{\delta,\eta}T_{P_{\mathbf{y}}}}\right)^2,$$

where $p_{\delta,\eta} = \min_{n\in\mathcal{B}_{\delta,\eta}} P_x(x=n)$.

Hence, if E_1 does not occur, then we have with a probability at least $1-\delta$:

$$T < \frac{1}{4(p_{\delta,\eta})^2}\left(\sqrt{\frac{1}{2}\log\frac{1}{\delta}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta} + 4p_{\delta,\eta}T_{P_{\mathbf{y}}}}\right)^2,$$

$$\Leftrightarrow T < \frac{1}{4(p_{\delta,\eta})^2}\log\frac{1}{\delta} + \frac{T_{P_{\mathbf{y}}}}{p_{\delta,\eta}} + \frac{1}{2(p_{\delta,\eta})^2}\sqrt{\frac{1}{2}\log\frac{1}{\delta}}\sqrt{\frac{1}{2}\log\frac{1}{\delta} + 4p_{\delta,\eta}T_{P_{\mathbf{y}}}},$$

$$\Rightarrow T < \frac{1}{4(p_{\delta,\eta})^2}\log\frac{1}{\delta} + \frac{T_{P_{\mathbf{y}}}}{p_{\delta,\eta}} + \frac{1}{4(p_{\delta,\eta})^2}\log\frac{1}{\delta} + \frac{1}{(p_{\delta,\eta})^2}\sqrt{\frac{p_{\delta,\eta}T_{P_{\mathbf{y}}}}{2}\log\frac{1}{\delta}},$$

$$\Rightarrow T < T_{P_{x,\mathbf{y}}} = \frac{T_{P_{\mathbf{y}}}}{p_{\delta,\eta}} + \frac{1}{(p_{\delta,\eta})^2}\sqrt{\frac{p_{\delta,\eta}T_{P_{\mathbf{y}}}}{2}\log\frac{1}{\delta}} + \frac{1}{2(p_{\delta,\eta})^2}\log\frac{1}{\delta}.$$

Let $\mathcal{N}_M$ bet the set of the $M = \lfloor\frac{\log\delta}{\log\eta}\rfloor$ most likely players. Let $n^* = \arg\min_{n\in\mathcal{N}_M} P_x(x=n)$, and $p^* = \min_{n\in\mathcal{N}_M} P_x(x=n)$.

Now, we consider the following event: $E_2 = \{n^* \notin \mathcal{B}_{\delta,\eta}\}$. By the definition of $\mathcal{B}_{\delta,\eta}$, the event E_2 is equivalent to the event $\{T_{n^*} < T_{P_{\mathbf{y}}}\}$. Then, we have:

$$\mathbb{P}\left(T_{n^*} < T_{P_{\mathbf{y}}}\right) = I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_{\mathbf{y}}}, 1 + T_{P_{\mathbf{y}}}\right),$$

where $I_a(b,c)$ denotes the incomplete beta function evaluated at a with parameters b, c.

Finally, with a probability at least $(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right))^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$, we have $p_{\delta,\eta} = p^*$.

□

Corollary 6 . *With a probability higher than* $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ DECENTRALIZED MEDIAN ELIMINATION *stops after:*

$$\mathcal{O}\left(\left(\frac{K}{\lfloor \frac{\log \delta}{\log \eta} \rfloor \epsilon^2} + \frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta}\right) \textit{ samples in } P_{x,\mathbf{y}}.$$

Proof. We have:

$$\begin{aligned}
&\eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor} \leq \delta = \eta^{\frac{\log \delta}{\log \eta}} \leq \eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor} \\
&\Rightarrow \frac{1}{\delta} \geq \frac{1}{\eta^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}} \\
&\Leftrightarrow \log\frac{1}{\eta} \leq \frac{1}{\lfloor \frac{\log \delta}{\log \eta} \rfloor}\log\frac{1}{\delta}
\end{aligned}$$

MEDIAN ELIMINATION algorithm [Even-Dar et al., 2002] finds an ϵ-optimal arm with a probability at least $1-\eta$, and needs at most:

$$T_{P_\mathbf{y}} = \mathcal{O}\left(\frac{K}{\epsilon^2}\log\frac{1}{\eta}\right) \leq \mathcal{O}\left(\frac{K}{\lfloor \frac{\log \delta}{\log \eta} \rfloor \epsilon^2}\log\frac{1}{\delta}\right) \text{ samples in } P_\mathbf{y}.$$

Then the use of Theorem 23 finishes the proof.

□

Corollary 7. *With a probability higher than* $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)\right)^{\lfloor \frac{\log \delta}{\log \eta} \rfloor}$ DECENTRALIZED SUCCESSIVE ELIMINATION *stops after:*

$$\mathcal{O}\left(\left(\frac{K}{\lfloor \frac{\log \delta}{\log \eta} \rfloor \epsilon^2} + \frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta} + \frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}\right) \textit{ samples in } P_{x,\mathbf{y}},$$

samples in $P_{x,\mathbf{y}}$.

Proof. SUCCESSIVE ELIMINATION algorithm [Even-Dar et al., 2002] finds an ϵ-optimal arm with a probability at least $1-\eta$, and needs at most:

$$T_{P_\mathbf{y}} = \mathcal{O}\left(\frac{K}{\epsilon^2}\log\frac{K}{\eta}\right) \leq \mathcal{O}\left(\frac{K}{\epsilon^2}\left(\log K + \frac{1}{\lfloor \frac{\log \delta}{\log \eta} \rfloor}\log\frac{1}{\delta}\right)\right)$$

samples in $P_{x,\mathbf{y}}$. Then using Theorem 23, we have:

$$T_{P_{x,\mathbf{y}}} = \frac{T_{P_\mathbf{y}}}{p^*} + \frac{1}{(p^*)^2}\sqrt{\frac{p^* T_{P_\mathbf{y}}}{2}\log\frac{1}{\delta}} + \frac{1}{2(p^*)^2}\log\frac{1}{\delta}, \tag{5.1}$$

$$\leq \mathcal{O}\left(\frac{K}{p^*\epsilon^2}\left(\log K + \frac{1}{\lfloor\frac{\log\delta}{\log\eta}\rfloor}\log\frac{1}{\delta}\right) + \frac{1}{(p^*)^2}\sqrt{\frac{Kp^*}{\epsilon^2}\left(\log K + \frac{1}{\lfloor\frac{\log\delta}{\log\eta}\rfloor}\log\frac{1}{\delta}\right)\log\frac{1}{\delta}} + \frac{1}{(p^*)^2}\log\frac{1}{\delta}\right), \tag{5.2}$$

$$\leq \mathcal{O}\left(\frac{K}{p^*\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2}\log\frac{1}{\delta} + \frac{1}{(p^*)^2}\log\frac{1}{\delta} + \frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}\right), \tag{5.3}$$

$$\leq \mathcal{O}\left(\left(\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2} + \frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta} + \frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}\right). \tag{5.4}$$

□

Theorem 24. *For $K \geq 2$, $\delta \in (0, 0.5]$, $\epsilon \geq \delta/K$, for the sequences of rewards where Assumption 4 holds, DSER3 is an (ϵ, η)-private algorithm, that exchanges at most $\lfloor\frac{\log\delta}{\log\eta}\rfloor K - 1$ messages, that finds an ϵ-optimal arm with a probability at least $(1-\delta)\left(1 - I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)\right)^{\lfloor\frac{\log\delta}{\log\eta}\rfloor}$, and that stops after:*

$$\mathcal{O}\left(\left(\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2} + \frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta} + \frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}\right)$$

samples in $P_{x,\mathbf{y}}$.

Proof. Theorem 24 is a straightforward application of Theorem 23, where $T_{P_\mathbf{y}}$ is stated in Theorem 1 [Allesiardo et al., 2017]. □

Theorem 25. *For $K \geq 2$, $\epsilon \geq \frac{\eta}{K}$, $\varphi \in (0, 1]$, $\epsilon \geq \delta/K$ for any sequences of rewards that can be splitted into sequences where Assumption 4 holds,* DSER4 *is an (ϵ, η)-private algorithm, that exchanges on average at most $\varphi T(\lfloor\frac{\log\delta}{\log\eta}\rfloor K - 1)$ messages, and that plays, with an expected probability at most $\delta + \varphi T I_{1-p^*}\left(T_{P_{x,\mathbf{y}}} - T_{P_\mathbf{y}}, 1 + T_{P_\mathbf{y}}\right)^{\lfloor\frac{\log\delta}{\log\eta}\rfloor}$, a suboptimal arm on average no more than:*

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\sqrt{\frac{\left(\frac{K}{p^*}\log K + \frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor}\right)\frac{S}{p^*}\log\frac{1}{\delta}}{\delta}} + \frac{1}{(p^*)^2}\log\frac{1}{\delta}\right)$$

times, where S is the number of switches of best arms, φ is the probability of reset in SER4*, T is the time horizon, and the expected values are taken with respect to the randomization of resets.*

Proof. The upper bound of the expected number of times a suboptimal arm is played by SER4, is stated in Corollary 2 [Allesiardo et al., 2017]. Then this upper bound is used in Theorem 23 to state the upper bound of the expected

number of times a suboptimal arm is played using DSER4:

$$T_{P_x,\mathbf{y}} \leq \mathcal{O}\left(\frac{\varphi}{\delta}\left(\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2}+\frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta}+\frac{\varphi}{\delta}\frac{K}{p^*\epsilon^2}\log K\sqrt{\log\frac{1}{\delta}}+\frac{S}{\varphi}\right), \tag{5.5}$$

$$\leq \mathcal{O}\left(\frac{\varphi}{\delta}\left(\frac{K}{p^*\epsilon^2}\log K+\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2}+\frac{1}{p^*}\right)\frac{1}{p^*}\log\frac{1}{\delta}+\frac{S}{\varphi}\right), \tag{5.6}$$

$$\leq \mathcal{O}\left(\frac{\varphi}{\delta}\left(\frac{K}{p^*\epsilon^2}\log K+\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor\epsilon^2}\right)\frac{1}{p^*}\log\frac{1}{\delta}+\frac{S}{\varphi}+\frac{1}{(p^*)^2}\log\frac{1}{\delta}\right). \tag{5.7}$$

Then setting:

$$\varphi=\sqrt{\frac{S\delta}{\left(\frac{K}{p^*}\log K+\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor}\right)\frac{1}{p^*}\log\frac{1}{\delta}}},$$

we obtain:

$$T_{P_x,\mathbf{y}} \leq \mathcal{O}\left(\frac{1}{\epsilon^2}\sqrt{\frac{\left(\frac{K}{p^*}\log K+\frac{K}{\lfloor\frac{\log\delta}{\log\eta}\rfloor}\right)\frac{S}{p^*}\log\frac{1}{\delta}}{\delta}}+\frac{1}{(p^*)^2}\log\frac{1}{\delta}\right). \tag{5.8}$$

The expected number of resets is φT. Theorem 23 provides the success probability of each run of DECENTRALIZED ELIMINATION, which states the expected failure probability of DSER4. Then using Theorem 22 the expected upper bound of the number of exchanged messages is stated. □

5.2 Collaborative Exploration and Exploitation in massively Multi-Player Bandits

Abstract. In this paper, we propose an approach to optimize the performance of Internet of Things networks. We formulate the optimization problem as a massive multi-player multi-armed bandit, where the devices are the players and the radio channels are the arms, with collisions possibly preventing message reception. Most previous works assume the number of channels to be larger than the number of players. We go beyond this assumption which radically changes the nature of the optimization problem that becomes intractable. Moreover, sharing the channels with much more players than the number of channels raises the problem of fairness between players. Assuming access to a model of the environment, we introduce two novel policies: Decreasing-Order-Reward-Greedy (DORG) focuses on the number of successful communications and is optimal under some conditions, while Decreasing-Order-Fair-Greedy (DOFG) also guarantees some measure of fairness between players. To estimate the environment's model, we implement a finite-sample federated exploration bandit with limited information exchanges between players. We show that its sample complexity is near optimal and that the pseudo-regret of the estimated DORG under certain

conditions is optimal with respect to the time horizon T, and that DOFG still guarantees fairness even with finite data. Finally, we provide experimental evidence that our algorithms outperform state-of-the-art competitors in terms of fairness and successful communications.

5.2.1 Targeted use case: minimizing energy in IoT network

IoT background

The Internet of Things (IoT) has gained a great attention in the last decade. The world has been witnessing such a massive growth in the node deployment that the IoT survey reported on the Forbes website [Newman, 2019] forecasts more than 75 billion connected IoT devices by 2025. Massive IoT applications require energy-efficient and low-complexity nodes. To support such requirements, Low Power Wide Area Networks (LPWANs), that provide large coverage areas, low transmission data rates with small packet data sizes, low device complexity and long battery life have evolved [Chaudhari et al., 2020]. LPWANs include several technologies operating in the unlicensed industrial, scientific and medical (ISM) frequency band (868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia), and the Long Range Wide Area Network (LoRaWAN) developed by the LoRa alliance [lor, 2021] is one of these massively deployed technologies for low power and long distance transmissions. However, with the great increase of IoT deployment, a major problem of systems' coexistence arises. Inside the unlicensed band, the different systems are not separated in the frequency domain but overlapping in the sense that they may use the same frequency resource at any time, causing interference and hence transmission failures.

Underlying assumptions

1. *The number of devices could be greater than the number of channels:* Indeed, in Internet of Things (IoT) networks, a large number of devices are connected to the Internet through wireless gateways, and hence the number of devices cannot be lesser than the number of channels.

2. *Each successful uplink transmission is followed by a downlink acknowledgement:* The communication protocols used in IoT allow to assign a binary outcome for each transmission (success or not) since each uplink transmission is followed by time windows during which the device listens to the gateway to receive the acknowledgement of the uplink transmission.

3. *Sensing information is not possible:* The devices cannot distinguish internal and external collisions, rather they can only observe the success or failure of their transmissions. This is known to be a difficult case for multi-player multi-armed bandits, however it is realistic for IoT networks, where sensing information is too costly in terms of energy consumption.

4. *Downlink transmissions do not fail:* We do not consider that collisions could occur when the gateway sends acknowledgements. Indeed, these downlink collisions require that at least two acknowledgements are sent from the gateway at the same time to different devices located at the same place, which cannot happen with a unique gateway using a protocol slotted in time, and which would be unlikely in a real Internet of Things (IoT) network, where a finite number of gateways is positioned to cover the maximum area.

5. *Each device has a probability of sending a packet at each time step:* The frequency of sending packets through the gateway depends on the application (healthcare, security, smart cities, marketing, home automation...). Moreover, for several real-time applications, the device has to send a packet when an unknown and uncontrolled event occurs. For instance, a user's device can interact with its environment in real-time, to get a green light when the user faces a crossroad, an ad when the user is in front of a shop, a ticket when getting on the bus, and more critical applications such as healthcare ones. Such packets has to be sent and processed as soon as possible, and therefore the authors in [Valach and Macko, 2022] suggest a modification in the LoRA@FIIT [Perešíni and Krajčovič, 2017] link-layer protocol, so such emergency packets are given the priority to be retransmitted in case of failure over other types of normal packets in order to guarantee QoS in LoRa networks.

6. *Devices are Socratic*[3]*:* Considering that the probability of sending a packet depends mainly on the type of devices, we assume that each device knows its own probability of sending a packet.

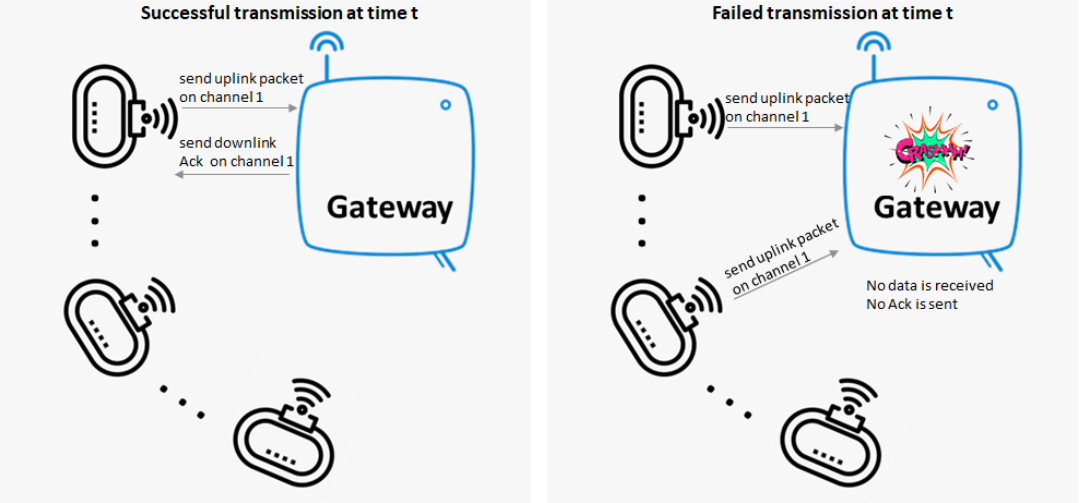
7. *Known number of devices:* We assume that the number of devices is known by the gateway, which is realistic in IoT protocols (the gateway can keep track of all the devices it has received packets from), and that the gateway sends this information to each device at the beginning of the game.

8. *Devices can share information by including their messages in the payload of the packet they need to send:* We allow the devices to share information by sending messages to other devices through the gateway using the IoT protocol. As in IoT networks the payload of each packet can contain up to 255 bytes [Augustin et al., 2016, Sornin and Yegin, 2018], we assume that in the same packet 8 bytes of the payload can be used to send a message to other players. We hereby distinguish between the two terms: a *packet* that corresponds to the regular transmissions of a device, and a *message* that corresponds to the information shared between the players.

Minimizing energy

IoT devices often run using battery, and hence minimizing energy consumption increases the lifetime of the devices. There are two ways for decreasing energy consumption of devices:

[3]from the ancient Greek aphorism "know thyself" attributed to Socrates.



(a) No collision occurs

(b) A collision occurs

Figure 5.9: A collision occurs when two devices send an uplink packet at the same time on the same channel: due to local interference, the gateway does not received the packets and hence does not send acknowledges.

1. **Minimizing collision**: when a transmission fails, the lost packets are retransmited by the collided devices, hence increasing the number of successful transmissions obviously decreases energy consumption.

2. **Choosing energy efficient channels**: the energy consumption of each channel is different.

Minimizing the energy consumption while maintaining a high successful are two incompatible objectives. For instance in LoRa network as the SF (Spreading Factor) and TP (Transmission Power) increase, the successful transmission rate increases, and the energy consumption increases.

In the following, we first focus on the first objective: minimizing collisions. Then, we discuss, adapt and test the proposed approach to LoRa network for minimizing collisions and minimizing energy consumption.

5.2.2 Introduction

We consider a large number N of devices communicating through a unique gateway on a limited number K of orthogonal (independent) channels ($N \geq K$). The devices use an acknowledgement protocol slotted in time, where an acknowledgement is sent by the gateway to the transmitting device after each successful transmission. A transmission fails if and only if a *collision* occurs. If this is the case, the packets of all colliding devices are lost and no acknowledgement is sent. There exist two types of collision. An *internal collision* occurs when two or more devices send packets to the gateway at the same time slot through the same channel. *External collisions* may also occur with unknown and uncontrolled devices. Therefore, even if only one device sends a packet on one channel at a given time slot, the packet may not be received by the gateway. Because of their nature, *external collisions* make the probabilities of successful transmission (and hence the channels' qualities) uncontrollably differ over channels.

Another important feature of the studied problem is that, in the general case, the gateway cannot know that packets have been sent by some devices if a collision occurs. As a consequence, the estimation of the channel quality can only be done at the device side in a decentralized way. In such conditions and in order to maximize the number of successful transmissions of the IoT network while consuming as little energy as possible, we model our problem as a massive multi-player multi-armed Bandit.

Related Work. The *decentralized multi-player multi-armed bandits* have been studied for opportunistic spectrum access in [Liu and Zhao, 2010, Anandkumar et al., 2010, Avner and Mannor, 2014, Nayyar et al., 2018]. In opportunistic spectrum access, primary users have a strict priority over secondary users, which are allowed *sensing* a channel before sending a packet in order to check that it is free. The objective of those works is to avoid collisions between concurrent secondary users, that share the same channels, while choosing the best channels, i.e. with the highest probabilities to be free of primary users. This line of work makes the assumption that there are less players than channels, that the collisions with other players are observed, and uses orthogonalization techniques to avoid collisions. In [Rosenski et al., 2016], the authors propose to use collisions to estimate in a first phase the number of players and the value of arms and then a Musical Chair approach to allocate each player on a different N-best arm. In [Hanawal and Darak, 2018], the authors improve this approach by reducing the first phase to the estimation of the value of arms and then use a trekking approach to allocate each player on a different N-best arm without the knowledge of the number of players. In [Boursier and Perchet, 2019b], the authors propose a communication protocol based on controlled collisions that achieves almost the same performance as a centralized algorithm. In [Wang et al., 2020], the authors improve this result by electing a leader that explores the arms and allocates other players on different estimated N-best arms. The leader communicates to other players the list of estimated N-best arms when it changes using the same communication protocol as in [Boursier and Perchet, 2019b]. This algorithm is asymptotically optimal. An interesting extension of the problem setting was proposed in [Boursier et al., 2020] for handling the case where the mean rewards of arms are not the same for each player. However, this thread of research makes the assumption that *sensing* information is available and the number of players is small ($N \leq K$), which are both unrealistic assumptions for IoT networks.

In [Boursier and Perchet, 2019b], the authors also propose an adaptation of their algorithm to the case where *sensing* is not allowed, that preserves the logarithmic behavior with respect to the time horizon. This approach has been improved in [Shi et al., 2020] for the case where sensing is not allowed thanks to the use of *Z-channel coding*, quantization of transmitted statistics and a tree structured communication, where a leader gathers the statistics and then decides for all players the best set of arms. In [Lugosi and Mehrabian, 2018], the authors define the multi-player stochastic multi-armed bandit as an anti-coordination game, where the goal is to quickly reach an approximate Nash equilibrium. Finally, in [Bubeck et al., 2019] the difficult case of non-stochastic multi-player multi-armed bandits is addressed. In all these works, the number of players is still assumed to be below the number of channels.

Motivated by IoT networks, in [Bonnefoi et al., 2018, Besson and Kaufmann, 2018] the authors propose a new

problem setting where *sensing* is not allowed, the number of players is larger than the number of channels, and the players asynchronously play: each player has the same probability to send a packet at each time slot. The authors show experimentally that *selfish UCB*, which consists in each player independently playing *UCB* [Auer et al., 2002a], works surprisingly well. This experimental result has been confirmed in the case of LoRa networks using stochastic and non-stochastic multi-armed bandits [Kerkouche et al., 2018] or in the case of IEEE 802.15.4 time-slotted channel hopping protocol [Dakdouk et al., 2018]. Despite its good experimental performance, this algorithm has no theoretical guarantees, and it has been shown that *selfish UCB* can fail badly on some cases [Besson and Kaufmann, 2018]. With a similar problem setting but with different probabilities to send packets the authors in [Dakdouk et al., 2020] propose a cooperative algorithm that aims to find a set of optimal arms while minimizing the number of plays. However that work does not optimize the number of optimal arms to find, and the exploitation policy followed by the players is uniform, which is clearly sub-optimal. These limitations are resolved with our proposed algorithms.

Contributions and paper organization. In this paper, we study the extension of the problem proposed in [Bonnefoi et al., 2018], where each player has a different probability to send a packet at a each time slot [Dakdouk et al., 2020]. We propose an *explore-then-exploit* approach, where a decentralized exploration algorithm outputs an estimation of the parameters, and then a target policy, which can be computed by the gateway, is used during the exploitation phase.

The remainder of this paper is organized as follows. In section 5.2.3, we formalize the objective of optimizing the successful transmissions. We show in Theorem 26, that there exists a deterministic policy (an assignment of players over arms) that is optimal. Then we propose two deterministic policies: `DORG` (decreasing-order-reward-greedy) that aims to optimize the number of successful transmissions, while `DOFG` (decreasing-order-fair-greedy) guarantees some fairness between players in terms of successful transmission rate. In Theorem 27, we show that `DORG` is optimal at least in the setting proposed in [Bonnefoi et al., 2018] (when $\forall n,\, p_n = p$), while Theorem 28 states fairness guarantees for `DOFG`. Since the packet loss can only be observed by players, in section 5.2.4 we propose a decentralized collaborative exploration algorithm that outputs an unbiased estimate of the mean rewards of arms, i.e. the channels' qualities, with classic concentration properties. Theorem 30 states an upper bound on the number of time steps needed to output a controlled approximation of the arms that is near optimal in comparison to the lower bound of K biased coin estimations in $\Omega\left(K/\epsilon^2 \log 1/\delta\right)$ [Anthony and Bartlett, 1999]. Furthermore, Theorem 29 states an upper bound of the communication cost in $O\left(NK \log(NK+N)/\delta\right)$. Then, in the setting proposed in [Bonnefoi et al., 2018], Theorem 31 states that when using `DORG`, the proposed algorithm benefits from a pseudo-regret upper bound that is optimal in T. We also state a pseudo-regret lower bound in $\Omega\left(T^{2/3}\frac{\log T}{N}\right)$, which holds for any *explore-then-exploit* algorithm, and reveals the difficulty of the studied problem in comparison to the multi-armed bandit and multi-player bandit problems. Finally, Theorem 33 states fairness guarantees of our *explore-then-exploit* algorithm with `DOFG`. In section 5.2.7, we compare our approach with the state-of-the-art methods on a large set of

synthetic problems. We show that when using DORG, the proposed algorithm outperforms the baselines in terms of successful communication rate, and when using DOFG it outperforms them in terms of fairness between players.

5.2.3 Collaborative Exploitation in massively multi-player bandits

Problem Formulation

Let $[N]$ be a set of N players, such that at each time slot t each player $n \in [N]$ has a constant probability p_n to send a packet, such that $1 > p > p_n > 0$, where p is the duty cycle that is imposed to the IoT network in order to share the free bandwidth with other users. Without loss of generality, in the following we assume that: $p_1 \geq ... \geq p_N$. At each time slot t, the set $\mathcal{N}_t$ of players sending packets is selected by N independent Bernoulli samples: $\mathcal{N}_t := \{n \in [N] \text{ such that } a_n = 1, \text{ with } a_n \sim \mathcal{B}(p_n)\}$. Let $[K]$ be the set of K arms. The transmission of a packet is successful if it does not collide with other packets. The random variable representing an external collision on arm k is denoted by $E^k \sim \mathcal{B}(\theta^k)$ (equals 0 if collision, 1 otherwise). Similarly, internal collisions between the controlled players are represented by the random variables $(I^k)_{k\in[K]}$ (equals 0 if collision, 1 otherwise) and depend on the implemented policy. After playing arm k, player n observes the binary outcome $Y^{k_n} = E^{k_n} I^{k_n}$, i.e., knows whether a collision occurred or not (through an acknowledgement) but cannot distinguish external and internal collisions.

We will call a *policy* a (possibly randomized) way for players to select the channel to use for their next transmission. Formally, a policy π will be a vector of probability distributions over the set of arms: $\pi = (\pi_1, ..., \pi_N)$, with $\pi_n = (\pi_n^1, ..., \pi_n^K)$, where π_n^k denotes the probability that player n chooses arm k for sending a packet. We denote by $\mu_{n,\boldsymbol{\theta}}^k(\pi)$ the expected reward in model $\boldsymbol{\theta} = \{\theta^1, ..., \theta^K\}$ of playing arm k while the other players follow policy π. It is the probability that no external collision occurs times the probability that no internal collision occurs:

$$\mu_{n,\boldsymbol{\theta}}^k(\pi) = \theta^k \prod_{n'=1, n'\neq n}^{N} (1 - p_{n'} \pi_{n'}^k). \tag{5.9}$$

Equation (5.9) shows the difficulty of the studied problem: the mean reward of an arm for a given player depends on the probabilities of the other players to send a packet and on the policies they follow. The aggregated average reward in model $\boldsymbol{\theta} = \{\theta^1, .., \theta^K\}$ per time slot over all players $\mu_{\boldsymbol{\theta}}(\pi)$ is:

$$\mu_{\boldsymbol{\theta}}(\pi) = \sum_{k=1}^{K} \theta^k \sum_{n=1}^{N} p_n . \pi_n^k \prod_{n'\in[N]\setminus\{n\}} (1 - p_{n'} \pi_{n'}^k). \tag{5.10}$$

This performance metric corresponds to the expected number of successful transmissions per time slot. The

optimization problem in Equation (5.10) with respect to π has a solution, since the objective function is continuous and the set of decision variables is compact. But the problem itself is not convex with respect to π_n^k, hence classical convex optimization methods cannot be applied.

Another approach is to consider a particular subset of policies: the subset of deterministic policies is obtained when $\forall(k,n) \in [K] \times [N]$, $\pi_n^k \in \{0,1\}$. Let k_n be the arm assigned to player n. The expected reward per time slot in model $\boldsymbol{\theta} = \{\theta^1, ..., \theta^K\}$ of a deterministic policy π can then be written as:

$$\begin{aligned} \mu_{\boldsymbol{\theta}}(\pi) &= \sum_{n=1}^{N} p_n \theta^{k_n} \prod_{n' \neq n, \text{ s.t. } k_{n'}=k_n} (1-p_{n'}) \\ &= \sum_{k=1}^{K} \theta^k \underbrace{\prod_{n \in [N], \text{ s.t. } k_n=k} (1-p_n)}_{z^k} \underbrace{\sum_{n \in [N], \text{ s.t. } k_n=k} \frac{p_n}{1-p_n}}_{\ell^k}, \end{aligned} \tag{5.11}$$

where z^k is the probability that all players assigned to arm k do not send packets, and ℓ^k is the sum of the activation odds for all players assigned to arm k.

Theorem 26. *There exists a policy maximizing the overall network utility (equation* (5.10)*) that is deterministic.*

Theorem 26 states that at least one solution is a deterministic policy, which justifies to consider only the subset of deterministic policies. However, as a deterministic policy is an assignment of players over arms, there are N^K deterministic policies. This means that when N and K are not small, finding the optimal policy is hopeless, and this even if the model $\boldsymbol{\theta}$ is known in advance, which is not the case.

Discussion

In face of these impossibility results for both stochastic and deterministic policies, for handling massively multi-player multi-armed bandits, we aim to find reasonably good deterministic target policies in the next section. Then, in section 5.2.4, we will propose an exploration algorithm that finds an unbiased and controlled approximation of the model $\boldsymbol{\theta}$ for computing the target policy. This *explore-then-exploit* approach allows to compute a controlled approximation of a target policy, even in the case where N and K are not small. Moreover if N and K are small, a controlled approximation of the optimal policy can be obtained. The alternative approach consisting of using an optimal multi-armed bandit algorithm that consider each deterministic policy as an arm will lead to a regret lower bound in $\Omega\left(\sqrt{N^K T}\right)$ [Bubeck and Cesa-Bianchi, 2012]. Notice that even when N and K are small (for instance in the order of 10) the dominant term of the regret lower bound is not T, but N^K.

Algorithm 21 Reward Greedy
(DORG if players are sorted in p_n decreasing order)

Inputs: $[K]$, $[N]$, $\{\theta^k\}_{k\in[K]}$, $\{p_n\}_{n\in[N]}$
Output: π
Init: per-arm inactivity probabilities: $z^k = 1$.
Init: per-arm activation odds sums: $\ell^k = 0$.

1: **for** $n = 1$ to N **do**
2: Set $k_n \in \arg\max_{k\in[K]} \theta^k z^k (1 - \ell^k)$.
3: Update $z^{k_n} \leftarrow z^{k_n}(1 - p_n)$.
4: Update $\ell^{k_n} \leftarrow \ell^{k_n} + \frac{p_n}{1-p_n}$.
5: Set $\pi_n^{k_n} = 1$, and $\forall k \neq k_n, \pi_n^k = 0$.
6: **end for**

Algorithm 22 Fairness Greedy
(DOFG if players are sorted in p_n decreasing order)

Inputs: $[K]$, $[N]$, $\{\theta^k\}_{k\in[K]}$, $\{p_n\}_{n\in[N]}$
Output: π
Init: per-arm inactivity probabilities: $z^k = 1$.

1: **for** $n = 1$ to N **do**
2: Let $k_n \in \arg\max_{k\in[K]} \theta^k z^k$
3: Update $z^{k_n} \leftarrow z^{k_n}(1 - p_n)$
4: Set $\pi_n^{k_n} = 1$, and $\forall k \neq k_n, \pi_n^k = 0$.
5: **end for**

Reward greedy algorithm

In this section, we propose a greedy algorithm that aims to maximize the network utility (equation (5.11)).

Lemma 9. *For a deterministic policy π, let $\mu_{\boldsymbol{\theta}}(\pi[n])$ denote the expected reward when only players $1, ..., n$ are playing (all players $n' > n$ are deactivated). Then we have the recursive expression:*

$$\mu_{\boldsymbol{\theta}}(\pi[n]) = \mu_{\boldsymbol{\theta}}(\pi[n-1]) + p_n \theta^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) z_{[n-1]}^{k_n},$$

where $z_{[n]}^k$ is the probability that arm k is not used by any of the first n players, and $\ell_{[n]}^k$ is the sum of activation odds of the n first players for arm k.

Lemma 9 reveals a recurrence relation over n of the expected total reward. Under the assumption that the problem parameters are known, Lemma 9 paves the way to the definition of DORG, decreasing-order-reward-greedy (Algorithm 21), a recursive algorithm that assigns player n to arm k_n such that the right-hand term of the recursive equation in Lemma 9 is maximized. The result is highly dependent on the order in which the players are added to the pool, but the following theorem shows Algorithm 21 can lead to an actual optimum.

Theorem 27. *If $\sum_{n\in[N]} \frac{p_n}{1-p_n} \leq K + 1$, then there exists an ordering over players $\sigma^* : [N] \to [N]$ such that Algorithm 21 returns an optimal policy.*

Remark 21. *When $\forall n, p_n = p$ ([Bonnefoi et al., 2018]) Theorem 27 states that DORG returns the optimal policy. The*

precondition of Theorem 27 clearly holds in IoT networks, where the duty cycle p is commonly set to less than 0.01.

Fairness greedy algorithm

Theorem 26 states that the resource assignment of an optimal deterministic policy is a Pareto optimum: as the network utility is maximum, if a user increases its own utility (equation (5.9)) another user has necessarily to decrease its utility (due to equation (5.10)). Notice that a Pareto optimum does not provide any guarantee about the *fairness* of the resource allocation among players. In this section, we design a policy to ensure fairness among players, for which we will use the definition below.

Definition 28 (α-fairness)**.** *A policy π is said to be α-fair if* $\frac{\min_{n\in[N]} \sum_{k=1}^{K} \mu_{n,\boldsymbol{\theta}}^{k}(\pi)}{\max_{n\in[N]} \sum_{k=1}^{K} \mu_{n,\boldsymbol{\theta}}^{k}(\pi)} \geq \alpha.$

Building a fair policy can be done by balancing the load with respect to the mean rewards of arms. The fair greedy algorithm (see Algorithm 22) assigns sequentially each player to the arm that maximizes the reward of the arm times the probability of no internal collision. The player scheduling also plays an important role and we prove a lower bound on the fairness of Algorithm 22, when players are sorted in decreasing order of p_n. In that case we coin this algorithm `DOFG`, which stands for decreasing-order-fair-greedy.

Theorem 28. *`DOFG` generates α-fair policies, with $\alpha \geq 1 - \max_{n\in[N]} p_n$.*

Theorem 28 implies that when the probability of sending packets of the most frequent player is not high, which is the case in IoT networks, `DOFG` is a fair policy.

Preliminary experiments

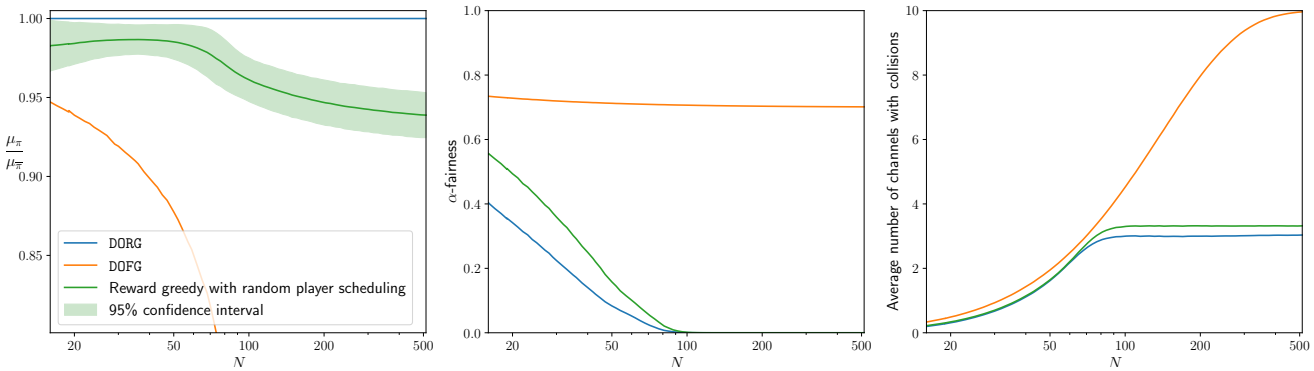


Figure 5.10: With number of arm $K = 10$ fixed, and for N values (ranging from 16 to 512 on a log scale), the algorithms have been compared in terms of expected reward ratio with `DORG` (left), $\bar{\pi}$ denotes the policy to be compared with `DORG`, α-fairness (center), and number of channels with internal collision (right).

In this section, we perform the following experiment: the problem parameters are sampled as follows: $\forall n \in [N], p_n \sim \mathcal{U}(0, 0.3)$[4] and $\forall k \in [K], \theta^k \sim \mathcal{U}(0, 1)$. Figure 5.10 compares the performance of `DORG`, `DOFG`, and Reward

[4]Such high values for p_n are used to graphically observe the expected properties. Experiments with realistic p_n values ($p_n < 0.01$) may be found in figure 5.12

Greedy (Algorithm 21) with random ordering, where each point is the average of 10,000 runs. Figure 5.10 (left) reveals that sorting the players in decreasing order is a good policy. However, it has to be noted that the difference between DORG and a random ordering is much thinner when p_n are smaller, as expected in a real setting. We also notice that DOFG expected reward loss, as compared to DORG, is below 20% until $N \approx 75$. Figure 5.10 (center) illustrates the result of Theorem 28, and indicates that the fairness lower bound is tight. It also shows that, while DOFG only loses 20% rewards when $N \approx 75$ as compared to DORG, its fairness is approximately 30 times larger.

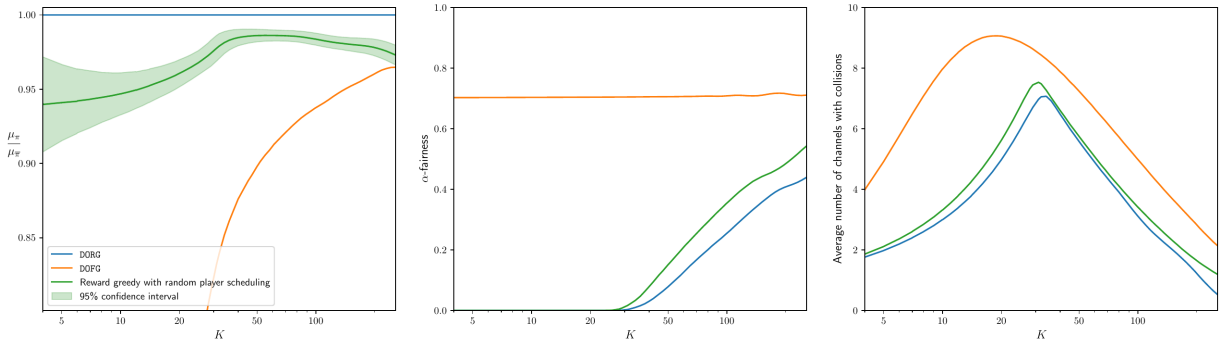


Figure 5.11: DORG and DOFG: experiments with N fixed and K varying. With number of players $N = 200$ fixed, and for K values (ranging from 4 to 256 on a log scale), the algorithms have been compared in terms of expected reward ratio with DORG (above), α-fairness (middle), and number of collided channels (below). Each point has been obtained from 10,000 problems where the parameters are sampled as follows: $\forall n \in [N], p_n \sim \mathcal{U}(0, 0.3)$ and $\forall k \in [K], \theta^k \sim \mathcal{U}(0, 1)$.

Further, on figure 5.10 (right), we notice that the expected number of channels with collisions stops increasing as N grows around $N = 100$. It is the moment when the channels get completely saturated. $N = 100$ coincides with the point where the fairness gets to 0 on figure 5.10 (center). We explain this phenomenon as follows: each channel k fills up, up to the point when $\ell^k > 1$. When all the channels reached this point, adding new players to the network actually decreases the expected reward, and DORG's strategy condemns the arms with the lowest θ^k and use them as a garbage bin for new players. These channels get so crowded that there is a collision on it with a very high probability, in order to keep the other channels functionally unspoiled. In comparison, to guarantee fairness DOFG does not throw away players on a bin channel.

Similar figures with $N = 200$ and 100 K values ranging from 4 to 256 on a log scale are available below.

5.2.4 Collaborative Exploration in Massively Multi-Player Bandits

Principle

Decentralized explore-then-exploit approach. The choice of the policy depends on the metric to be maximized: for maximizing network utility, DORG policy (Algorithm 21) should be used, while to guarantee some fairness among players, DOFG policy (Algorithm 22) is to be used. However both policies necessitate the model $\boldsymbol{\theta}$, which is unknown: hence exploration is necessary. Since the gateway cannot observe the collisions (packet losses), the learning (ex-

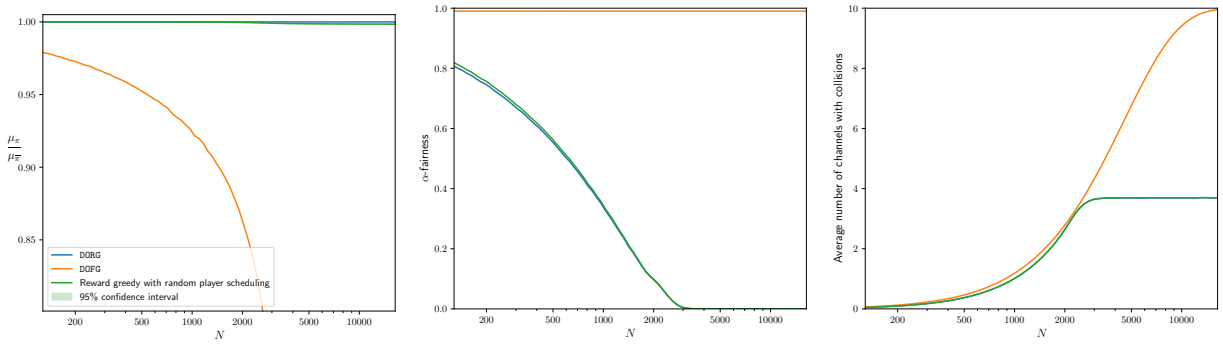


Figure 5.12: DORG and DOFG: experiments with K fixed and N varying, and smaller p_n. With number of players $K = 10$ fixed, and for N values (ranging from 128 to 16384 on a log scale), the algorithms have been compared in terms of expected reward ratio with DORG (above), α-fairness (middle), and number of collided channels (below). Each point has been obtained from 10,000 problems where the parameters are sampled as follows: $\forall n \in [N], p_n \sim \mathcal{U}(0, 0.01)$ and $\forall k \in [K], \theta^k \sim \mathcal{U}(0, 1)$.

ploration) is done at the device (player) side. Therefore, we propose a decentralized exploration algorithm performed with the packets that the players have to send and followed by an exploitation phase, i.e. *explore-then-exploit* approach: an exploration algorithm shares the probabilities of sending packets of players at the beginning, which is necessary to compute the exploration policy of each player and outputs an ϵ-approximation of the model $\boldsymbol{\theta}$ with high probability for a sufficiently small ϵ, and then a target policy is used during the exploitation phase.

Definition 29 (ϵ-approximation)**.** *$\hat{\theta}^k$ is said to be an ϵ-approximation of arm k, if the difference between it and θ^k is less than ϵ : $|\theta^k - \hat{\theta}^k| \leq \epsilon$.*

Collaboration. In order to reduce the exploration time needed to find an ϵ-approximation of each arm, we propose to distribute the exploration task on the players: each player is responsible of a predefined number of samples t_n^* for each arm according to its probability of sending a packet, so that all players would finish their estimations almost at the same time. At the end of the exploration phase, each player sends its ϵ-approximation of each arm to other players through the gateway. Then, the target policy can be computed in a centralized way (by the gateway) or separately within each player. Our exploration algorithm as the algorithm in [Féraud et al., 2019], belongs to the federated multi-armed bandits as defined in [Shi and Shen, 2021], as the players learn independently on different data and share their knowledge afterwards. In Algorithm 23, we assume that a message to other player can be sent with the packet at the same time slot (see section 5.2.1). As in [Féraud et al., 2019], we define a message as follows:

Definition 30 (message)**.** *A message is a random variable, that is sent to other player.*

Description of the algorithm

The function $\mathrm{send}(s)$, used in Algorithm 23, means that message s is sent by player n and broadcast to other players through the gateway on a channel chosen uniformly over K. The function $\mathrm{send}(s)$ returns 1 if an acknowledgement

Algorithm 23 Collaborative Exploration in Massively Multi-Player Multi-Armed Bandits

Inputs: $[K]$, $[N]$, $\epsilon \in [0,1]$, $\delta \in (0,1)$
Output: $\hat{\theta} = \{\hat{\theta}^k, \forall k \in [K]\}$
Init: $t := 0$; $\forall n \in [N] : t_n^* := \infty$, $ack1_n := 0$; $\forall (n,k) \in [N] \times [K]$: $ack2_n^k := 0, ack3_n^k := 0$

1: **repeat**
2: $\mathcal{N}_t := \{n \in [N], a_n \sim \mathcal{B}(p_n), a_n = 1\}$
3: **for** $n \in \mathcal{N}_t$ **do**
4: $k_n \sim \mathcal{U}(1,K)$
5: $Y_n^{k_n}(t_n^{k_n}) := I_n^{k_n} E^{k_n}$
6: $\hat{\mu}_n^{k_n}(\pi_u) := \sum_{t=1}^{t=t_n^{k_n}} Y_n^{k_n}(t)/t_n^{k_n}$
7: $t_n^{k_n} := t_n^{k_n} + 1$
8: **if** $ack1_n = 0$ **then**
9: $ack1_n := send(p_n)$
10: **else**
11: **if** $\forall i \in [N], ack1_i = 1$ **then**
12: $\forall i, t_i^* := \dfrac{p_i \log(2K/\delta)}{2(\epsilon \rho_i^k(\pi_u))^2 \sum_{j=1}^N p_j}$
13: **end if**
14: **if** $\exists k, t_n^k \geq t_n^*$ **then**
15: **if** $ack2_n^k = 0$ **then**
16: $ack2_n^k := \text{send}(\hat{\theta}_n^k)$
17: **else if** $ack3_n^k = 0$ **then**
18: $ack3_n^k := \text{send}(t_n^k)$
19: **end if**
20: **end if**
21: **end if**
22: **end for**
23: $t = t + 1$
24: **until** $\exists \mathcal{N}' \subset \mathcal{N}, \begin{cases} \forall k \sum\limits_{n \in \mathcal{N}'} t_n^k \geq \sum\limits_{n \in \mathcal{N}} t_n^* \\ \forall k \sum\limits_{n \in \mathcal{N}'} ack2_n^k = |\mathcal{N}'| \end{cases}$
25: all players calculate $\hat{\theta}^k := \dfrac{\sum_{n \in \mathcal{N}'} \hat{\theta}_n^k t_n^k}{\sum_{n \in \mathcal{N}'} t_n^k}$

is received by player n from the gateway or 0 else. When player n receives the probabilities of sending packets of all other players (Algorithm 23 line 11), it computes the required number of samples of each arm t_n^* according to Lemma 10. When player n samples at least t_n^* times an arm k, it sends its estimation $\hat{\theta}_n^k$ and t_n^k to other players (Algorithm 23 lines 16,18) each in a distinct message (in distinct time slots). $\hat{\theta}_n^k$ is computed according to equation (5.12). The exploration phase ends when the arms have been sampled enough by a subset of players and the estimations of this subset have been successfully sent (Algorithm 23 line 20). Finally, the players compute the global estimations of arms by combining the received local ones (Algorithm 23 line 21).

The sampling strategy used in *collaborative exploration* is the *Uniform Policy* π_u: $\forall n, \forall k, \pi_n^k = \frac{1}{K}$. Then, player n can estimate the mean reward of arms using:

$$\hat{\theta}_n^k = \frac{\hat{\mu}_n^k(\pi_u)}{\rho_n^k(\pi_u)}, \text{ where} \tag{5.12}$$
$$\rho_n^k(\pi_u) = \prod_{n'=1,n'\neq n}^{N} (1 - p_{n'}\pi_{n'}^k) = \prod_{n'=1,n'\neq n}^{N} (1 - p_{n'}/K)$$

Lemma 10. *With Algorithm 23, to obtain with a probability* $1-\delta$ *an* ϵ*-approximation of the mean rewards of arms, player* n *needs to sample each arm at least*

$$t_n^* = \left\lceil \frac{p_n \log(2K/\delta)}{2\epsilon^2 (\prod_{n'\neq N}(1 - p_{n'}/K))^2 \sum_{i=1}^{N} p_i} \right\rceil \textit{ times.}$$

5.2.5 Analysis of the algorithm

Theorem 29. ***Communication cost.*** *When Algorithm 23 stops, the number of messages sent is, with probability* $1-\delta$*, less than* $C(N(1+2K))$*, where*

$$C(m) = m \left\lceil \frac{\log m/\delta}{\log\left(1 - \sum_{k=1}^{K} \frac{(1-p_1/K)^{N-1}}{K}\theta^k\right)^{-1}} + 1 \right\rceil.$$

The communication cost presents the number of transmissions needed to successfully send the messages of Algorithm 23. Theorem 29 states an upper bound on the number of messages issued by the N players for sharing the probabilities of sending packets, and for sharing their estimations that is in $O\left(NK \log \frac{NK+N}{\delta}\right)$.

Theorem 30. ***Exploration duration.*** *With a probability at least* $1-\delta$*, when* $N \geq 2$ *Algorithm 23 stops while finding the* ϵ*-approximations of model* $\boldsymbol{\theta} = \{\theta^1, ..., \theta^K\}$ *at:*

$$t^* \leq \frac{K \log(NK/\delta)}{2\epsilon^2((1-p_1/K))^{2N-2} \sum_{i=1}^{N} p_i} \left(1 + \sqrt{\frac{K}{2p_N}}\right) + \frac{K^2}{2(p_N)^2} \log \frac{NK}{\delta} + \left(\frac{K}{p_N}\right)^{3/2} \sqrt{\frac{C(3)}{2} \log \frac{NK}{\delta}} + \frac{KC(3)}{p_N},$$

where $p_N = \min_{n\in[N]} p_n$*,* $p_1 = \max_{n\in[N]} p_n$*, and* $C(3)$ *is the needed number of transmissions to successfully send* 3 *messages.*

Theorem 30 states an upper bound on the number of time slots needed by all players to finish their estimations of the mean rewards of the arms and to share them. The left term in $O\left(K^{3/2}/\epsilon^2 \log K/\delta\right)$ is the dominating term of the upper bound of the sample complexity. It is near optimal in comparison to the lower bound of K biased coin estimations in $\Omega\left(K/\epsilon^2 \log 1/\delta\right)$ [Anthony and Bartlett, 1999].

For the regret analysis of our proposed algorithm, we define the pseudo-regret as follows:

Definition 31 (Pseudo-regret)**.** *Let π_t be a policy generated at time t by an algorithm, and $\mu_{\boldsymbol{\theta}}(\pi_t)$ be its value in model $\boldsymbol{\theta} = \{\theta^1, ..., \theta^K\}$, we define the pseudo-regret with respect to the optimal policy $\pi^*_{\boldsymbol{\theta}}$ as $R(T) = \sum_{t=1}^{T}(\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi_t))$.*

Theorem 31. ***Pseudo-regret upper bound.*** *when $\forall n \in [N], p_n = p$, and $N \geq 3$, the pseudo-regret with respect to the optimal policy $\pi^*_{\boldsymbol{\theta}}$ of Algorithm 23 followed by the policy $\pi^*_{\hat{\boldsymbol{\theta}}}$ is upper bounded by:*

$$R(T) \leq O\left(\frac{T^{2/3}\log NKT}{p^{3/2}(1-p/K)^{2N-2}N} + K^{9/4}T^{2/3}\right).$$

To show how tight this bound is we provide below a lower bound on the pseudo-regret of any explore-then-exploit approach.

Theorem 32. ***Pseudo-regret lower bound.*** *There exists a model $\boldsymbol{\theta} = \{\theta^1, ..., \theta^k\}$ and a distribution of players $p_1, ..., p_N$ such that the pseudo-regret with respect to the deterministic optimal policy $\pi^*_{\boldsymbol{\theta}}$ of any exploration algorithm that outputs an ϵ-approximation of each arm θ^k with probability at least $1 - 1/T$ and which is followed by the optimal policy using the estimated model is at least:*

$$R(T) \geq \Omega\left(T^{2/3}\frac{\log T}{N}\right).$$

Theorem 32 reveals the difficulty of the studied problem in comparison to the multi-armed bandit and multi-player bandit problems. Indeed, in the case of bandit the pseudo-regret lower bound of *explore-then-exploit* algorithms is in $\Omega(\sqrt{KT\log T})$ [Garivier et al., 2016], and in the case of multi-player bandit there exists an *explore-then-exploit* algorithm with a regret upper bound in $O(K\sqrt{T\log T})$ [Boursier and Perchet, 2019b]. The difference in power of T of the pseudo-regret lower bounds of bandits and massively multi-player bandits is due to the fact that in the problem studied, the whole model $\boldsymbol{\theta}$ is needed to compute the optimal policy, and not only the N best arms: when the exploration stops, there is no guarantee that the arms are sufficiently sampled to compute the optimal policy without mistakes of assignment of players over arms. The independence of K of the pseudo-regret lower bound of massively multi-player bandits is due to the fact that, at each time step, K players can sample the arms. Finally, the pseudo-regret lower and upper bounds are tight in T, since the pseudo-regret upper bound of 23 followed by the policy $\pi^*_{\hat{\boldsymbol{\theta}}}$ reaches the pseudo-regret lower bound (Theorem 31).

Theorem 33. ***Fairness.*** *Applying Algorithm 23 followed by `DOFG` (Algorithm 22) on $\hat{\boldsymbol{\theta}}$ returns with a probability $1-\delta$ an α-fair policy in the true model $\boldsymbol{\theta}$, with*

$$\alpha \geq 1 - p_1 - \frac{2K\epsilon}{\max_{n\in[N]} \frac{(\theta^{k_n}-\epsilon)z^{k_n}}{1-p_n}}.$$

Theorem 33 implies that, using ϵ-approximations of arms, with high probability DOFG still has the same fairness guarantee minus a term that decreases with ϵ.

5.2.6 Tests on simulated environment

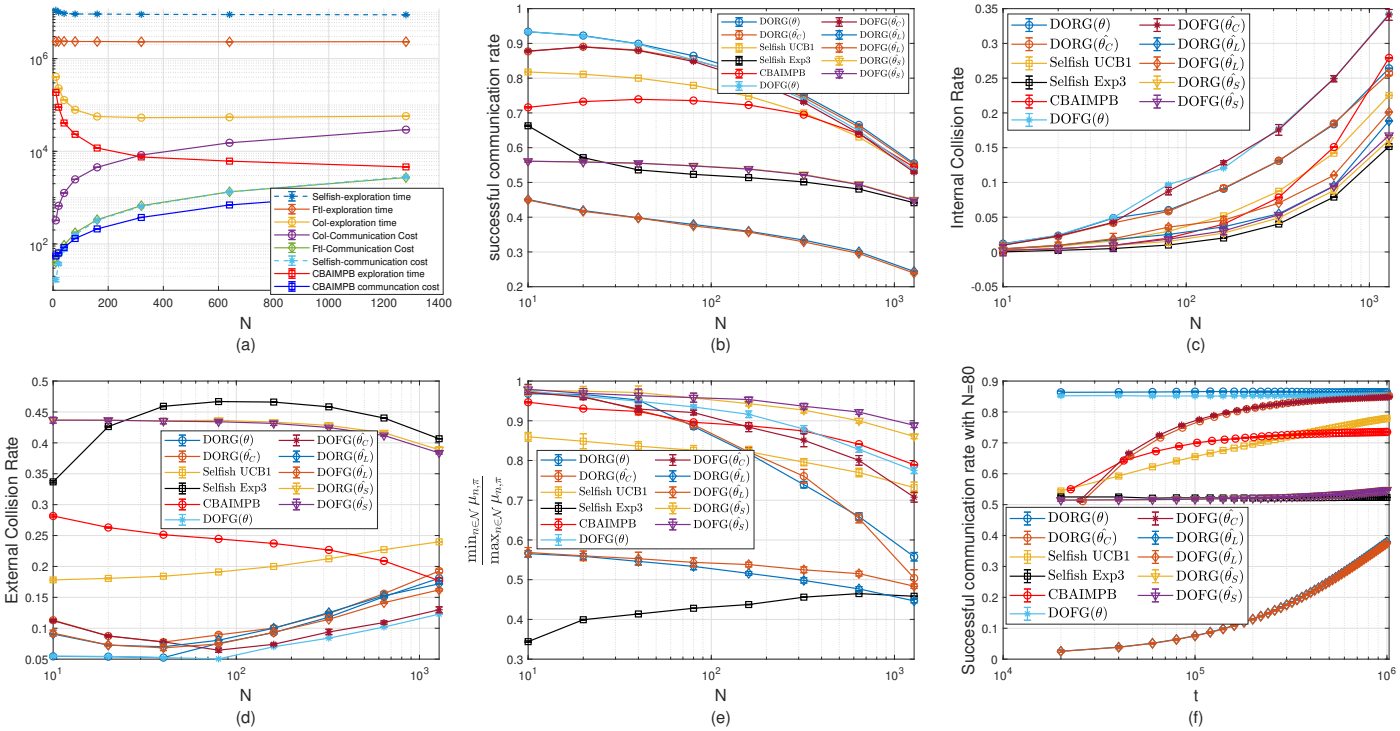


Figure 5.13: (a) exploration phase, (b) successful communication rate, (c) internal collision rate, (d) external collision rate, (e) fairness, (f) successful communication rate versus time. The successful communication and collision rates are cumulative over time. $\hat{\theta}_{\mathbf{C}}$ when *collaborative exploration* is used, $\hat{\theta}_{\mathbf{S}}$ when *selfish exploration* is used, and $\hat{\theta}_{\mathbf{L}}$ when *follow-the-leader exploration* is used. θ is the ground truth.

In order to illustrate and complete the analysis of the aforementioned algorithms, we first compare the performance of *collaborative exploration* (Algorithm 23) with *selfish exploration*, where each player explores selfishly, and with *follow-the-leader exploration (FtL)*, where only the most frequent player explores. Then we compare *collaborative exploration* followed by DORG($\hat{\theta}$) and DOFG($\hat{\theta}$), with *selfish UCB* [Bonnefoi et al., 2018] and *selfish EXP3* [Auer et al., 2002b], which respectively consist in independently playing *UCB* and *EXP3* on each player, and with CBAIMPB [Dakdouk et al., 2020], where the players find $(\epsilon', m)-$ optimal arms and exploit them uniformly with $m = 5, \epsilon' = 0.2$. We run simulations with a high number of players $N \in [10, 1200]$, and $K = 10$, such that

$\forall k, \theta^k \sim \mathcal{U}(0,1)$. The distribution of players is uniform and the upper bound of the distribution is chosen such that the internal collision rate does not exceed 0.15 when the number of players reaches 1300 and play the arms uniformly, so $\forall n, p_n \sim \mathcal{U}(3.10^{-4}, 2.2.10^{-3})$. $\delta = 0.05$, $\epsilon = 0.1$. The curves are averaged over 10 trials and run on 10^6 time steps.

In figure 5.13a, we observe that the exploration time of *collaborative exploration* is two orders of magnitude less than *follow-the-leader exploration* and three orders of magnitude less than *selfish exploration* but one order of magnitude more than *CBAIMPB*, which stops exploration when it finds the best arms. Concerning the communication cost, we observe that the communication cost of the *collaborative exploration* is only one order of magnitude greater than other exploration algorithms, however it is more than two times less than the upper bound stated in Theorem 29, which is in the order of $O\left(NK \log \frac{NK+N}{\delta}\right)$. This is due to the fact that the stopping condition of Algorithm 23 does not imply that all players have been sampled enough, but that the arms have been sampled enough. As a consequence, all the estimations of all players do not need to be shared, but only those of players that have finished their estimations.

The performance differences of the exploration policies affect the whole performance of $\texttt{DORG}(\hat{\boldsymbol{\theta}})$ and $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$, which consist of the exploration algorithm followed by the corresponding exploitation phase. That is why in figures 5.13b and 5.13f, the successful communication rate when using *selfish exploration* and *follow-the-leader exploration* are dramatically lesser than the one of *collaborative exploration*. In figures 5.13b and 5.13f, $\texttt{DOFG}(\boldsymbol{\theta})$ is slightly outperformed in terms of successful communication rate by $\texttt{DORG}(\boldsymbol{\theta})$. $\texttt{DORG}(\hat{\boldsymbol{\theta}})$ and $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$ exhibit the same behavior, and we can notice that $\texttt{DORG}(\hat{\boldsymbol{\theta}})$ and $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$ clearly outperform *selfish UCB1*, *selfish Exp3* and *CBAIMPB* , and tend to perform as well as $\texttt{DORG}(\boldsymbol{\theta})$ and $\texttt{DOFG}(\boldsymbol{\theta})$ as N increases (figure 2b). This improvement is due to their low external collision rate (figure 2d) thanks to playing more the best arms, while because of playing more the best arms their internal collision rate is higher (figure 2c). Finally, while *Selfish Exp3* is theoretically better suited for our problem setting, it is clearly outperformed by *Selfish UCB*.

Concerning fairness, $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$ clearly outperforms *selfish UCB1*, *selfish Exp3* and $\texttt{DORG}(\hat{\boldsymbol{\theta}})$, while $\texttt{DORG}(\hat{\boldsymbol{\theta}})$ is outperformed by them when N is high (figure 5.13e). *CBAIMPB* offers a high fairness between players due to the uniform selection of the arms by all players during both exploration and exploitation phases. The use of *selfish exploration* leads to high fairness level due to its very long uniform exploration phase, in contrast to *follow-the-leader exploration* that suffers of very low fairness level due to the fact that during the exploration time, only the leader can send messages.

The observed fairness of $\texttt{DOFG}(\boldsymbol{\theta})$ in figure 5.13e differs from the theoretical one (Theorem 28). This is due to the fact that the mean rewards of players are observed on a finite number of time slots (10^6). Figure 5.14 shows the progress of the fairness level achieved by $\texttt{DOFG}(\boldsymbol{\theta})$ policy as time passes. The black plot corresponds to the theoretical fairness level proved in Theorem 28. In order to reach the theoretical fairness level, the observed mean rewards of all players have to reach their expected values. Due to the low probabilities of sending packets of the

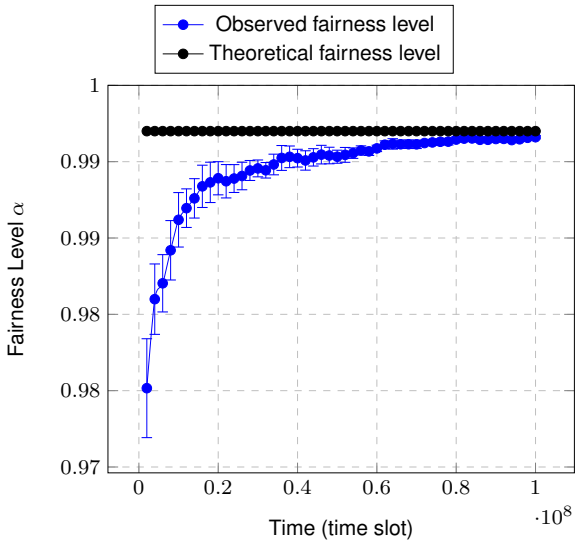


Figure 5.14: Fairness level achieved by DOFG($\boldsymbol{\theta}$) as a function of time with 10 players.

Table 5.1: Spreading factors and corresponding SNR required for ADR [Ghoslya, 2021], antenna sensitivities [Semtech,], and Inter-SF collision threshold [Waret et al., 2018]

SF	Required SNR for ADR (dB)	LoRa gateway antenna sensitivity (dBm)	Inter-SF collision threshold (dB)
SF7	-7.5	-123	-7.5
SF8	-10	-126	-9
SF9	-12.5	-139	-13.5
SF10	-15	-132	-15
SF11	-17.5	-134.5	-18
SF12	-20	-137	-22.5

players, this would take a long time. As shown by figure 5.14, the observed fairness tends to the theoretical fairness in 10^8 times steps for 10 players.

5.2.7 Tests on LoRa Network

LoRaWAN Technology

LoRaWAN is a LPWAN protocol designed to optimize LPWANs for battery lifetime, capacity, range, and cost. LoRaWAN follows a pure-ALOHA principle and is basically a single-hop technology that relays messages from the nodes to the central server via gateways. It is based on the chirp spread spectrum modulation technique, that supports 6 orthogonal spreading factors corresponding to 6 different data rates: SF7 (50 kbps) to SF12 (300 bps). The different SFs are orthogonal, allowing simultaneous transmissions of multiple frames with different SFs. At each transmission, the node selects the communication parameters including the spreading factor, radio channel, and the transmitting power that varies between 2 dBm and 14 dBm. The higher the SF (i.e., the lower the data rate and

the slower the transmission), the longer the communication range. Consequently, the choice of an SF can be seen as a trade-off between coverage and message duration (and thus, energy consumption) [Adelantado et al., 2017].

As any IoT technology, LoRa is bonded with many constraints, including the maximum duty cycle, which defines a maximum percentage of time during which an end-device can occupy a channel. Therefore, LoRaWAN nodes follow a pseudo-random channel hopping at each transmission while meeting the duty-cycle constraint which is 1% in EU 868 for example. The resulting frequency diversity makes the system more robust to interference. The choice of the spreading factor as well as the transmitting power is done at the gateway using the so-called Adaptive Data Rate (ADR) algorithm. ADR compromises between energy consumption and packet loss depending on the past performance of each end node. It was established for stationary end nodes and stable radio channel environments [Kufakunesu et al., 2020].

Adaptive Data Rate Algorithm

An Adaptive Data Rate (ADR) mechanism is built into LoRaWAN for dynamically managing each end node's link parameters in order to increase the packet delivery ratio and to decrease energy consumption. It is only suitable for static devices and should not be applied on mobile devices since the radio channel changes dramatically with every frame. We hereby present a simple baseline way to implement this decision mechanism recommended by **Semtech** [Semtech Corporation, 2021]. Its performance has been evaluated in [Marini et al., 2021]. This algorithm in its present form is limited to EU868 Industrial Scientific and Medical bands, and to 6 data rates (SF12/125kHz to SF7/125kHz). The ADR mecanism adjusts the data rate (SF) and the transmitting power of an end device based on the values of the Signal to Noise Ratio (SNR) of the last 20 transmissions (i.e., for each transmission, it considers the maximum of the various SNRs reported by the different gateways who received this given frame) following the steps below:

- an SNR margin is calculated such that:

$$\text{SNR}_{\text{margin}} = \text{SNR}_{\text{max}} - \text{SNR(SF)} - \text{margin\_db},$$

  where:

  - $\mathbf{SNR}_{\max}$ is the maximum SNR value among the last 20 received packets,
  - **margin_db** is the installation margin of the network which is a device specific static parameter, It is typically 10 dB in most networks [Semtech Corporation, 2021],
  - **SNR(SF)** is the required SNR to successfully demodulate a frame, and is a function of the SF of the end-device's last received frame and presented in table 5.1.

- $\text{N}_{\text{step}} := round(\ \text{SNR}_{\text{margin}}/3)$ is calculated to determine the number of steps to perform:

– if N_{step} is negative (i.e. SNRs are low), the transmitting power is incremented by $3 \times N_{step}$ dBm,

– if N_{step} is positive (i.e. SNRs are high), SF is decreased by N_{step}, in order to decrease the time-on-air and save energy, if SF7 is reached and there are still steps remaining, then the transmitting power is decreased by 3 dBm for each remaining step until the minimum power (2 dBm) is reached.

The end-device has also the possibility to manage its transmission parameters itself by making use of ADR mechanism that resides at the end-device side. If the end-device does not receive any downlink frame from the gateway for a certain number of sent packets, it must try to regain connectivity by first stepping up the transmit power to default power (i.e., the max power 14 dBm). It must further lower its data rate (increase the SF) step by step every predefined number of sent packets until it reaches the lowest data rate (i.e., SF12) [lor, 2021].

Notice that ADR is a heuristic and is not based on any optimization objective: it increases and decreases SF and transmitting power depending on the SNR values. It also treats each device individually regardless of other devices in the network. In this work, we contrarily aim to optimize the global network capacity by adapting massively multi-player multi-armed bandits for handling the trade-off between energy consumption and packet losses. We compared the performance of the ADR algorithm with different multi-armed bandit algorithms using a LoRa network simulator presented below.

LoRa Network Simulator

For our simulations we extended a realistic LoRa network simulator [Varsier and Schwoerer, 2017a] and adapt it to our settings. It is described below.

Network Operation. By default, LoRa devices use pure ALOHA for transmissions. However, due to the need of synchronized nodes and referring to [Ali et al., 2019] that shows that slotted-ALOHA (where a device can only transmit data in the start of a time slot) outperforms pure-ALOHA in terms of packet error rate, throughput, collision, and energy consumption, we propose that the devices transmit according to the slotted-ALOHA protocol. Each node n transmits at the beginning of a time slot with a fixed probability p_n. The time slot is of a configurable duration that together with p_n respect a duty cycle of 1%. We consider devices of class A, which after each uplink transmission open two short reception windows in order to receive a downlink transmission from the gateway as an acknowledgement of their uplink transmission reception at the gateway. The devices always receive an acknowledgement if their uplink transmission is successful. In case of a packet loss, an end-device n retransmits its packet in the next time slots with a probability $p_n^{\dagger} > p_n$ whose value depends on the application. The maximum possible number of retransmissions is configurable and depends on the device (we consider 8 maximum retransmissions in the simulator).

Transmission Success and Collision Rules. The success or failure of a transmission mainly depends on two important metrics: the Received Signal Strength Indicator (RSSI) which characterizes the power level of a received

radio signal, and the Signal to Noise Ratio (SNR). A packet is successfully received by a gateway if it does not collide with any other packets, and if its RSSI is strictly greater than the antenna sensitivity. The antenna sensitivity depends on the SF of the sent transmission as reported in table 5.1. A collision may occur when two or more packets sent on the same radio channel are received simultaneously. There are two types of collisions:

- Intra-SF collisions: occurs when the colliding packets (packet a and packet b) are of the same SF. The packet with the highest power will be decoded if it is at least 6 dB higher than the other LoRa packets: $\text{RSSI}_a - \text{RSSI}_b \geq 6$ dB.
- Inter-SF collisions: occurs when the colliding packets are of different SFs ($\text{SF}_a \neq \text{SF}_b$). The packet is demodulated if the power difference is strictly greater than the inter-SF collision threshold which depends on the SF of the corresponding frame (see table 5.1): packet "a" is demodulated if: $\text{RSSI}_a - \text{RSSI}_b > \text{Thr}(\text{SF}_a)$.

Propagation Model. Propagation is modeled by the universal Okumura-Hata model, which is an accurate and widely used propagation model for predicting path loss in urban areas. Adaptations to rural and suburban areas are also added as recommended by ETSI for GSM 900 MHz [ETSI, 2005]. This model takes into account the effects of diffraction, reflection and scattering caused by city structures. It is generally used for frequency ranges of 150 MHz to 1500 MHz, for a link distance varying from 1 km to 20 km and for antenna heights varying from 30 m to 200 m and from 1 m to 10 m for the transmitter and the base station antenna respectively [Deme et al., 2013]. Typical indoor penetration losses are considered (18 dB, 15 dB, 12 dB and 10 dB for dense urban, urban, suburban and rural environments respectively) along with additional 6 dB loss for deep indoor environments [Rodriguez et al., 2013, Ferreira et al., 2006].

Environment Modeling. Two main environmental aspects are modeled: shadowing and fast fading. Shadowing is the effect causing the received signal power to fluctuate due to objects obstructing the propagation path between the transmitter and the receiver. The resulting loss is modeled as a random variable following a log-normal distribution with a standard deviation of 12 dB (resp., 6 dB) for outdoor (resp., indoor) settings. Fast fading or Rayleigh fading is the variation of the signal power due to multipath propagation, and its resulting loss is modeled using a Rayleigh distribution.

Optimizing LoRa Communications using Massive Multi-Player Multi-Armed Bandit

At each transmission, a node selects the corresponding SF and TP , and then observes a reward. We have a set of 30 arms of pairs of (SF, TP) corresponding to the 6 possible spreading factors (SF7, SF8, SF9, SF10, SF11 and SF12) and 5 transmitting power (2 dBm, 5 dBm, 8 dBm, 11 dBm and 14 dBm). Minimizing the energy consumption while maintaining a high packet delivery ratio (PDR) are two incompatible objectives: as SF and TP increase PDR increases and energy consumption increases. That is why our approach for handling energy consumption is to

introduce a parametric function used to penalize high-energy consuming arms. We first normalize the values of the energy consumption of each arm with respect to the largest possible consumed energy (the arm with the highest power and greatest SF (SF12, 14 dBm)). Let $e^k \in (0, 1]$ be the value of the normalized energy consumed on arm k. The values of e^k are presented in table 5.2. We consider the following penalty function according to the energy consumption of arm k:

$$\xi_{\alpha,q}(e^k) = (1 - \alpha e^k)^q. \tag{5.13}$$

$\xi_{\alpha,q}(e^k)$ is a decreasing function of the energy consumption e^k. The parameters $\alpha \in [0, 1)$ and $q \geq 1$ allow to shape it, depending on the energy consumption of arms (table 5.2).

Table 5.2: The normalized energy consumption per arm e^k, where the colors from blue to red correspond to the values from low to high

TP (in dBm)	SF7	SF8	SF9	SF10	SF11	SF12
2	0.003	0.005	0.009	0.016	0.032	0.063
5	0.005	0.009	0.018	0.031	0.063	0.126
8	0.01	0.018	0.037	0.063	0.126	0.251
11	0.021	0.037	0.073	0.125	0.251	0.501
14	0.042	0.073	0.146	0.25	0.5	1

As mentioned previously, a packet is successfully received if it does not collide with any internal or external transmissions, and the RSSI is strictly greater than the antenna sensitivity. To model packet delivery, we consider three random variables for every arm k:

- $E^k \in \{0, 1\}$ denotes the event *'no external collision occurs'* (intra-SF or inter-SF collision with an unknown node),
- $I_n^k \in \{0, 1\}$ denotes the event *'no internal collision occurs for node* n*'*(intra-SF or inter-SF collision with an known node),
- $D_n^k \in \{0, 1\}$ denotes the event *'no decoding error occurs'* (RSSI lower than the antenna sensitivity).

Consequently, the event *'transmission is successful'* for node n is denoted $T_n^k \in \{0, 1\}$, such that:

$$T_n^k = E^k I_n^k D_n^k. \tag{5.14}$$

To handle both energy consumption and packet delivery we combine equations (5.13) and (5.14) in the reward function of node n playing arm k below:

$$R_n^k(\alpha, q) = (1 - \alpha e^k)^q T_n^k. \tag{5.15}$$

To handle packet delivery, the used propagation model takes into account all conditions impacting it. Inter-SF or intra-SF collisions may occur even if the transmissions are not performed using the same parameters (SF, TP). Moreover, the propagation model introduces a decoding error, which depends on the topography, the position of the node, and the position of the gateway. Notice that this realistic propagation model violates two assumptions made by the theoretical model described in section 5.2.3: the channels are orthogonal, and the arms are the same for all player. Moreover, the re-transmissions are not taken into account in the utility function (equation 5.10), and hence in the target policies `DORG` and `DOFG`. Finally, we did not modify the LoRa protocol for including an optional 8 bytes overhead for exchanging messages between players. We simply consider the messages between players as a regular transmissions. Despite there is a significant gap between the theoretical model and the true model, in the next section we will see that Massively Multi-Player Multi-Armed Bandits is a competitive candidate for choosing the connection parameters of LoRa transmissions in order to minimize the energy consumption while ensuring high reliability.

Experimental results

Experimental setup. For our simulations, we consider a network operating in the LoRa European band $863 - 870$ MHz. We consider only one gateway and assume all transmissions are done on one frequency channel (868 MHz). The network configuration and input parameters are summarized in table 5.3. We consider the worst case of a deep indoor LoRa network in an urban city. The frame size is 11 bytes (4 bytes of payload for the consumption index and 7 bytes Zigbee Cluster Library application protocol overhead) [Varsier and Schwoerer, 2017b] corresponding to a smart metering application. We consider a set of $N = 400$ end nodes where each node n has a fixed probability p_n to send a packet at the beginning of a time slot. The distribution of the nodes is uniformly chosen such that $\forall n, p_n \sim \mathcal{U}(7.10^{-4}, 5.10^{-3})$. We consider the maximum number of transmissions $= 8$. In case of a packet loss of any node n, it will increase its probability to send packets to $p_n^{\dagger} = p_n \times 8$ in order to be able to retransmit it before a new packet is needed to be sent. The communication parameters of the retransmissions are chosen according to the policy the nodes follow. In such settings, we compare the performances of ADR algorithm [Ghoslya, 2021], *selfish* UCB [Besson and Kaufmann, 2018], *selfish* Exp3 [Auer et al., 2002b], which is a commonly-used algorithm in non-stochastic environments, CBAIMPB [Dakdouk et al., 2021], and *collaborative exploration* followed by `DORG` or `DOFG`.

Due to the very slow increase of energy values near 0 and very fast increase near 1 as shown in table 5.2, we set the parameters of the penalty function (5.15) to $\alpha = 0.5$ and $q = 4$. Although `DORG` and `DOFG` assume that the mean rewards of the arms are the same for all the nodes which necessitates that all nodes be located at the same distance from the gateway, we consider here that the nodes are uniformly distributed around the gateway at three different distances $d = \{500, 1000, 2000\}$. For each trial, 5.10^5 packets are sent by the nodes. The figures present the averaged values over 40 trials with 95% confidence intervals.

Table 5.3: The network configuration and input parameters

Channel Frequency	868 MHz
Bandwidth	125 kHz
Number of Gateways	1
Gateway noise figure	3 dB
Gateway antenna gain	5 dBi
Indoor penetration loss	15 dB
Additional deep indoor loss	6 dB
Gateway antenna height	30 m
End-device height	1.5 m
End-device antenna gain	0 dBi
Targeted C/N after despreading	6 dB

We perform two different experiments, each considering different external traffics.

Experiment 1 In the first experiment, to simulate external traffic, we consider $S = 200$ static devices, each sends packets with a fixed probability $p = 0.01$. These external nodes elect an arm k for each transmission with a probability $l^k \sim \mathcal{U}(0, 1)$, such that $\sum_{k=1}^{K} l^k = 1$, which makes the environment stationary. Notice however that for selfish Exp3 or selfish UCB, which does not take into account other nodes, the environment cannot be considered as stationary, since the internal nodes can change arm and hence due to the collisions the reward function (5.15) evolves during time.

In figure 5.15 we present the average values of the total energy consumed by the end nodes, the total number of lost packets and the total sum of rewards gained by the end-devices. It clearly shows that the nodes when implementing the ADR algorithm suffer of very high energy consumption and packet loss compared to the learning methods with any inter-site distance. This directly leads to greater sum of rewards for all the learning methods, and implies that MAB algorithms guarantee better management of the trade-off between energy consumption and packet loss, and provides a better QoS.

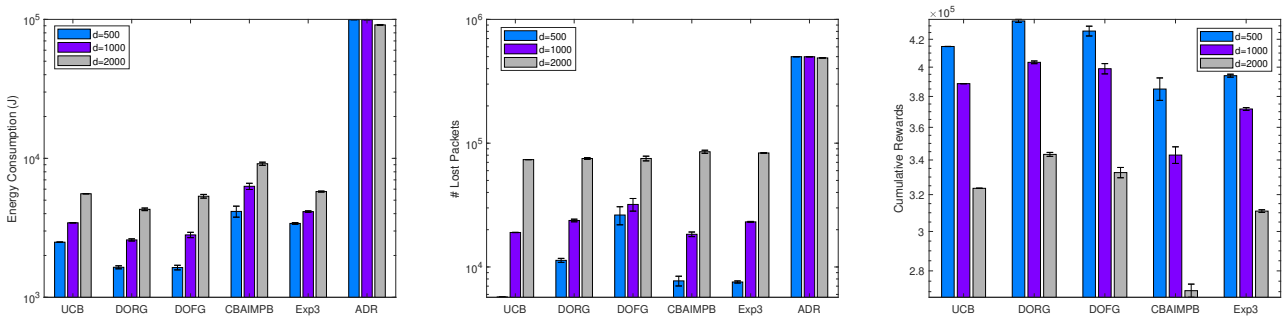


Figure 5.15: **Experiment 1**: Performance of the LoRa network with end nodes distributed in hexagonal areas centered by the gateway with three different radii and external nodes following a fixed policy. From left to right: energy consumption, number of lost packets, gathered rewards.

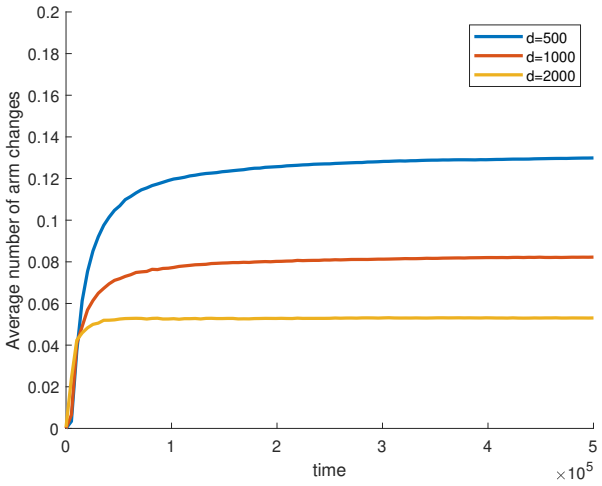


Figure 5.16: Average number of arm changes with respect to the number of plays

Despite there still being a gap between the theoretical model and the true model, DORG and DOFG largely outperform ADR in terms of energy consumption and packet losses and outperform UCB by compromising energy consumption and packet loss (see figure 5.15), while the latter shows to be highly robust against collisions. We also notice that *selfish* UCB outperform *selfish* Exp3 even though the stationary assumption is violated.

Experiment 2 In this experiment, we consider that the external nodes are LoRa devices that follow the ADR mechanism. Due to ADR mechanism, the external nodes can change of arm at each time step. This introduces a non-stationarity even for the collaborative algorithms DORG and DOFG: the percentage of ADR nodes that change their arm tends to the order of 8% (figure 5.16).

Notice that despite the non-stationary environment, the results are very similar to those in the previous experiment: all MAB algorithms outperform ADR, and our developed algorithms outperform other state-of-the-art MAB algorithms (figure 5.17). This experiment reveals that if there exist some nodes that does not following our collaborative algorithms but ADR, they will lose in term of data rate, while using more energy. Finally, notice that the *explore-then-exploit* algorithms DORG and DOFG are more appropriate for low-complexity devices (used in IoT networks) than selfish MAB algorithms, since after the exploration phase ends no computation takes place at the device side, while using MAB algorithms the devices keep computing confidence bounds or distributions to find the next arm to select.

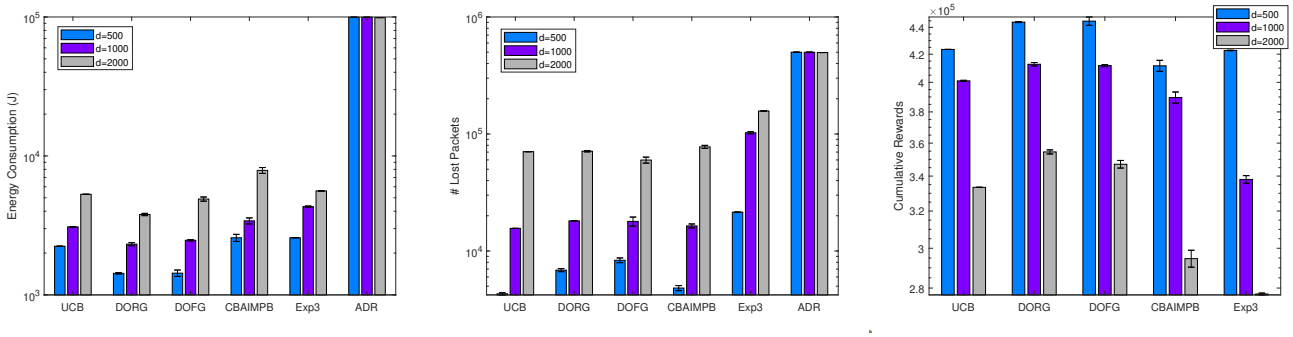


Figure 5.17: **Experiment 2**: Performance of the LoRa network with end nodes distributed in hexagonal areas centered by the gateway with three different radii and external nodes following ADR algorithm. From left to right: energy consumption, number of lost packets, gathered rewards.

5.2.8 Conclusion

With the aim of optimizing transmission in IoT networks, we proposed two greedy policies DORG and DOFG that are efficient with any number of players, and can handle internal and external collisions without *sensing*. We have shown that DORG is optimal in the setting proposed in [Bonnefoi et al., 2018] (when $\forall n, p_n = p$), and that DOFG is fair. Then, we showed that using an ϵ-approximation of the model $\boldsymbol{\theta}$, the pseudo-regret lower bound of $\texttt{DORG}(\hat{\boldsymbol{\theta}})$ is optimal with respect to T and that $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$ is fair up to an additive term that decreases with ϵ. Our experiments on simulated environment confirm the good behavior of *selfish UCB* and *CBAIMPB*, but show that both are outperformed in terms of network successful communication rate by $\texttt{DORG}(\hat{\boldsymbol{\theta}})$ and $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$, and in terms of fairness by $\texttt{DOFG}(\hat{\boldsymbol{\theta}})$. Moreover, We tested the proposed algorithms in a realistic LoRa simulator. Despite the fact that the theoretical model does not perfectly fit the true LoRa model (orthogonal channels, same arms for all players, and no re-transmissions), our experiments show that the proposed algorithms dominate ADR in terms of packet losses and energy consumption. Finally, our experiments showed that if a team of nodes plays ADR, and another team plays our collaborative algorithms, the first team is dominated.

This work can be extended in many directions: studying *explore-and-exploit* approach for the proposed problem, handling an evolving number of active players, handling more general non-stationary environments for instance using an efficient change point detection [Alami et al., 2020], handling players with different mean rewards of arms using contextual bandits [Féraud et al., 2016].

5.2.9 Broader Impact

Optimizing the communications in IoT networks has a clear *positive environmental impact*. Indeed, when the number of collisions decreases, obviously the amount of wasted energy also decreases. Moreover, IoT devices often work on batteries, and minimizing the wasted energy increases the lifetime of batteries, which reduces the amount of batteries that need to be recycled. The decrease of the number of collisions is done thanks to the cooperation

between players. In this work, we develop the concept of *fairness between players*, which is a necessary condition of cooperation. We believe that providing a mathematical framework to guarantee the fairness and then to favor cooperation is a necessity in our world where more and more automatic devices equipped with machine learning algorithms exchange information. This work is a first step in this direction.

In a real life implementation of this work, to take care about ethical consideration, we will need also to take into account that the purposes of the devices is not the same. Some of them could have significant packets to transmit, for instance for heath care and emergency purpose. The fairness has to be weighted by the purpose of the devices. Finally, in a real life application the system has to be protected against malicious players that may lie about its probability of being active or about the rewards of a channel for bypassing the fairness constraint of algorithms. We believe that this issue can be fixed by the gateway that can check the consistency of the observed rewards and probabilities of each player's activity.

5.2.10 proof

Notations

For the sake of ease the reading of proofs, we provide below the notations.

notation	meaning
N	number of players.
$[N]$	set of players.
p_n	probability that player n sends a packet.
K	number of arms.
$[K]$	set of arms.
θ_k	mean reward of arms k.
$\boldsymbol{\theta}$	model $\boldsymbol{\theta} = (\theta_1, ..., \theta_K)$.
$\hat{\theta}_k$	estimated mean reward of arms k.
$\hat{\boldsymbol{\theta}}$	estimated model $\hat{\theta} = (\hat{\theta}_1, ..., \hat{\theta}_K)$.
ϵ	approximation term.
δ	probability of failure.
π_n^k	probability that player n chooses arm k.
π_n	policy of player n, $\pi_n = (\pi_n^1, ..., \pi_n^K)$.
π	policy of players, $\pi = (\pi_1, ..., \pi_n)$.
π_u	uniform policy.
$\pi^\dagger$	decreasing order fair greedy policy generated by Algorithm 2.
$\pi^*_{\boldsymbol{\theta}}$	optimal policy in model $\boldsymbol{\theta}$, which is deterministic, when it is clear in the context, we use π^*.
$\mu_{\boldsymbol{\theta}}(\pi)$	mean reward in model $\boldsymbol{\theta}$ of the policy π, when it is clear in the context, we use $\mu(\pi)$.
	For a stochastic policy: $\mu_{\boldsymbol{\theta}}(\pi) = \sum_{k=1}^K \theta^k \sum_{n=1}^N p_n \pi_n^k \prod_{n' \neq n} (1 - p_{n'} \pi_{n'}^k)$.
	For a deterministic policy $\mu_{\boldsymbol{\theta}}(\pi) = \sum_{k=1}^K \theta^k z^k l^k$.
z^k	probability that arm k is not used by any other players, $z^k = \prod_{n' \in [N], k_n = k} (1 - p_n)$.
l^k	sum of activation odds on arm k of other players, $l^k = \sum_{n' \in [N], k_n = k} \frac{p_n}{1 - p_n}$.
k_n	arm assigned to player n.
$\pi[n]$	policy π when players $n' > n$ do not play.
$z^k[n]$	probability that arm k is not used by any of the first n players.
$l^k[n]$	sum of activation odds of the n first players for arm k.
$\rho_n^k(\pi)$	probability that no other players have chosen arm k using policy π.

Proof of Theorem 26

There exists an optimal policy which is deterministic.

Proof. We may write the global objective as:

$$\mu_{\boldsymbol{\theta}}(\pi) = \sum_{k=1}^{K} \underbrace{\theta^k}_{\text{mean reward of arm k}} \sum_{n=1}^{N} \underbrace{p_n.\pi_n^k}_{\text{probability that player n chooses arm k}} \underbrace{\prod_{n'=1,n'\neq n}^{N} (1 - p_{n'}.\pi_{n'}^k)}_{\text{probability that no collision occurs}} \tag{5.16}$$

Let us assume that $\pi^* = \{\pi_n\}_{n\in[N]}$ is optimal. Let us fix all player policies but player n's. Then, we notice that $\mu_{\boldsymbol{\theta}}(\pi)$ is linear (see (5.16)) in each $\pi_n^k, k = 1, ..., K$, meaning that the maximum is achieved for any $k_n^* \in_{k\in[K]} \frac{\partial \mu_{\boldsymbol{\theta}}(\pi)}{\partial \pi_n^k}$, and therefore the optimal policy may have been chosen so that π_n is deterministic: $\pi_n^{k^*} = 1$ and $\forall k \neq k^*, \pi_n^k = 0$. The same reasoning can be repeated for the other players, so that there exists an optimal policy that is deterministic. □

Proof of Lemma 9

For a deterministic policy π, let $\mu_{\boldsymbol{\theta}}(\pi[n])$ denote the aggregated expected reward when only the players $1, ..., n$ are playing (all players $n' > n$ are deactivated). Then we have the recursive expression

$$\mu_{\boldsymbol{\theta}}(\pi[n]) = \mu_{\boldsymbol{\theta}}(\pi[n-1]) + p_n \theta^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) z_{[n-1]}^{k_n},$$

where $z_{[n]}^k$ is the probability that arm k is not used by any of the first n players, and $\ell_{[n]}^k$ is the sum of activation odds of the n first players for arm k.

Proof. We have:

$$\begin{aligned}
\mu_{\boldsymbol{\theta}}(\pi[n]) &= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + \mu_{\boldsymbol{\theta}}(\pi[n]) - \mu_{\boldsymbol{\theta}}(\pi[n-1]) \\
&= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + \sum_{k\in[K]} \theta^k z_{[n]}^k \ell_{[n]}^k - \sum_{k\in[K]} \theta^k z_{[n-1]}^k \ell_{[n-1]}^k \\
&= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + \theta^{k_n} z_{[n]}^{k_n} \ell_{[n]}^{k_n} - \theta^{k_n} z_{[n-1]}^{k_n} \ell_{[n-1]}^{k_n} \qquad (5.17) \\
&= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + \theta^{k_n} \left(z_{[n]}^{k_n} \ell_{[n]}^{k_n} - z_{[n-1]}^{k_n} \ell_{[n-1]}^{k_n}\right) \\
&= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + \theta^{k_n} \left((1-p_n) z_{[n-1]}^{k_n} \left(\ell_{[n-1]}^{k_n} + \frac{p_n}{1-p_n}\right) - z_{[n-1]}^{k_n} \ell_{[n-1]}^{k_n}\right) \\
&= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + \theta^{k_n} \left(-p_n z_{[n-1]}^{k_n} \ell_{[n-1]}^{k_n} + p_n z_{[n-1]}^{k_n}\right) \\
&= \mu_{\boldsymbol{\theta}}(\pi[n-1]) + p_n \theta^{k_n} z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right),
\end{aligned}$$

where the line (5.17) comes from the fact that $z_{[n]}^k = z_{[n-1]}^k$ and $\ell_{[n]}^k = \ell_{[n-1]}^k$ for all $k \neq k_n$. □

Proof of Theorem 27

Lemma 11. *As long as $\ell_{n-1}^k \leq 2$, the reward-greedy criterion for Algorithm 21 decreases as we add a new player n:*

$$z_{[n]}^k \left(1 - \ell_{[n]}^k\right) \leq z_{[n-1]}^k \left(1 - \ell_{[n-1]}^k\right). \tag{5.18}$$

Proof. We look at the difference:

$$\begin{aligned}
\forall k \neq k_n, \quad z_{[n]}^k \left(1 - \ell_{[n]}^k\right) - z_{[n-1]}^k \left(1 - \ell_{[n-1]}^k\right) &= 0 && (5.19)\\
z_{[n]}^{k_n} \left(1 - \ell_{[n]}^{k_n}\right) - z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) &= (1-p_n) z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n} - \frac{p_n}{1-p_n}\right) \\
&\quad - z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) && (5.20)\\
&= (1-p_n) z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) - p_n z_{[n-1]}^{k_n} \\
&\quad - z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) && (5.21)\\
&= -p_n z_{[n-1]}^{k_n} \left(1 - \ell_{[n-1]}^{k_n}\right) - p_n z_{[n-1]}^{k_n} && (5.22)\\
&= -p_n z_{[n-1]}^{k_n} \left(2 - \ell_{[n-1]}^{k_n}\right) && (5.23)
\end{aligned}$$

Since p_n and $z_{[n-1]}^{k_n}$ are always positive, we may conclude.

□

Theorem 27: *If $\sum_{n\in[N]} \frac{p_n}{1-p_n} \leq K+1$, then, there exists an ordering over players $\sigma^* : [N] \to [N]$ such that Algorithm 21 returns an optimal policy.*

Proof. The proof makes use of Lemma 11 which states that, as long as $\ell_{n-1}^k \leq 2$, the reward-greedy criterion for Algorithm 21 decreases as we add a new player n.

We prove below that this Lemma applies for all picked arms if $\sum_{n\in[N]} \frac{p_n}{1-p_n} \leq K+1$. By *reductio ad absurdum*, we assume that $\sum_{n\in[N]} \frac{p_n}{1-p_n} \leq K+1$ and that there exists some arm k and some player ordering σ (not necessarily σ^*) such that $\pi^*(\sigma(N)) = k$ and $\ell_{\sigma([N-1])}^k > 2$, where π^* is an optimal policy and $\sigma([N-1])$ denotes the $N-1$ first indexes in the σ reordering. Then, there must exist an arm k' for which $\ell_{\sigma([N-1])}^{k'} < 1$, otherwise we would have $\sum_{n\in[N]} \frac{p_n}{1-p_n} > \sum_{n\in[N-1]} \frac{p_{\sigma(n)}}{1-p_{\sigma(n)}} > K+1$. It means that, for k', the reward-greedy criterion $z_{\sigma([N-1])}^{k'} \left(1 - \ell_{\sigma([N-1])}^{k'}\right)$ is positive, and therefore larger than that of k: $z_{\sigma([N-1])}^{k} \left(1 - \ell_{\sigma([N-1])}^{k}\right)$, which is negative. As Lemma 9 states that the reward-greedy criterion is incrementally optimal, it means that k' would have been a strictly better arm for player $\sigma(N)$, which contradicts the assumption that π^* is optimal.

Let an optimal policy π^* be given, and let us construct the player ordering σ^* such that Algorithm 21 applied on the σ^* ordering returns π^*.

Algorithm 24 Reconstruction of a player ordering that allows Algorithm 21 to return π^*

Inputs: $[K]$, $[N]$, $\{\theta^k\}_{k\in[K]}$, $\{p_n\}_{n\in[N]}$, π^*
Output: σ^* such that Algorithm 21 returns π^*
Init: per-arm inactivity probabilities: $z^k = 1$.
Init: per-arm activation odds sums: $\ell^k = 0$.
Init: Set of players remaining to be assigned: $\mathcal{N} = [N]$.
1: **for** $n = 1$ to N **do**
2: Let $\sigma^*(n)$ be an element of $\mathcal{N}$ such that $\pi^*(\sigma^*(n)) \in \arg\max_{k\in[K]} \theta^k z^k (1 - \ell^k)$.
3: Update $\mathcal{N} \leftarrow \mathcal{N} - \{\sigma^*(n)\}$.
4: Update $z^{k_n} \leftarrow z^{k_n} \left(1 - p_{\sigma^*(n)}\right)$.
5: Update $\ell^{k_n} \leftarrow \ell^{k_n} + \frac{p_{\sigma^*(n)}}{1-p_{\sigma^*(n)}}$.
6: **end for**

It is direct to understand that Algorithm 21 applied on a σ^* player ordering would retrieve π^*. Indeed, Algorithm 24 makes it so the players are ordered to be incrementally optimal. The last piece of the proof is to check the existence of a player $\sigma^*(n)$ assigned to a reward-greedy arm on line 2.

Again by *reductio ad absurdum*, we assume that there is no remaining player that π^* assigned to a reward-greedy arm k^*. Then, it means that until the last selection, this arm will not be picked and another arm k will be picked instead. We showed at the beginning of the proof that the reward-greedy criterion is only decreasing as the arms are being selected, and that the reward-greedy criterion of an arm not being selected, such as k^*, is constant. So it means that $\pi^*(\sigma^*(N))$ should be k^*, hence, the contradiction.

We may therefore conclude the proof by stating that Algorithm 24 will never fail to construct σ^* and that Algorithm 21 applied to the σ^* player ordering will return π^*. □

Proof of Theorem 28

DOFG generates α-fair policies, with

$$\alpha \geq 1 - \max_{n\in[N]} p_n. \tag{5.24}$$

Proof. Let $\pi^\dagger$ be the policy generated by DOFG. For every arm, we have the following equality:

$$\mu_{n,\boldsymbol{\theta}}(\pi^\dagger) = \theta^{k_n} \prod_{n'\neq n, \text{ s.t. } k_{n'}=k_n} (1 - p_{n'}) = \frac{\theta^{k_n} z^{k_n}}{1 - p_n}. \tag{5.25}$$

We prove now that $\min_{n\in[N]} \mu_{n,\boldsymbol{\theta}}(\pi^\dagger) = \mu_{N,\boldsymbol{\theta}}(\pi^\dagger)$. We proceed by induction. The base case is direct for $N = 1$. Now, we prove the induction step by assuming that it is true for N and prove it for $N + 1$. We have to distinguish two cases whether k_N equals k_{N+1} or not.

Case $k_N = k_{N+1}$, then from Equation 5.25, we have $\mu_{N+1,\boldsymbol{\theta}}(\pi^\dagger) = \frac{1-p_N}{1-p_{N+1}} \mu_{N,,\boldsymbol{\theta}}(\pi^\dagger)$. Since we know by construction that $p_{N+1} \leq p_N$, we may conclude that $\mu_{N+1,\boldsymbol{\theta}}(\pi^\dagger) \leq \mu_{N,\boldsymbol{\theta}}(\pi^\dagger)$.

Case $k_N \leq k_{N+1}$, then stating that $\mu_{N+1,\boldsymbol{\theta}}(\pi^\dagger) > \mu_{N,\boldsymbol{\theta}}(\pi^\dagger)$ would imply that k_N was not optimally selecting the arm at the previous step, which brings a contradiction.

Let us assume without loss of generality that player N has been assigned to arm K. Since $\pi_N^\dagger$ has been chosen so that to maximize $\theta^k z^k$ at iteration N, it means that:

$$\min_{n\in[N]} \mu_{n,\boldsymbol{\theta}}(\pi^\dagger) = \mu_{N,\boldsymbol{\theta}}(\pi^\dagger) \geq \max_{k\in[K]} \theta^k z^k. \tag{5.26}$$

We also know that:

$$\begin{aligned}
\max_{n\in[N]} \mu_{n,\boldsymbol{\theta}}(\pi^\dagger) &= \max_{n\in[N]} \frac{\theta^{k_n} z^{k_n}}{1-p_n} && (5.27)\\
&\leq \frac{\max_{k\in[K]} \theta^k z^k}{1-\max_{n\in[N]} p_n} && (5.28)\\
&\leq \frac{1}{1-p_1} \min_{n\in[N]} \mu_{n,\boldsymbol{\theta}}(\pi^\dagger), && (5.29)
\end{aligned}$$

which concludes the demonstration. □

Proof of Lemma 10

By using Algorithm 23, in order to obtain with a probability $1-\delta$ an ϵ-approximation of the mean rewards of arms, player n needs to sample each arm at least

$$t_n^* = \left\lceil \frac{p_n \log(2K/\delta)}{2\epsilon^2 (\prod_{n'\neq N}(1-p_{n'}/K))^2 \sum_{i=1}^N p_i} \right\rceil \text{ times.}$$

Proof. Due to equations 5.9 and 5.12, for a given probability of failure $\delta \in [0,1]$, and a given approximation factor ϵ, $\forall n \in [N]$, $\forall k \in [K]$ we have:

$$P(|\mu^k - \hat{\mu}_n^k| \geq \epsilon) \leq \frac{\delta}{K} \iff P(|\theta^k - \hat{\theta}_n^k| \geq \epsilon'_n) \leq \frac{\delta}{K}, \tag{5.30}$$

where $\epsilon'_n = \epsilon . \prod_{n'\neq n}(1-p_{n'}/K)$.

Applying Hoeffding's inequality:

$$P(|\theta_n^k - \hat{\theta}_n^k| \geq \epsilon'_n) \leq 2e^{-2t_n^k {\epsilon'_n}^2}. \tag{5.31}$$

Therefore for obtaining an ϵ-approximation of arm k on player n with a probability $1-\frac{\delta}{K}$:

$$t_n^k \geq \frac{\log(2K/\delta)}{2\epsilon_n'^2} \iff t_n^k \geq \frac{\log(2K/\delta)}{2\epsilon^2(\prod_{n'\neq n}(1-p_{n'}/K))^2} \geq t^\dagger = \frac{\log(2K/\delta)}{2\epsilon^2(\prod_{n'\neq N}(1-p_{n'}/K))^2}$$

Now, as Algorithm 23 shares the estimations of the N players for finding ϵ-approximation of arm k with high probability, we need $\sum_{n=1}^N t_n^* = t^\dagger$ samples.

Hence, if each player samples arm k at least $t_n^* \geq [] \frac{p_n \log(2K/\delta)}{2\epsilon^2(\prod_{n'\neq N}(1-p_{n'}/K))^2 \sum_{i=1}^N p_i}$ times, an ϵ-approximation of arm θ^k is obtained with a probability $1-\frac{\delta}{K}$.

□

Proof of Theorem 29

Lemma 12. *In Algorithm 23, so that player n sends successfully m messages, with a probability $1-\delta$ player n needs to issue a number of transmissions $C(m)$, which is at most:*

$$m\left\lceil \frac{\log m/\delta}{\log\left(1-\sum_{k=1}^K \frac{(1-p_1/K)^{N-1}}{K}\theta^k\right)^{-1}} + 1 \right\rceil \textit{ transmissions.}$$

Proof. Let $C(1)$ be the random variable corresponding to the number of transmissions of player n to send a message. $C(1)$ follows a geometric distribution with a probability of success $p = \mu_n(\pi_u) = \sum_{k=1}^K \frac{\rho_n(\pi_u)}{K}\theta^k$, and probability of failure $q = 1-p$. Let F be the number of failures before the success. We have:

$$\mathbb{P}(C(1) \leq F+1) = 1-q^F = 1-\delta,$$
$$\implies F = \left\lceil \frac{\log\delta}{\log q} \right\rceil$$

Assuming that $p_1 \geq p_2, .., p_{N-1} \geq p_N$, we get $\rho_n(\pi_u) = \prod_{n'\neq n}(1-p_{n'}/K) \leq (1-p_1/K)^{N-1}$. Consequently, for sending m messages, with a probability $1-\delta$ any player needs at most :

$$C(m) \leq m\left\lceil \frac{\log\delta/m}{\log(1-\sum_{k=1}^K \frac{(1-p_1/K)^{N-1}}{K}\theta^k)} + 1 \right\rceil$$

□

Theorem 29 *When Algorithm 23 stops, the number of messages sent is, with probability $1-\delta$, less than $C(N(1+2K))$, where*

$$C(m) = m\left\lceil \frac{\log m/\delta}{\log\left(1-\sum_{k=1}^K \frac{(1-p_1/K)^{N-1}}{K}\theta^k\right)^{-1}} + 1 \right\rceil.$$

Proof. The required number of messages to send during Algorithm 23 is at most $N(1+2K)$. Using Lemma 12, the total number of transmissions done by all players to send successfully their messages is with probability $1-\delta$:

$$C(N(1+2K)) \leq N(1+2K)\left\lceil \frac{\log \delta/(N(1+2K))}{\log(1-\sum_{k=1}^{K}\frac{(1-p_1/K)^{N-1}}{K}\theta^k)} + 1 \right\rceil \tag{5.32}$$

□

Proof of Theorem 30

With a probability at least $1-\delta$, when $N \geq 3$, Algorithm 23 stops while finding the ϵ-approximations of θ at:

$$t^* \leq \frac{K\log(NK/\delta)}{2\epsilon^2((1-p_1/K))^{2N-2}\sum_{i=1}^{N}p_i}\left(1+\sqrt{\frac{K}{2p_N}}\right) + \frac{K^2}{2(p_N)^2}\log\frac{NK}{\delta} + \left(\frac{K}{p_N}\right)^{3/2}\sqrt{\frac{C(3)}{2}\log\frac{NK}{\delta}} + \frac{KC(3)}{p_N},$$

where p_N is the lowest probability of sending a packet among the players, and $C(3)$ is the needed number of transmissions to successfully send 3 messages.

Proof. A player n stops, while finding its estimations with high probability, when it plays each arm k at least t_n^* times (Lemma 10). Let t_n^k be the number of plays of arm k by player n before the algorithm stops at time t^* with high probability. t_n^k is a binomial random variable with parameters t^* and p_n/K. Then we have:

$$\mathbb{E}[t_n^k] = \frac{p_n}{K}.t^* \tag{5.33}$$

The estimation does not terminate if this event occurs: $E = \{\exists n \in [N], \exists k \in [K], t_n^k < t_n^* + C(3)\}$.

Applying Hoeffding's inequality we get:

$$\mathcal{P}(t_n^k - \frac{p_n}{K}.t^* < -\epsilon) \leq \exp^{-2\frac{\epsilon^2}{t^*}} = \frac{\delta}{NK}. \tag{5.34}$$

Hence, when E does not occur $\implies \forall n$ we have with probability at most δ:

$$t_n^* + C(3) - \frac{p_n}{K}.t^* < -\sqrt{\frac{t^*}{2}\log\frac{NK}{\delta}}, \tag{5.35}$$

$$\Leftrightarrow -\frac{p_n}{K}.t^* + \sqrt{\frac{t^*}{2}\log\frac{NK}{\delta}} + t_n^* + C(3) < 0, \tag{5.36}$$

$$\Leftrightarrow \sqrt{t^*} > \frac{K}{2p_n}\left(\sqrt{\frac{1}{2}\log\frac{NK}{\delta}} + \sqrt{\frac{1}{2}\log\frac{NK}{\delta} + 4\frac{p_n}{K}(t_n^* + C(3))}\right), \tag{5.37}$$

$$\Leftrightarrow t^* > \frac{K^2}{4(p_n)^2}\left(\sqrt{\frac{1}{2}\log\frac{NK}{\delta}} + \sqrt{\frac{1}{2}\log\frac{NK}{\delta} + 4\frac{p_n}{K}(t_n^* + C(3))}\right)^2, \tag{5.38}$$

Then, when E does not occur and hence the estimation terminates, we have $\forall n$ with probability at least $1-\delta$:

$$t^* \leq \frac{K^2}{4(p_n)^2}\left(\sqrt{\frac{1}{2}\log\frac{NK}{\delta}}+\sqrt{\frac{1}{2}\log\frac{NK}{\delta}+4\frac{p_n}{K}(t_n^*+C(3))}\right)^2, \tag{5.39}$$

$$\Leftrightarrow t^* \leq \frac{K^2}{4(p_n)^2}\log\frac{NK}{\delta}+\frac{K(t_n^*+C(3))}{p_n}+\frac{K^2}{2(p_n)^2}\sqrt{\frac{1}{2}\log\frac{NK}{\delta}}\sqrt{\frac{1}{2}\log\frac{NK}{\delta}+4\frac{p_n}{K}(t_n^*+C(3))}, \tag{5.40}$$

$$\Rightarrow t^* \leq \frac{K^2}{4(p_n)^2}\log\frac{NK}{\delta}+\frac{K(t_n^*+C(3))}{p_n}+\frac{K^2}{4(p_n)^2}\log\frac{NK}{\delta}+\frac{K}{p_n}\sqrt{\frac{K}{2p_n}(t_n^*+C(3))\log\frac{NK}{\delta}}, \tag{5.41}$$

$$\Rightarrow t^* \leq \frac{K}{p_n}\left(t_n^*+C(3)+\sqrt{\frac{K}{2p_n}(t_n^*+C(3))\log\frac{NK}{\delta}}\right)+\frac{K^2}{2(p_n)^2}\log\frac{NK}{\delta}. \tag{5.42}$$

Then using Lemma 10, the following inequality holds with a probability at least $1-\delta$:

$$t^* \leq \frac{K}{p_n}\left(\frac{p_n\log(2K/\delta)}{2\epsilon^2(\prod_{n'\neq N}(1-p_{n'}/K))^2\sum_{i=1}^N p_i}+C(3)+\sqrt{\frac{K}{2p_n}\left(\frac{p_n\log(2K/\delta)}{2\epsilon^2(\prod_{n'\neq N}(1-p_{n'}/K))^2\sum_{i=1}^N p_i}+C(3)\right)\log\frac{NK}{\delta}}\right) \tag{5.43}$$

$$+\frac{K^2}{2(p_n)^2}\log\frac{NK}{\delta},$$

$$t^* \leq \frac{K}{p_n}\left(\frac{p_n\log(NK/\delta)}{2\epsilon^2(\prod_{n'\neq N}(1-p_{n'}/K))^2\sum_{i=1}^N p_i}+C(3)+\sqrt{\frac{K}{2p_n}}\frac{p_n\log(NK/\delta)}{2\epsilon^2(\prod_{n'\neq N}(1-p_{n'}/K))^2\sum_{i=1}^N p_i}+\sqrt{\frac{K}{2p_n}C(3)\log(NK/\delta)}\right) \tag{5.44}$$

$$+\frac{K^2}{2(p_n)^2}\log\frac{NK}{\delta},$$

$$t^* \leq \frac{K\log(NK/\delta)}{2\epsilon^2((1-p_1/K))^{2N-2}\sum_{i=1}^N p_i}\left(1+\sqrt{\frac{K}{2p_N}}\right)+\frac{K^2}{2(p_N)^2}\log\frac{NK}{\delta}+\left(\frac{K}{p_N}\right)^{3/2}\sqrt{\frac{C(3)}{2}\log\frac{NK}{\delta}}+\frac{KC(3)}{p_N}, \tag{5.45}$$

where p_N and p_1 are respectively the lowest and the greatest probability of sending a packet among the players.

□

Proof of Theorem 31

Lemma 13. *The expected instantaneous regret in the model $\boldsymbol{\theta}$ of the target policy $\pi^*_{\hat{\boldsymbol{\theta}}}$ using the estimated model $\hat{\boldsymbol{\theta}}$ with respect to the optimal policy $\pi^*_{\boldsymbol{\theta}}$ using the true model $\boldsymbol{\theta}$ is upper bounded by:*

$$\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}})-\mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})\leq 2K\epsilon, \tag{5.46}$$

where $\mu_{\boldsymbol{\theta}}(\pi)$ denotes the mean reward of the policy π in the model $\boldsymbol{\theta}$.

Proof.

$$\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}}) = \mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\boldsymbol{\theta}}) + \mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\hat{\boldsymbol{\theta}}}) + \mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\hat{\boldsymbol{\theta}}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}}) \tag{5.47}$$

Then, we have:

- $\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\boldsymbol{\theta}}) = \sum_{k=1}^{K} z^k l^k \theta^k - \sum_{k=1}^{K} z^k l^k \hat{\theta}^k \leq K\epsilon$,
- $\mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\hat{\boldsymbol{\theta}}}) \leq 0$, since $\pi^*_{\hat{\boldsymbol{\theta}}}$ is the best policy in the model $\hat{\boldsymbol{\theta}}$.
- $\mu_{\hat{\boldsymbol{\theta}}}(\pi^*_{\hat{\boldsymbol{\theta}}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}}) = \sum_{k=1}^{K} \hat{z}^k \hat{l}^k \hat{\theta}^k - \sum_{k=1}^{K} \hat{z}^k \hat{l}^k \theta^k \leq K\epsilon$.

□

Theorem 31. *When $N \geq 3$, and $\forall n \in [N], p_n = p$, the pseudo-regret with respect to the target policy π^* of Algorithm 3 followed by a policy $\pi^*_{\hat{\boldsymbol{\theta}}}$ is upper bounded by:*

$$R(T) \leq O\left(\frac{T^{2/3}\log NKT}{p^{3/2}(1-p/K)^{2N-2}N} + K^{9/4}T^{2/3}\right).$$

Proof. Let T be the time horizon, π_u be the uniform policy used in Algorithm 3, which outputs an ϵ-approximation with high probability of $\boldsymbol{\theta}$, and $\pi^*_{\boldsymbol{\theta}}$ be the optimal policy. Let t^* be stopping time of the exploration phase. Then, the pseudo-regret with respect to a target policy $\pi^*_{\boldsymbol{\theta}}$ of Algorithm 3 is expressed as:

$$R(T) = t^*((\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - (\mu_{\boldsymbol{\theta}}(\pi_u)) + (T-t^*)((\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})), \tag{5.48}$$

where $\mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})$ denotes the mean reward in the model $\boldsymbol{\theta}$ of the optimal policy using the estimated model $\hat{\boldsymbol{\theta}}$. The left term of equation 5.48 is the instantaneous pseudo-regret of the exploration policy $\pi_{\mathcal{U}}$, and the right term is the instantaneous pseudo-regret of the estimated optimal policy $\pi^*_{\hat{\boldsymbol{\theta}}}$.

Theorem 30 allows us to upper-bound the stopping time of Algorithm 3 with t^* on an event of high probability $1-\delta$:

$$t^* \leq \frac{K\log(NK/\delta)}{2\epsilon^2((1-p_1/K))^{2N-2}\sum_{i=1}^{N} p_i}\left(1+\sqrt{\frac{K}{2p_N}}\right) + \frac{K^2}{2(p_N)^2}\log\frac{NK}{\delta} + \left(\frac{K}{p_N}\right)^{3/2}\sqrt{\frac{C(3)}{2}\log\frac{NK}{\delta}} + \frac{KC(3)}{p_N}. \tag{5.49}$$

When $\forall n \in [N], p_n = p$, with a probability $1-\delta$, we have:

$$t^* \leq \frac{K\log(NK/\delta)}{2\epsilon^2(1-p/K)^{2N-2}Np}\left(1+\sqrt{\frac{K}{2p}}\right) + \frac{K^2}{2p^2}\log\frac{NK}{\delta} + \left(\frac{K}{p}\right)^{3/2}\sqrt{\frac{C(3)}{2}\log\frac{NK}{\delta}} + \frac{KC(3)}{p}. \tag{5.50}$$

The instantaneous pseudo-regret of uniform policy with respect to the optimal policy $\pi^*_{\boldsymbol{\theta}}$ is upper bounded by:

$$\mu_\theta(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi_u) \leq K$$

and on the other hand we know by Lemma 13 that:

$$\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}}) \leq 2K\epsilon \tag{5.51}$$

Then the pseudo-regret is controlled by the trivial upper bound KT on the complementary event of probability less than δ:

$$R(T) \leq t^*(\mu_\theta(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi_u)) + (T - t^*)(\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})) + \delta KT \tag{5.52}$$

$$\tag{5.53}$$

Then, by setting $\delta = 1/T$, the pseudo-regret of Algorithm 23 followed by a policy $\pi^*_{\hat{\boldsymbol{\theta}}}$ is:

$$\begin{aligned} R(T) &\leq Kt^* + (T - t^*) \times 2K\epsilon + K, && (5.54)\\ &\leq Kt^* + 2K\epsilon T + K, && (5.55)\\ &\leq O\left(\frac{K^{5/2}\log NKT}{p^{3/2}\epsilon^2(1-p/K)^{2N-2}N} + \frac{K^2}{2p^2}\log NKT + KT\epsilon\right). && (5.56) \end{aligned}$$

Finally, by setting $\epsilon = K^{5/4}/\sqrt[3]{T}$, we conclude the proof:

$$R(T) \leq O\left(\frac{T^{2/3}\log NKT}{p^{3/2}(1-p/K)^{2N-2}N} + K^{9/4}T^{2/3}\right). \tag{5.57}$$

□

Proof of Theorem 32

There exists a model $\boldsymbol{\theta} = \{\theta^1, ..., \theta^k\}$ and a distribution of players $p_1, ..., p_N$ such that the pseudo-regret with respect to the deterministic optimal policy $\pi^*_{\boldsymbol{\theta}}$ of any exploration algorithm that outputs an ϵ-approximation of each arm θ^k with probability at least $1 - 1/T$ and which is followed by the optimal policy using the estimated model is at least:

$$R(T) \geq \Omega\left(T^{2/3}\frac{\log T}{N}\right).$$

Proof. In the following we show that a lower bound holds for a class of models $\boldsymbol{\theta}$ and distribution of players $p_1, ..., p_N$. Without loss of generality, we assume in the following that:

- $\theta^1 \geq \theta^2, ..., \theta^{K-1} \geq \theta^K$,
- $p_1 \geq p_2, ..., p_{N-1} \geq p_N$.

Choice of a class of problems. The most difficult point for evaluating a regret lower bound is that in the general case, the optimal policy, which maximizes the mean reward (see equation (5.11)), is unknown. For handling this point we choose a particular class of problems, where $N = K + 1$. Then, we assume that the distribution of players and the mean rewards of arms are such that:

$$\begin{cases} \forall k \in [K-1] \quad \theta^k = \theta^{k+1} + \epsilon, \\ p_1 > p_2 = ... = p_K > p_{K+1}, \\ p_1(1-p_{K+1}) + p_{K+1}(1-p_1) = p_2, \\ p_2(1-p_{K+1}) + p_{K+1}(1-p_2) > p_2, \\ \forall k \in [K] \quad \frac{\epsilon}{2p_k} < \theta^k. \end{cases} \tag{5.58}$$

The optimal policy. When $\frac{\epsilon}{2p_k} < \theta^k$ (equation (5.58)), superposing players on any arm provides less reward than spreading players on the arms. Indeed, let Δ_s be the gap between the mean reward of two players $k_1, k_2, k_1 < k_2 \leq K$ assigned on different arms, and the mean reward of two players assigned on the same arm:

$$\Delta_s = p_{k_1}\theta^{k_1} + p_{k_2}\theta^{k_2} - p_{k_1}\theta^{k_1}(1-p_{k_2}) - p_{k_2}\theta^{k_1}(1-p_{k_1}), \tag{5.59}$$

$$= p_{k_2}(\theta^{k_2} - \theta^{k_1}) + 2p_{k_1}p_{k_2}\theta^{k_1}, \tag{5.60}$$

$$= -p_{k_2}\epsilon + 2p_{k_1}p_{k_2}\theta^{k_1} > 0. \tag{5.61}$$

Let $\Delta_{1,2}$ be the difference between the mean reward of policy that assigns player $K+1$ on arm 1 and the one that assigns it on arm 2.

$$\Delta_{1,2} = (p_1(1-p_{K+1}) + p_{K+1}(1-p_1))\theta^1 + p_2\theta^2 - p_1\theta^1 - (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^2 \tag{5.62}$$

$$= p_2\theta^1 - p_1\theta^1 + p_2\theta^2 - (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^2 < 0 \tag{5.63}$$

Now let $\Delta_{2,k}$ be the difference between the mean reward of policy that assigns player $K+1$ on arm 2 and the one that assigns it on arm $k > 2$.

$$\Delta_{2,k} = (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^2 + p_2\theta^k - p_2\theta^2 - (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^k \tag{5.64}$$

$$= (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))(\theta^2 - \theta^k) - p_2(\theta^2 - \theta^k) > 0 \tag{5.65}$$

Hence, when equation (5.58) holds, the optimal assignment of players over arms is:

$$\pi^*_{\boldsymbol{\theta}} = (p_1, \theta^1), (p_2, p_{K+1}, \theta^2), ..., (p_{K-1}, \theta^{K-1}), (p_K, \theta^K). \tag{5.66}$$

The optimal exploration policy. As an ϵ-approximation of each arm is needed to compute the optimal policy. The optimal exploration policy plays each arm the same expected (with respect to the distribution of players $\boldsymbol{p}$) number of times. When equation (5.58) holds, any optimal exploration policy belongs to the following set:

$$\pi^*_E \in \{m \in [K], \forall n \in [K] \setminus \{1\}, k \in [K] \setminus \{m\} : (p_n, \theta^k), (p_1, p_{K+1}, \theta^m)\}. \tag{5.67}$$

Hence any other assignment of players over arms generates more collisions.

Pseudo-regret decomposition. Let T be the time horizon. Let π^*_E be the optimal (in term of sample complexity) exploration policy that outputs an ϵ-approximation with high probability of $\boldsymbol{\theta}$, i.e. each arm θ^k, and $\pi^*_{\boldsymbol{\theta}}$ be the optimal policy. We consider the time t^*, where the optimal exploration algorithm π^*_E outputs exactly an ϵ-approximation of model $\boldsymbol{\theta}$. Then, the pseudo-regret with respect to the deterministic policy $\pi^*_{\boldsymbol{\theta}}$ is expressed as:

$$R(T) = t^*(\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_E)) + (T - t^*)(\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})), \tag{5.68}$$

where $\mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})$ denotes the mean reward in the model $\boldsymbol{\theta}$ of the optimal policy using the estimated model $\hat{\boldsymbol{\theta}}$.

Lower bound of the right term. The right term equation (5.68) is the instantaneous regret of the estimated optimal policy $\pi^*_{\hat{\boldsymbol{\theta}}}$. For stating a lower bound on this term, we lower bound it by the minimal gap between the optimal

policy and the estimated optimal policy when a mistake in the ranking of two arms is done. As the probability of making a mistake in the estimation the model $\boldsymbol{\theta}$ is not null, it exists $c \in (0, \delta)$ such that:

$$\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}}) \geq c \min_{k \in [K], \hat{\theta}^{k+1} > \hat{\theta}^k} \left(\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})\right). \tag{5.69}$$

The minimal gap, between the mean reward of the optimal policy (see equation (5.66)) and a policy where an arm is not well ranked, is obtained when the ranks of arms 2 and 3 are inverted.

$$\begin{aligned}\min_{k \in [K], \hat{\theta}^{k+1} > \hat{\theta}^k} \left(\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})\right) \geq (p_2(1-p_{K+1}) + p_{K+1}(1-p2))\theta^2 + p_2\theta^3 \\ - p_2\theta^2 - (p_2(1-p_{K+1} + p_{K+1}(1-p2))\theta^3\end{aligned} \tag{5.70}$$

Hence we have:

$$\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}}) \geq c_{\boldsymbol{p}}\epsilon, \text{ where } c_{\boldsymbol{p}} > 0. \tag{5.71}$$

Lower bound of the left term. The left term of equation (5.68) is the instantaneous regret of the optimal exploration policy π^*_E. The optimal exploration policy cannot be the optimal policy since estimating ϵ-approximations of arms necessitates to play the same expected number of times the arms, and hence assigning p_1 and p_{K+1} on the same arm, which is not optimal. There are three possibilities:

- p_1 and p_{K+1} are on arm 1:

$$\begin{aligned}\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_E) \geq \quad & p_1\theta^1 + (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^2 \\ & - (p_1(1-p_{K+1}) + p_{K+1}(1-p_1))\theta^1 - p_2\theta^2,\end{aligned}$$

- p_1 and p_{K+1} are on arm $m \in [K] \setminus \{1, 2\}$:

$$\begin{aligned}\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_E) \geq \quad & p_1\theta^1 + p_m\theta^m + (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^2 \\ & - p_2\theta^1 - (p_1(1-p_{K+1}) + p_{K+1}(1-p_1))\theta^m - p_2\theta^2,\end{aligned}$$

- p_1 and p_{K+1} are on arm 2:

$$\begin{aligned}\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_E) \geq \quad & p_1\theta^1 + (p_2(1-p_{K+1}) + p_{K+1}(1-p_2))\theta^2 \\ & - p_2\theta^1 - (p_1(1-p_{K+1}) + p_{K+1}(1-p_1))\theta^2.\end{aligned}$$

Hence we have:

$$\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_E) \geq c_{\boldsymbol{\theta},\boldsymbol{p}}, \tag{5.72}$$

where $c_{\boldsymbol{\theta},\boldsymbol{p}} > 0$ is a constant depending on the problem parameters $\boldsymbol{\theta}$ and $p_1, ..., p_N$.

Lower bound of the regret. Now, injecting the lower bound of $\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_E)$ (equation (5.72)) and the lower bound of $\mu_{\boldsymbol{\theta}}(\pi^*_{\boldsymbol{\theta}}) - \mu_{\boldsymbol{\theta}}(\pi^*_{\hat{\boldsymbol{\theta}}})$ (equation (5.71)) in the pseudo-regret decomposition (equation (5.68)), we obtain:

$$R(T) \geq t^* c_{\boldsymbol{\theta},\boldsymbol{p}} + (T - t^*)c_{\boldsymbol{p}}\epsilon, \tag{5.73}$$

$$\geq t^* c_{\boldsymbol{\theta},\boldsymbol{p}} + T\epsilon\Delta_{\boldsymbol{p}} - t^* c_{\boldsymbol{p}}\epsilon. \tag{5.74}$$

The lower bound of number of samples for finding a bias ϵ of a coin is $\Omega\left(1/\epsilon^2 \log 1/\delta\right)$ [**?**]. At each time step, a maximum of N players are sampled. Hence, the time t^* where π^*_E finds exactly an ϵ-approximation of each arm θ^k is at least:

$$\Omega\left(\frac{K}{N\epsilon^2}\log\frac{1}{\delta}\right) \Leftrightarrow \exists c_1 > 0, t^* = c_1\frac{K}{N\epsilon^2}\log\frac{1}{\delta}. \tag{5.75}$$

We have:

$$R(T) \geq c_1 c_{\boldsymbol{\theta},\boldsymbol{p}}\frac{K}{N\epsilon^2}\log\frac{1}{\delta} + Tc_{\boldsymbol{p}}\epsilon - c_1 c_{\boldsymbol{p}}\epsilon\frac{K}{N\epsilon}\log\frac{1}{\delta}. \tag{5.76}$$

Finally setting $\delta = 1/T$ and $\epsilon = \sqrt{K}/\sqrt[3]{T}$, obtain:

$$E[R(T)] \geq \Omega\left(T^{2/3}\frac{\log T}{N} + T^{2/3} - \frac{K^{1/2}}{N}T^{1/3}\log T\right). \tag{5.77}$$

Hence, we have:

$$E[R(T)] \geq \Omega\left(T^{2/3}\frac{\log T}{N}\right). \tag{5.78}$$

□

Proof of Theorem 33

Applying Algorithm 22 on a model estimate $\hat{\boldsymbol{\theta}}$ returns with a probability $1-\delta$ an α-fair policy in the true model $\boldsymbol{\theta}$:

$$\alpha \geq 1 - \max_{n\in[N]} p_n - \frac{2\|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|_\infty}{\max_{n\in[N]}\frac{\hat{\theta}^{k_n} z^{k_n}}{1-p_n}} \tag{5.79}$$

Proof. Theorem 28 states that the policy returned by Algorithm 22, denoted as $\pi^\dagger$ has the following fairness guarantees:

$$\hat{\alpha} = \frac{\min_{n\in[N]}\mu_{n,\hat{\boldsymbol{\theta}}}(\pi^\dagger)}{\max_{n\in[N]}\mu_{n,\hat{\boldsymbol{\theta}}}(\pi^\dagger)} \geq 1 - \max_{n\in[N]} p_n, \tag{5.80}$$

with $\mu_{n,\hat{\boldsymbol{\theta}}}(\pi^\dagger)$ denoting the expectation of rewards received by player n in estimated model $\hat{\boldsymbol{\theta}}$ when following policy $\pi^\dagger$. We may write it as follows:

$$\mu_{n,\hat{\boldsymbol{\theta}}}(\pi^\dagger) = \hat{\theta}^{k_n} \prod_{n',\text{ s.t. } k_{n'}=k_n}(1-p_{n'}) = \frac{\hat{\theta}^{k_n} z^{k_n}}{1-p_n}. \tag{5.81}$$

We therefore get:

$$\begin{aligned}
\alpha &= \frac{\min_{n\in[N]}\mu_{n,\boldsymbol{\theta}}(\pi^\dagger)}{\max_{n\in[N]}\mu_{n,\boldsymbol{\theta}}(\pi^\dagger)} && (5.82)\\
&= \frac{\min_{n\in[N]}\frac{\theta^{k_n} z^{k_n}}{1-p_n}}{\max_{n\in[N]}\frac{\theta^{k_n} z^{k_n}}{1-p_n}} && (5.83)\\
&\geq \frac{\min_{n\in[N]}\frac{\hat{\theta}^{k_n} z^{k_n}}{1-p_n} - \|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|_\infty}{\max_{n\in[N]}\frac{\hat{\theta}^{k_n} z^{k_n}}{1-p_n} + \|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|_\infty} \qquad \text{since } \frac{z^{k_n}}{1-p_n} \leq 1, \forall n && (5.84)\\
&= \hat{\alpha} - \frac{2\|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|_\infty}{\max_{n\in[N]}\frac{\hat{\theta}^{k_n} z^{k_n}}{1-p_n} + \|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|_\infty} && (5.85)\\
&\geq 1 - \max_{n\in[N]} p_n - \frac{2\|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\|_\infty}{\max_{n\in[N]}\frac{\hat{\theta}^{k_n} z^{k_n}}{1-p_n}} && (5.86)
\end{aligned}$$

Now, Theorem 30 states that with a probability $1-\delta$ Algorithm 23 stops while finding ϵ-approximations of model

$\boldsymbol{\theta}$. Finally, we get:

$$\alpha \geq 1 - \max_{n \in [N]} p_n - \frac{2\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_\infty}{\max_{n \in [N]} \frac{(\theta^{k_n} - \epsilon) z^{k_n}}{1 - p_n}} \tag{5.87}$$

□

Bibliography

[lor, 2021] (2021). LoRa alliance. Available online: https://www.lora-alliance.org (last accessed on 17 June 2021).

[Abbasi-Yadkori et al., 2018] Abbasi-Yadkori, Y., Bartlett, P., Gabillon, V., Malek, A., and Valko, M. (2018). Best of both worlds: Stochastic amp; adversarial best-arm identification. In Proceedings of the 31st Conference On Learning Theory.

[Abbasi-yadkori et al., 2011] Abbasi-yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In Advances in Neural Information Processing Systems.

[Abbasi-Yadkori et al., 2012] Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. (2012). Online-to-confidence-set conversions and application to sparse stochastic bandits. In Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, pages 1–9.

[Abdullah et al., 2019] Abdullah, M., Iqbal, W., and Bukhari, F. (2019). Containers vs virtual machines for auto-scaling multi-tier applications under dynamically increasing workloads. In Intelligent Technologies and Applications.

[Adelantado et al., 2017] Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J., and Watteyne, T. (2017). Understanding the limits of LoRaWAN. IEEE Communications magazine, 55(9):34–40.

[Agarwal et al., 2014] Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R. (2014). Taming the monster: A fast and simple algorithm for contextual bandits. In Proceedings of the 31st International Conference on Machine Learning.

[Agarwal et al., 2009] Agarwal, D., Chen, B.-C., and Elango, P. (2009). Explore/exploit schemes for web content optimization. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM '09, pages 1–10, Washington, DC, USA. IEEE Computer Society.

[Agrawal and Goyal, 2012] Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In Proceedings of the 25th Annual Conference on Learning Theory (COLT).

[Agrawal and Goyal, 2013] Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In ICML (3), pages 127–135.

[Al-Dhuraibi et al., 2018] Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., and Merle, P. (2018). Elasticity in cloud computing: State of the art and research challenges. IEEE Trans. Serv. Comput.

[Alami et al., 2017] Alami, R., Maillard, O., and Féraud, R. (2017). Memory Bandits: a Bayesian approach for the Switching Bandit Problem. In BayesOpt 2017 NIPS Workshop on Bayesian Optimization.

[Alami et al., 2020] Alami, R., Maillard, O., and Féraud, R. (2020). Restarted bayesian online change-point detector achieves optimal detection delay. In International Conference on Machine Learning.

[Ali et al., 2019] Ali, Z., Henna, S., Akhunzada, A., Raza, M., and Kim, S. W. (2019). Performance evaluation of LoRaWAN for green internet of things. IEEE Access, 7:164102–164112.

[Allesiardo and Féraud, 2015] Allesiardo, R. and Féraud, R. (2015). Exp3 with drift detection for the switching bandit problem. In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA 2015).

[Allesiardo and Féraud, 2017] Allesiardo, R. and Féraud, R. (2017). Selection of learning experts. In 2017 International Joint Conference on Neural Networks (IJCNN). IEEE.

[Allesiardo et al., 2014] Allesiardo, R., Féraud, R., and Bouneffouf, D. (2014). A neural networks committee for the contextual bandit problem. In Neural Information Processing - 21st International Conference, ICONIP.

[Allesiardo et al., 2017] Allesiardo, R., Féraud, R., and Maillard, O.-A. (2017). The non-stationary stochastic multi-armed bandit problem. International Journal of Data Science and Analytics.

[Anandkumar et al., 2010] Anandkumar, A., Michael, N., and Tang, A. (2010). Opportunistic spectrum access with multiple users: Learning under competition. In 2010 Proceedings IEEE INFOCOM.

[Anantharam et al., 1987] Anantharam, V., Varaiya, P., and Walrand, J. (1987). Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i: I.i.d. rewards. IEEE Transactions on Automatic Control.

[Anthony and Bartlett, 1999] Anthony, M. and Bartlett, P. L. (1999). Neural Network Learning: Theoretical Foundations. Cambridge University Press, USA, 1st edition.

[Audibert and Bubeck, 2010] Audibert, J.-Y. and Bubeck, S. (2010). Best arm identification in multi-armed bandits. In Proceedings of the 23rd Annual Conference on Learning Theory (COLT).

[Auer et al., 2002a] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. Machine Learning.

[Auer et al., 2002b] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multi-armed bandit problem. SIAM Journal on Computing, 32(1):48–77.

[Augustin et al., 2016] Augustin, A., Yi, J., Clausen, T., and Townsley, W. M. (2016). A study of lora: Long range & low power networks for the internet of things. Sensors, 16(9):1466.

[Avner and Mannor, 2014] Avner, O. and Mannor, S. (2014). Concurrent bandits and cognitive radio networks. In Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I.

[Ayimba et al., 2019] Ayimba, C., Casari, P., and Mancuso, V. (2019). SQLR: short term memory q-learning for elastic provisioning. CoRR.

[Balakrishnan et al., 2019a] Balakrishnan, A., Bouneffouf, D., Mattei, N., and Rossi, F. (2019a). Incorporating behavioral constraints in online AI systems. In The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 3–11. AAAI Press.

[Balakrishnan et al., 2019b] Balakrishnan, A., Bouneffouf, D., Mattei, N., and Rossi, F. (2019b). Using multi-armed bandits to learn ethical priorities for online AI systems. IBM J. Res. Dev., 63(4/5):1:1–1:13.

[Baransi et al., 2014] Baransi, A., Maillard, O.-A., and Mannor, S. (2014). Sub-sampling for multi-armed bandits. In Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I.

[Barrett et al., 2013] Barrett, E., Howley, E., and Duggan, J. (2013). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurr. Comput. Pract. Exp.

[Bellman, 1957] Bellman, R. (1957). Dynamic Programming. Dover Publications.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. Found. Trends Mach. Learn.

[Besson and Kaufmann, 2018] Besson, L. and Kaufmann, E. (2018). Multi-Player Bandits Revisited. In Proceedings of Algorithmic Learning Theory.

[Biau, 2012] Biau, G. (2012). Analysis of a random forests model. Journal Machine Learning Research.

[Bonnefoi et al., 2018] Bonnefoi, R., Besson, L., Moy, C., Kaufmann, E., and Palicot, J. (2018). Multi-armed bandit learning in iot networks: Learning helps even in non-stationary settings. In Marques, P., Radwan, A., Mumtaz, S., Noguet, D., Rodriguez, J., and Gundlach, M., editors, Cognitive Radio Oriented Wireless Networks, pages 173–185. Springer International Publishing.

[Bouneffouf and Féraud, 2016] Bouneffouf, D. and Féraud, R. (2016). Multi-armed bandit problem with known trend. Neurocomputing.

[Bouneffouf et al., 2021a] Bouneffouf, D., Féraud, R., Upadhyay, S., Khazaeni, Y., and Rish, I. (2021a). Double-linear thompson sampling for context-attentive bandits. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[Bouneffouf et al., 2021b] Bouneffouf, D., Féraud, R., Upadhyay, S., Agarwal, M., Khazaeni, Y., and Rish, I. (2021b). Toward skills dialog orchestration with online learning. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[Bouneffouf et al., 2021c] Bouneffouf, D., Féraud, R., Upadhyay, S., Rish, I., and Khazaeni, Y. (2021c). Toward optimal solution for the context-attentive bandit problem. In International Joint Conferences on Artificial Intelligence (IJCAI).

[Bouneffouf et al., 2019] Bouneffouf, D., Parthasarathy, S., Samulowitz, H., and Wistuba, M. (2019). Optimal exploitation of clustering and history information in multi-armed bandit. In Kraus, S., editor, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, pages 2016–2022. ijcai.org.

[Bouneffouf and Rish, 2019] Bouneffouf, D. and Rish, I. (2019). A survey on practical applications of multi-armed and contextual bandits. CoRR, abs/1904.10040.

[Bouneffouf et al., 2017] Bouneffouf, D., Rish, I., Cecchi, G. A., and Féraud, R. (2017). Context attentive bandits: Contextual bandit with restricted context. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pages 1468–1475.

[Boursier and Perchet, 2019a] Boursier, E. and Perchet, V. (2019a). Sic-mmab: Synchronisation involves communication in multiplayer multi-armed bandits. In Advances in Neural Information Processing Systems.

[Boursier and Perchet, 2019b] Boursier, E. and Perchet, V. (2019b). Sic-mmab: Synchronisation involves communication in multiplayer multi-armed bandits. In Advances in Neural Information Processing Systems 32, pages 12048–12057.

[Boursier et al., 2020] Boursier, E., Perchet, V., Kaufmann, E., and Mehrabian, A. (2020). A practical algorithm for multiplayer bandits when arm means vary among players. In AISTATS.

[Breiman, 2001] Breiman, L. (2001). Random forests. Machine Learning.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and Regression Trees. Wadsworth International Group.

[Bubeck and Cesa-Bianchi, 2012] Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Foundations and Trends® in Machine Learning.

[Bubeck et al., 2019] Bubeck, S., Li, Y., Peres, Y., and Sellke, M. (2019). Non-stochastic multi-player multi-armed bandits: Optimal rate with collision information, sublinear without.

[Bubeck et al., 2009] Bubeck, S., Munos, R., and Stoltz, G. (2009). Pure exploration in multi-armed bandits problems. In Algorithmic Learning Theory.

[Bubeck and Slivkins, 2012] Bubeck, S. and Slivkins, A. (2012). The best of both worlds: Stochastic and adversarial bandits. In COLT 2012 - The 25th Annual Conference on Learning Theory.

[Cano et al., 2018] Cano, I., Chen, L., Fonseca, P., Chen, T., Cheah, C., Gupta, K., Chandra, R., and Krishnamurthy, A. (2018). ADARES: Adaptive resource management for virtual machines. arXiv.

[Carpentier and Munos, 2012] Carpentier, A. and Munos, R. (2012). Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, pages 190–198.

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). Prediction, Learning, and Games. Cambridge University Press, USA.

[Cesa-Bianchi et al., 2011] Cesa-Bianchi, N., Shalev-Shwartz, S., and Shamir, O. (2011). Efficient learning with partially observed attributes. J. Mach. Learn. Res., 12(null):2857–2878.

[Chakrabarti et al., 2008] Chakrabarti, D., Kumar, R., Radlinski, F., and Upfal, E. (2008). Mortal multi-armed bandits. In Advances in Neural Information Processing Systems.

[Chakraborty et al., 2017] Chakraborty, M., Chua, K. Y. P., Das, S., and Juba, B. (2017). Coordinated versus decentralized exploration in multi-agent multi-armed bandits. In Proceedings of the 26th International Joint Conference on Artificial Intelligence.

[Chapelle and Li, 2011] Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In Advances in neural information processing systems, pages 2249–2257.

[Chaudhari et al., 2020] Chaudhari, B. S., Zennaro, M., and Borkar, S. (2020). LPWAN technologies: Emerging application characteristics, requirements, and design considerations. Future Internet, 12(3):46.

[Chu, 1955] Chu, J. T. (1955). On bounds for the normal integral. Biometrika.

[Chu et al., 2011] Chu, W., Li, L., Reyzin, L., and Schapire, R. (2011). Contextual bandits with linear payoff functions. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.

[Coutinho et al., 2015] Coutinho, E. F., de Carvalho Sousa, F. R., Rego, P. A. L., Gomes, D. G., and de Souza, J. N. (2015). Elasticity in cloud computing: a survey. Ann. des Télécommunications.

[Cover and Thomas, 2005] Cover, T. M. and Thomas, J. A. (2005). Entropy, Relative Entropy, and Mutual Information, chapter 2, pages 13–55.

[Dakdouk et al., 2021] Dakdouk, H., Féraud, R., Laroche, R., Varsier, N., and Maillé, P. (2021). Collaborative Exploration and Exploitation in massively Multi-Player Bandits. working paper or preprint.

[Dakdouk et al., 2020] Dakdouk, H., Féraud, R., Varsier, N., and Maillé, P. (2020). Collaborative exploration in stochastic multi-player bandits. In Asian Conference on Machine Learning.

[Dakdouk et al., 2018] Dakdouk, H., Tarazona, E., Alami, R., Féraud, R., Papadopoulos, G. Z., and Maillé, P. (2018). Reinforcement learning techniques for optimized channel hopping in ieee 802.15.4-tsch networks. In Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWIM '18.

[Dean and Ghemawat, 2008] Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. Communications of the ACM.

[Delande et al., 2021] Delande, D., Stolf, P., Féraud, R., Pierson, J.-M., and Bottaro, A. (2021). Horizontal scaling in cloud using contextual bandits. In European Conference on Parallel Processing.

[Deme et al., 2013] Deme, A., Dajab, D., Buba Bajoga, M., and Choji, D. (2013). Hata-okumura model computer analysis for path loss determination at 900mhz for maiduguri, nigeria. Mathematical Theory and Modeling, 3(3):1–9.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Domingos and Hulten, 2000] Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[Dubey and Pentland, 2020] Dubey, A. and Pentland, A. (2020). Differentially-private federated linear bandits. Advances in Neural Information Processing Systems.

[Duchi et al., 2014] Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2014). Privacy aware learning. J. ACM.

[Dudík et al., 2011] Dudík, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., and Zhang, T. (2011). Efficient optimal learning for contextual bandits. CoRR.

[Dudik et al., 2011] Dudik, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., and Zhang, T. (2011). Efficient optimal learning for contextual bandits. In Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence.

[Durand and Gagné, 2014] Durand, A. and Gagné, C. (2014). Thompson sampling for combinatorial bandits and its application to online feature selection. In AAAI, Workshop Sequential Decision-Making with Big Data:.

[Dutreilh et al., 2011] Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., and Truck, I. (2011). Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow. ICAS.

[Dwork et al., 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Proceedings of the Third Conference on Theory of Cryptography.

[ETSI, 2005] ETSI, T. (June 2005). Digital cellular telecommunications system (phase 2+); radio network planning aspects (3gpp tr 03.30 version 8.4.0 release 1999). ETSI TR 101 362 V8.4.0.

[Even-Dar et al., 2002] Even-Dar, E., Mannor, S., and Mansour, Y. (2002). Pac bounds for multi-armed bandit and markov decision processes. In Proceedings of the 15th Annual Conference on Computational Learning Theory.

[Even-Dar et al., 2006] Even-Dar, E., Mannor, S., and Mansour, Y. (2006). Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. Journal Machine Learning Research.

[Fernández-Delgado et al., 2014] Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research.

[Ferreira et al., 2006] Ferreira, L., Kuipers, M., Rodrigues, C., and Correia, L. M. (2006). Characterisation of signal penetration into buildings for GSM and UMTS. In 3rd International Symposium on Wireless Communication Systems, pages 63–67. IEEE.

[Filippi et al., 2010] Filippi, S., Cappé, O., Garivier, A., and Szepesvári, C. (2010). Parametric bandits: The generalized linear case. In Advances in Neural Information Processing Systems.

[Foster et al., 2016] Foster, D., Kale, S., and Karloff, H. (2016). Online sparse linear regression. In COLT.

[Féraud, 1997] Féraud, R. (1997). Un modele connexionniste utilisant un principe de longueur de description minimale : application a la detection de visages. PhD thesis.

[Féraud et al., 2019] Féraud, R., Alami, R., and Laroche, R. (2019). Decentralized exploration in multi-armed bandits. In Proceedings of the 36th International Conference on Machine Learning.

[Féraud et al., 2016] Féraud, R., Allesiardo, R., Urvoy, T., and Clérot, F. (2016). Random forest for the contextual bandit problem. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics.

[Féraud et al., 2008] Féraud, R., Boullé, M., Clérot, F., and Fessant, F. (2008). Vers l'exploitation de grandes masses de données. In Extraction et Gestion des Connaissances.

[Féraud et al., 2010] Féraud, R., Boullé, M., Clérot, F., Fessant, F., and Lemaire, V. (2010). The orange customer analysis platform. In Industrial Conference on Data Mining.

[Féraud and Urvoy, 2012] Féraud, R. and Urvoy, T. (2012). A stochastic bandit algorithm for scratch games. In Proceedings of the Asian Conference on Machine Learning.

[Féraud and Urvoy, 2013] Féraud, R. and Urvoy, T. (2013). Exploration and exploitation of scratch games. Machine Learning.

[Gabillon et al., 2012] Gabillon, V., Ghavamzadeh, M., and Lazaric, A. (2012). In Advances in Neural Information Processing Systems.

[Gajane et al., 2018] Gajane, P., Urvoy, T., and Kaufmann, E. (2018). Corrupt bandits for preserving local privacy.

[Ganta et al., 2008] Ganta, S. R., Kasiviswanathan, S. P., and Smith, A. (2008). Composition attacks and auxiliary information in data privacy. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[Gari et al., 2020] Gari, Y., Monge, D. A., Pacini, E., Mateos, C., and Garino, C. G. (2020). Reinforcement learning-based autoscaling of workflows in the cloud: A survey. CoRR.

[Garivier et al., 2016] Garivier, A., Lattimore, T., and Kaufmann, E. (2016). On explore-then-commit strategies. In Advances in Neural Information Processing Systems.

[Garivier and Moulines, 2011] Garivier, A. and Moulines, E. (2011). On upper-confidence bound policies for switching bandit problems. In Algorithmic Learning Theory.

[Ghoslya, 2021] Ghoslya, S. (2021). How does LoRaWAN adaptive data rate work? last accessed June, 23rd 2021.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, AISTATS.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. ICASSP.

[Hanawal and Darak, 2018] Hanawal, M. K. and Darak, S. J. (2018). Multi-player bandits: A trekking approach. arXiv preprint arXiv:1809.06040.

[Hanna et al., 2022] Hanna, O. A., Yang, L. F., and Fragouli, C. (2022). Solving multi-arm bandit using a few bits of communication. In International Conference on Artificial Intelligence and Statistics (AISTATS).

[Hartland and M., 2006] Hartland, C., B. N. G. S. T. O. S. and M. (2006). Multi-armed bandit, dynamic environments and meta-bandits. Online Trading of Exploration and Exploitation Workshop.

[Hernández-Lobato et al., 2017] Hernández-Lobato, J. M., Requeima, J., Pyzer-Knapp, E. O., and Aspuru-Guzik, A. (2017). Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In Proceedings of the 34th International Conference on Machine Learning - Volume 70.

[Hillel et al., 2013] Hillel, E., Karnin, Z., Koren, T., Lempel, R., and Somekh, O. (2013). Distributed exploration in multi-armed bandits. In Proceedings of the 26th International Conference on Neural Information Processing Systems.

[Hoeffding, 1994] Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In The collected works of Wassily Hoeffding, pages 409–426. Springer.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks.

[Horvitz and Thompson, 1952] Horvitz, D. and Thompson, D. (1952). A generalization of sampling without replacement from a finite universe. Journal of the American Statistical Association.

[Hwang et al., 6 01] Hwang, K., Bai, X., Shi, Y., Li, M., Chen, W., and Wu, Y. (2016-01). Cloud performance modeling with benchmark evaluation of elastic scaling strategies. IEEE Transactions on Parallel and Distributed Systems.

[Jaksch et al., 2010] Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. Journal of Machine Learning Research.

[Jin et al., 2018] Jin, Y., Bouzid, M., Kostadinov, D., and Aghasaryan, A. (2018). Model-free resource management of cloud-based applications using reinforcement learning. In ICIN.

[Kakade et al., 2008a] Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. (2008a). Efficient bandit algorithms for online multiclass prediction. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 440–447, New York, NY, USA. ACM.

[Kakade et al., 2008b] Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. (2008b). Efficient bandit algorithms for online multiclass prediction. In Proceedings of the 25th International Conference on Machine Learning.

[Kalyanakrishnan et al., 2012] Kalyanakrishnan, S., Tewari, A., Auer, P., and Stone, P. (2012). Pac subset selection in stochastic multi-armed bandits. In Proceedings of the 29th International Coference on International Conference on Machine Learning.

[Kanade et al., 2012] Kanade, V., Liu, Z., and Radunovic, B. (2012). Distributed non-stochastic experts. In Advances in Neural Information Processing Systems.

[Kaufmann et al., 2016] Kaufmann, E., Cappé, O., and Garivier, A. (2016). On the complexity of best arm identification in multi-armed bandit models. Journal of Machine Learning Research.

[Kaufmann and Kalyanakrishnan, 2013] Kaufmann, E. and Kalyanakrishnan, S. (2013). Information complexity in bandit subset selection. In Proceedings of the 26th Annual Conference on Learning Theory.

[Kaufmann et al., 2012] Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In Algorithmic Learning Theory.

[Kerkouche et al., 2018] Kerkouche, R., Alami, R., Féraud, R., Varsier, N., and Maillé, P. (2018). Node-based optimization of lora transmissions with multi-armed bandit algorithms. In 2018 25th International Conference on Telecommunications (ICT).

[Khatua et al., 2010] Khatua, S., Ghosh, A., and Mukherjee, N. (2010). Optimizing the utilization of virtual resources in cloud environment. In VECIMS.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

[Kleinberg et al., 2010] Kleinberg, R., Niculescu-Mizil, A., and Sharma, Y. (2010). Regret bounds for sleeping experts and bandits. Machine Learning.

[Kocsis and Szepesvári, 2006a] Kocsis, L. and Szepesvári, C. (2006a). Bandit based monte-carlo planning. In Proceedings of the 17th European Conference on Machine Learning.

[Kocsis and Szepesvári, 2006b] Kocsis, L. and Szepesvári, C. (2006b). Discounted ucb. In 2nd PASCAL Challenges Workshop.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems.

[Kufakunesu et al., 2020] Kufakunesu, R., Hancke, G. P., and Abu-Mahfouz, A. M. (2020). A survey on adaptive data rate optimization in LoRaWAN: Recent solutions and major challenges. Sensors, 20(18):5044.

[Lai and Robbins, 1985] Lai, T. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics, 6(1):4–22.

[Landgren et al., 2016] Landgren, P., Srivastava, V., and Leonard, N. E. (2016). Distributed cooperative decision-making in multiarmed bandits: Frequentist and bayesian algorithms. 2016 IEEE 55th Conference on Decision and Control (CDC).

[Langford and Strehl, 2008] Langford, J. and Strehl, A. (2008). Exploration scavenging. In ICML.

[Langford and Zhang, 2007] Langford, J. and Zhang, T. (2007). The epoch-greedy algorithm for multi-armed bandits with side information. In Advances in Neural Information Processing Systems.

[Laroche and Feraud, 2018] Laroche, R. and Feraud, R. (2018). Reinforcement learning algorithm selection. In International Conference on Learning Representations (ICLR).

[Lattimore and Szepesvári, 2020] Lattimore, T. and Szepesvári, C. (2020). Bandit Algorithms. Cambridge University Press.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551.

[Li et al., 2010a] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010a). A contextual-bandit approach to personalized news article recommendation. WWW '10.

[Li et al., 2010b] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010b). A contextual-bandit approach to personalized news article recommendation. CoRR.

[Liu and Zhao, 2010] Liu, K. and Zhao, Q. (2010). Distributed learning in multi-armed bandit with multiple players. IEEE Transactions on Signal Processing.

[Lorido-Botran et al., 2014] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing.

[Lugosi and Mehrabian, 2018] Lugosi, G. and Mehrabian, A. (2018). Multiplayer bandits without observing collision information.

[Mannor and Tsitsiklis, 2004] Mannor, S. and Tsitsiklis, J. N. (2004). The sample complexity of exploration in the multi-armed bandit problem. Journal Machine Learning Research.

[Marini et al., 2021] Marini, R., Cerroni, W., and Buratti, C. (2021). A novel collision-aware adaptive data rate algorithm for lorawan networks. IEEE Internet of Things Journal, 8(4):2670–2680.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., and al (2015). Human-level control through deep reinforcement learning. Nature.

[Mousavi, 2010] Mousavi, N. (2010). How tight is chernoff bound. Unpublished manuscript.

[Nayyar et al., 2018] Nayyar, N., Kalathil, D. M., and Jain, R. (2018). On regret-optimal learning in decentralized multiplayer multi-armed bandits. IEEE Transactions on Control of Network Systems.

[Neu, 2015] Neu, G. (2015). Explore no more: Improved high-probability regret bounds for non-stochastic bandits. In Advances in Neural Information Processing Systems.

[Newman, 2019] Newman, D. (2019). Return on iot: Dealing with the iot skills gap. Available online: https://www.forbes.com/sites/danielnewman/ 2019/07/30/return-on-iot-dealing-with-the-iot-skills-gap/27017efb7091 (accessed on 16 June 2021).

[Nguyen et al., 2013] Nguyen, H., Shen, Z., Gu, X., Subbiah, S., and Wilkes, J. (2013). AGILE: Elastic distributed resource scaling for infrastructure-as-a-service. In ICAC.

[Nikravesh et al., 2015] Nikravesh, A. Y., Ajila, S. A., and Lung, C. (2015). Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In SEAMS.

[Oswal et al., 2020] Oswal, U., Bhargava, A., and Nowak, R. (2020). Linear bandits with feature feedback. AAAI.

[Papadimitriou and Tsitsiklis, 1987] Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of markov decision processes. Math. Oper. Res.

[Pascual et al., 2018] Pascual, J. A., Lozano, J. A., and Miguel-Alonso, J. (2018). Effects of reducing VMs management times on elastic applications. Journal of Grid Computing.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543.

[Perchet et al., 2015] Perchet, V., Rigollet, P., Chassang, S., and Snowberg, E. (2015). Batched bandit problems. In Proceedings of The 28th Conference on Learning Theory.

[Perešíni and Krajčovič, 2017] Perešíni, O. and Krajčovič, T. (2017). More efficient iot communication through lora network with lora@fiit and stiot protocols. In 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), pages 1–6.

[Puterman, 1994] Puterman, M. L. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley Sons, Inc.

[Qin et al., 2014] Qin, L., Chen, S., and Zhu, X. (2014). Contextual combinatorial bandit and its application on diversified online recommendation. In Proceedings of the 2014 SIAM International Conference on Data Mining, pages 461–469. SIAM.

[Qu et al., 2018] Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. ACM Computing Surveys.

[Raina et al., 2009] Raina, R., Madhavan, A., and Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. ICML.

[Rodriguez et al., 2013] Rodriguez, I., Nguyen, H. C., Jørgensen, N. T., Sørensen, T. B., Elling, J., Gentsch, M. B., and Mogensen, P. (2013). Path loss validation for urban micro cell scenarios at 3.5 ghz compared to 1.9 ghz. In IEEE global communications conference (GLOBECOM), pages 3942–3947.

[Rosenski et al., 2016] Rosenski, J., Shamir, O., and Szlak, L. (2016). Multi-player bandits – a musical chairs approach. In ICML.

[Rumelhart et al., 1986] Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. MIT Press, Cambridge, MA, USA.

[Santivanez et al., 2006] Santivanez, C., Ramanathan, R., Partridge, C., Krishnan, R., Condell, M., and Polit, S. (2006). Opportunistic spectrum access: Challenges, architecture, protocols. In Proceedings of the 2nd Annual International Workshop on Wireless Internet, WICON '06.

[Schuler et al., 2020] Schuler, L., Jamil, S., and Kühl, N. (2020). AI-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments. arXiv.

[Seldin and Slivkins, 2014] Seldin, Y. and Slivkins, A. (2014). One practical algorithm for both stochastic and adversarial bandits. In 31th Intl. Conf. on Machine Learning (ICML).

[Semtech,] Semtech. Understanding the LoRa adaptive data rate. available on: semtech.com/LoRa, December 2019.

[Semtech Corporation, 2021] Semtech Corporation (2021). Lorawan – simple rate adaptation recommended algorithm. Online; accessed 25-November-2021.

[Serfling, 1974] Serfling, R. (1974). Probability inequalities for the sum in sampling without replacement. In The Annals of Statistics, Vol 2, No.1, pages 39–48.

[Sharabiani et al., 2015] Sharabiani, A., Bress, A., Douzali, E., and Darabi, H. (2015). Revisiting warfarin dosing using machine learning techniques. Computational and mathematical methods in medicine, 2015.

[Shariffdeen et al., 2016] Shariffdeen, R. S., Munasinghe, D. T. S. P., Bhathiya, H. S., Bandara, U. K. J. U., and Bandara, H. M. N. D. (2016). Adaptive workload prediction for proactive auto scaling in PaaS systems. In CloudTech.

[Shi and Shen, 2021] Shi, C. and Shen, C. (2021). Federated multi-armed bandits. In Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI).

[Shi et al., 2020] Shi, C., Xiong, W., Shen, C., and Yang, J. (2020). Decentralized multi-player multi-armed bandits with no collision information. In International Conference on Artificial Intelligence and Statistics, pages 1519–1528. PMLR.

[Silver et al., 2017] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2017). Mastering the game of go with deep neural networks and tree search. Nature.

[Singh et al., 2019] Singh, P., Gupta, P., Jyoti, K., and Nayyar, A. (2019). Research on auto-scaling of web applications in cloud: Survey, trends and future directions. Scalable Computing: Practice and Experience.

[Slud, 1977] Slud, E. V. (1977). Distribution inequalities for the binomial law. The Annals of Probability.

[Soare et al., 2014] Soare, M., Lazaric, A., and Munos, R. (2014). Best-arm identification in linear bandits. In Advances in Neural Information Processing Systems.

[Sornin and Yegin, 2018] Sornin, N. and Yegin, A. (July 2018). Lorawan^{tm} specification, v1.0.3.

[Srinivas et al., 2009] Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv preprint arXiv:0912.3995.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). Reinforcement Learning: An Introduction. The MIT Press, second edition.

[Sweeney, 2002] Sweeney, L. (2002). K-anonymity: A model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst.

[Szorenyi et al., 2013] Szorenyi, B., Busa-Fekete, R., Hegedus, I., Ormandi, R., Jelasity, M., and Kegl, B. (2013). Gossip-based distributed stochastic bandit algorithms. In Proceedings of the 30th International Conference on Machine Learning.

[Tadakamalla and Menasce, 2018] Tadakamalla, V. and Menasce, D. A. (2018). Model-driven elasticity control for multi-server queues under traffic surges in cloud environments. In ICAC.

[Thompson, 1933] Thompson, W. R. (1933). ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES. Biometrika, 25(3-4):285–294.

[Tomas et al., 2013] Tomas, M., Kai, C., Greg, C., and Jeffrey, D. (2013). Efficient estimation of word representations in vector space.

[Toslali et al., 2020] Toslali, M., Parthasarathy, S., Oliveira, F., and Coskun, A. K. (2020). JACKPOT: Online experimentation of cloud microservices. In HotCloud.

[Urdaneta et al., 2009] Urdaneta, G., Pierre, G., and van Steen, M. (2009). Wikipedia workload analysis for decentralized hosting. Elsevier Computer Networks.

[Urvoy et al., 2013] Urvoy, T., Clerot, F., Féraud, R., and Naamane, S. (2013). Generic exploration and k-armed voting bandits. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28.

[Valach and Macko, 2022] Valach, A. and Macko, D. (2022). Upper confidence bound based communication parameters selection to improve scalability of lora@fiit communication. IEEE Sensors Journal, 22(12):12415–12427.

[Valiant, 1984] Valiant, L. G. (1984). A theory of the learnable. In Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing.

[Valko et al., 2013] Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. (2013). Finite-time analysis of kernelised contextual bandits. In Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence.

[Vapnik, 1995] Vapnik, V. N. (1995). The nature of statistical learning theory. Springer-Verlag New York, Inc.

[Varsier and Schwoerer, 2017a] Varsier, N. and Schwoerer, J. (2017a). Capacity limits of LoRaWAN technology for smart metering applications. In IEEE international conference on communications (ICC), pages 1–6.

[Varsier and Schwoerer, 2017b] Varsier, N. and Schwoerer, J. (2017b). Capacity limits of lorawan technology for smart metering applications. In 2017 IEEE international conference on communications (ICC), pages 1–6. IEEE.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[Wang et al., 2020] Wang, P.-A., Proutiere, A., Ariu, K., Jedra, Y., and Russo, A. (2020). Optimal algorithms for multiplayer multi-armed bandits. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics.

[Waret et al., 2018] Waret, A., Kaneko, M., Guitton, A., and El Rachkidy, N. (2018). Lora throughput analysis with imperfect spreading factor orthogonality. IEEE Wireless Communications Letters, 8(2):408–411.

[Wei et al., 2019] Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L., and Meng, X. (2019). A reinforcement learning based auto-scaling approach for SaaS providers in dynamic cloud environment. Mathematical Problems in Engineering.

[Xu et al., 2020] Xu, H., Liu, Y., Lau, W. C., Zeng, T., Guo, J., and Liu, A. X. (2020). Online resource allocation with machine variability: A bandit perspective. IEEE/ACM Transactions on Networking.

[Yu and Mannor, 2009] Yu, J. Y. and Mannor, S. (2009). Piecewise-stationary bandit problems with side observations. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML.

[Zhou et al., 2020] Zhou, D., Li, L., and Gu, Q. (2020). Neural contextual bandits with UCB-based exploration. In Proceedings of the 37th International Conference on Machine Learning.

Titre: De l'intérêt des algorithmes de bandits dans le monde digital

Mots clés: algorithmes de bandits, environnement non-stationnaire, bandits massivement multi-joueurs

Résumé: Dans le monde digital de plus en plus d'agents autonomes prennent des décisions automatiques pour optimiser un critère en apprenant de leurs décisions passés. Leur multiplication rapide implique que ces agents vont commencer à interagir entre eux, alors que les algorithmes qu'ils utilisent n'ont pas forcément été conçus pour cela. Des comportements émergents inattendus pourraient donc survenir. Dans cette thèse, nous contribuons à un premier pas vers le contrôle d'un grand nombre d'agents autonomes, qui apprennent en interagissant avec un environnement inconnu, mais aussi avec d'autres agents. Nous nous concentrons sur les algorithmes de bandits: un agent cherche à minimiser son regret par rapport à la meilleure politique possible en choisissant les meilleures actions. Comme l'agent n'observe que le résultat des actions qu'il a prises, il doit résoudre le dilemme *exploration-exploitation*: faut-il choisir l'action dont le résultat est le plus mal connu, ou jouer l'action dont le résultat est empiriquement le meilleur ? Nos contributions commencent par l'étude des bandits lorsque l'environnement évolue, ce qui est souvent le cas dans les applications. Nous proposons ensuite des contributions sur les applications des bandits contextuels dans le monde digital: l'allocation dynamique de ressources pour le cloud computing, l'optimisation des campagnes marketing, ou le dialogue automatique. Nous posons ensuite le problème des bandits massivement décentralisés pour résoudre un problème très commun dans le monde digital, l'A/B testing, mais avec la contrainte de respecter la vie privée des utilisateurs. Finalement, pour l'optimisation des communications dans l'Internet des objets, nous proposons les bandits massivement multi-joueurs.

Title: On the relevance of bandit algorithms in the digital world

Keywords: bandit algorithms, non-stationary environment, massively multi-player bandits

Abstract: In the digital world, more and more autonomous agents make automatic decisions to optimize a criterion by learning from their past decisions. Their rapid multiplication implies that these agents are going to interact with each other, whereas the algorithms they use were not necessarily designed for this. Unexpected emergent behaviors could therefore occur. In this thesis, we contribute to a first step towards the control of a large number of autonomous agents, which learn by interacting with an unknown environment, but also with other agents. We focus on bandit algorithms: an agent aims to minimize its regret with respect to the best possible policy by choosing the best actions. As the agent observes only the outcomes of the actions it has chosen, it must handle the *exploration-exploitation* dilemma: should one choose the action whose outcome is loosely estimated, or play the action whose result is empirically the best? Our contributions begin with the study of bandits in evolving environment, which is often the case in applications. We then propose contributions on the applications of contextual bandits in the digital world: dynamic allocation of resources for cloud computing, optimization of marketing campaigns, or automatic dialogue. We then state the problem of massively decentralized bandits to handle a very common problem in the digital world, A/B testing, but with the constraint of guarantying privacy. Finally, for optimizing communications in Internet of Things, we propose the massively multiplayer bandits.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France